4D v11 SQL

Mise à jour
Windows®/Mac OS®

4D v11 SQL
Mise à jour

Sommaire

1 Introduction

Bienvenue dans 4D v11 SQL, la nouvelle version de l'environnement de développement 4D et 4D Server.

4e Dimension devient 4D

Afin d'accompagner et de faciliter son expansion internationale, 4D a unifié les noms des produits de la gamme v11 SQL :

- Le nom générique "4e Dimension" (dans les pays francophones) ou "4th Dimension" (dans le reste du monde) est remplacé par **4D**.
- L'application 4e Dimension monoposte / 4th Dimension *single user* est désormais appelée **4D Developer**.

La gamme 4D v11 SQL est désormais composée des applications 4D Developer, 4D Server et 4D Client.

Cette appellation sera progressivement mise à jour dans tous les documents techniques et commerciaux de 4D.

Présentation

La version 11 de 4D apporte de nombreuses nouveautés, aussi bien au niveau de l'architecture interne du programme que des fonctions mises à la disposition des développeurs. Ces nouveautés s'articulent autour des axes suivants :

En premier lieu, le **moteur de base de données** de 4D a été en grande partie réécrit en version 11. Plus puissant, plus performant, ce moteur nouvelle génération repousse toutes les limites et permet de nombreuses avancées pour le bénéfice immédiat de vos applications

sans aucune modification de code, comme par exemple la recherche en texte intégral.

4D comporte également un nouveau **moteur SQL complet** permettant de traiter tout type de requête SQL. L'intégration du langage SQL a été effectuée au coeur du nouveau moteur de 4D, pour un accès direct à la structure et aux données.

De nouvelles **fonctions utilitaires de maintenance et de référencement** ont été ajoutées dans 4D v11 : nouveau Centre de sécurité et de maintenance, recherche et remplacement dans la base, etc. En conséquence, les programmes 4D Insider et 4D Tools ne sont plus nécessaires et ne sont plus fournis avec cette version.

L'**interface et l'ergonomie** de l'environnement de travail de 4D ont été modifiées de manière à faciliter la prise en main et l'accès aux fonctions du programme. Les menus des modes Utilisation et Structure ont été fusionnés, 4D comporte désormais le **mode Développement** et le **mode Application**.

Enfin, l'**atelier de développement a été enrichi** : de nombreuses fonctions ont été ajoutées et modifiées en mode Développement et dans le langage de 4D. Ces nouveautés concernent notamment la gestion des composants 4D, l'éditeur de structure, ainsi que les listes hiérarchiques, les menus, les listboxes, etc.

Contenu de ce manuel

Ce manuel se compose des chapitres suivants :

- **Migration** Ce chapitre décrit les principes et étapes de la conversion des anciennes bases en 4D v11, ainsi que les mécanismes obsolètes.
- **Architecture des bases de données 4D** Ce chapitre décrit les nouveautés relatives à la mise en place et à l'exploitation du nouveau moteur de bases de données dans 4D v11, les nouveaux composants, ainsi que les modifications liées à la gestion des fichiers.
- **Interface et navigation en mode Développement** Ce chapitre présente l'environnement de travail unifié de 4D v11 ainsi que les nouvelles fonctions globales qu'il propose, relatives à l'Explorateur, au déplacement d'objets, rechercher et remplacer, feuilles de style, etc.

- **Editeur de structure** Ce chapitre détaille les modifications apportées à l'éditeur de structure de 4D v11.
- **Editeur de méthodes** Ce chapitre détaille les modifications apportées à l'éditeur de méthodes de 4D v11.
- **Formulaires et objets** Ce chapitre détaille les nouveautés relatives aux formulaires de 4D et aux objets qu'ils contiennent.
- **Editeur de menus** Ce chapitre détaille les modifications apportées à l'éditeur de menus de 4D v11.
- **Centre de sécurité et de maintenance** Ce chapitre décrit cette nouvelle boîte de dialogue, centralisant les fonctions de contrôle et de sauvegarde des bases de données 4D.
- **Moteur SQL de 4D** Ce chapitre détaille le fonctionnement du nouveau moteur SQL de 4D.
- **Serveur Web** Ce chapitre décrit les modifications apportées au fonctionnement du serveur Web intégré de 4D v11.
- **Langage** Ce chapitre liste les nouvelles commandes et les commandes modifiées dans le langage de 4D v11.

Configuration minimale

Les applications de la gamme 4D v11 SQL requièrent au minimum les configurations suivantes :

	Windows	**Mac OS**
Ordinateur	Compatible PC avec microprocesseur Pentium III	Macintosh avec microprocesseur Intel ou Power PC
Système d'exploitation	Windows Vista, Windows XP	Mac OS version 10.4.5 et ultérieure
Mémoire minimale	512 Mo	512 Mo
Mémoire conseillée	1 Go	1 Go
Résolution écran	1280*1024 pixels	

2 Migration des bases en version précédente

Les bases de données créées avec des versions 6.x, 2003.x et 2004.x de 4e Dimension ou de 4D Server sont compatibles avec 4D version 11 (fichier de structure et fichier de données). Ces bases doivent toutefois être converties en version 11 et ne pourront plus être ouvertes avec leur version d'origine.

Les modifications structurelles apportées au niveau du moteur de la base de données de 4D v11 nécessitent une conversion en profondeur de la structure et des données. Cette conversion s'effectue via un assistant spécifique (cf. paragraphe ci-dessous). Par sécurité, l'assistant effectue une copie de la base d'origine avant sa conversion, de manière à ce que vous puissiez à tout moment revenir en arrière.

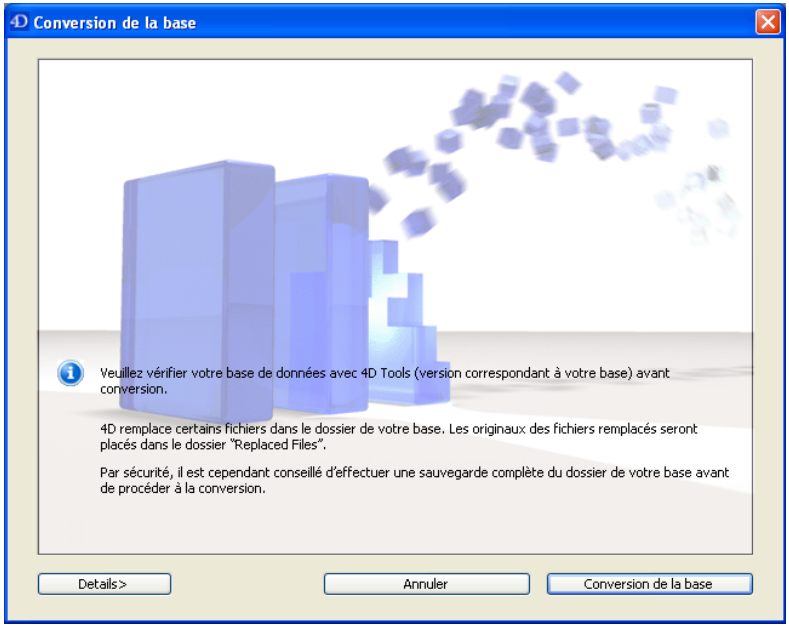
En outre, certains mécanismes obsolètes ou anciens ne sont plus pris en charge et seront supprimés ou remplacés au cours de la conversion (cf. paragraphe "Mécanismes non conservés", page 28).

Conversion des anciennes bases : cas général

Vous pouvez convertir tout fichier de structure interprété. Le fichier peut contenir le code compilé ; dans ce cas, il sera nécessaire de recompiler la base après conversion.

Il est conseillé de vérifier l'intégrité de la base avant la conversion à l'aide de l'utilitaire 4D Tools (compactage, vérification et, éventuellement, réparation de la base). Dans le cas contraire, si une anomalie est détectée durant la conversion, la procédure sera stoppée et l'assistant de conversion vous invitera à utiliser 4D Tools (version correspondant à celle de la base d'origine).

Pour convertir une ancienne base, il suffit de la sélectionner dans la boîte de dialogue d'ouverture de 4D v11. L'assistant de conversion apparaît automatiquement :



Cliquez sur le bouton **Conversion de la base** pour débuter le processus standard de conversion de la base et de son fichier de données. Vous pouvez également visualiser ou modifier les paramètres par défaut utilisés pour la conversion. Pour cela, cliquez sur le bouton **Détails >** (cf. paragraphe suivant).

Par mesure de sécurité, l'assistant de conversion effectuera systématiquement une copie complète de la base (structure et données) avant de débuter l'opération. En cas d'échec de la conversion, vous retrouverez ainsi votre base d'origine intacte. La duplication de la base nécessite davantage d'espace sur votre disque. Si l'espace disponible est insuffisant, un message d'alerte vous le signale et la conversion est stoppée.

Les fichiers d'origine sont copiés dans un dossier nommé **Replaced Files (Conversion)**, créé à côté des fichiers originaux. Si le fichier de données se trouve dans le même dossier que le fichier de structure, le dossier **Replaced Files (Conversion)** contient les deux fichiers d'origine. Si le fichier de données se trouve dans un autre dossier ou sur un autre

volume, la conversion crée un dossier **Replaced Files (Conversion)** pour chaque fichier.

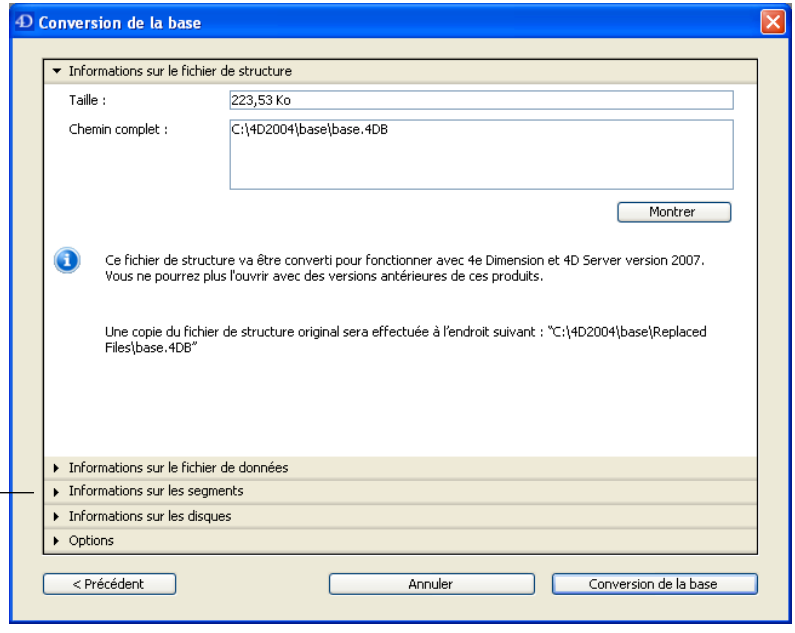
Note Le fichier d'historique courant de la base convertie est également recopié dans le dossier **Replaced Files (Conversion)** et un nouveau fichier d'historique vierge est créé.

Si l'assistant a rencontré des erreurs non critiques durant la conversion, elles sont consignées dans le fichier d'historique de la conversion, nommé *DataConversion_Log.log* et placé à côté du fichier de structure converti.

Note Les préférences utilisateur définies pour le débogueur (points d'arrêt, expressions enregistrées, etc.) sont réinitialisées au cours de la conversion.

Visualiser ou modifier les paramètres de conversion

Si vous souhaitez visualiser ou modifier certains paramètres avant d'effectuer la conversion, cliquez sur le bouton **Détails** >. La boîte de dialogue affiche alors diverses informations concernant les paramètres de conversion. Vous accédez aux pages d'information en cliquant sur les zones correspondantes :



Pages d'information

- **Informations sur le fichier de structure** : cette page indique la taille et l'emplacement du fichier de structure d'origine de la base, ainsi que l'emplacement futur de la copie de sécurité de ce fichier.
- **Informations sur le fichier de données** : cette page indique la taille et l'emplacement du fichier de données d'origine de la base, ainsi que l'emplacement futur de la copie de sécurité de ce fichier.
  Elle permet également de définir le fichier de données à convertir. Par défaut, le fichier de données courant est sélectionné. Vous pouvez créer un nouveau fichier de données vierge (option **Créer un nouveau fichier de données**) ou choisir de convertir un autre fichier de données en cliquant sur le bouton **Choisir un autre fichier de données...** Pour plus d'informations sur la conversion de plusieurs fichiers de données, reportez-vous au paragraphe "Conversion de bases utilisant plusieurs fichiers de données", page 25.
  Le bouton **Changer la destination...** vous permet de modifier l'emplacement du nouveau fichier de données après conversion.
- **Informations sur les segments** : cette page liste les segments de données associés à la base. Pour plus d'informations sur ce point, reportez-vous au paragraphe "Conversion de bases en multi-segments", page 26.
- **Informations sur les disques** : cette page indique l'espace disponible sur le disque de la base.
- **Options** : cette page contient une option de conversion.
  - **Ne pas exécuter de code à l'ouverture** : lorsque cette option est cochée, le code éventuellement exécuté au démarrage de la base est désactivé au premier lancement de la base convertie. Cette option s'applique au code appelé via la méthode base Sur ouverture. Elle permet de prévenir les erreurs d'initialisation qui peuvent se produire à la suite des conversions.

Fichiers des bases converties

Après conversion en version 11, les bases de données 4D contiennent plusieurs fichiers supplémentaires :

- ***NomBase*.4DIndy** : contient l'index de la structure.
- ***NomBase*.4DIndx** (facultatif) : contient les index des données.

Note Dans 4D v11, les index sont désormais stockés par défaut à l'extérieur de la base. Pour plus d'informations sur ce point, reportez-vous au paragraphe "Gestion des index", page 39.

- **DataConversion_Log.log** : ce fichier stocke les éventuelles anomalies rencontrées lors de la conversion de la base.
- Dossier **Replaced files** : ce dossier contient la copie des fichiers originaux de la base (structure, données et historique) avant conversion.

Conversion des anciennes bases : cas particuliers

Conversion de bases utilisant plusieurs fichiers de données

Si la base que vous souhaitez convertir utilise plusieurs fichiers de données différents, la conversion de la base et de chaque fichier de données devra être effectuée séparément. La conversion des fichiers de données additionnels est effectuée via la commande **Ouvrir>Fichier de données** du menu **Fichier**.

1 **Convertissez la base en version 11 avec le fichier de données courant.**

Cette procédure est décrite dans le paragraphe "Conversion des anciennes bases : cas général", page 21.

2 **Dans la structure convertie, choisissez la commande Ouvrir>Fichier de données... dans le menu Fichier.**

Une boîte de dialogue standard d'ouverture de documents apparaît, vous permettant de désigner le fichier de données que vous souhaitez convertir.

3 **Sélectionnez le fichier de données à convertir et cliquez sur le bouton Ouvrir.**

La fenêtre de l'assistant de conversion apparaît alors. Les informations relatives au fichier de données sont mises à jour en fonction du fichier sélectionné. Le fichier de données à convertir sera préalablement dupliqué dans un dossier **Replaced files (Conversion)**, selon les principes de conversion standard (cf. paragraphe "Conversion des anciennes bases : cas général", page 21).

4 **Validez la fenêtre de l'assistant afin de lancer la conversion.**

A l'issue de la conversion, le fichier de données converti devient le fichier de données courant de la base.

Conversion de bases avec composants

Il n'est pas possible de convertir une structure contenant un composant d'ancienne génération (antérieur à la version 11). Il est impératif de désinstaller tout composant à l'aide de la version de 4D Insider correspondante avant de procéder à la conversion d'une structure en version 11.

Vous devrez ensuite obtenir une version mise à jour du composant afin de pouvoir l'utiliser dans l'environnement v11. Pour plus d'informations sur les composants nouvelle génération, reportez-vous au paragraphe "Nouvelle architecture des composants", page 50.

Conversion de bases en multi-segments

Dans 4D v11, la taille du fichier de données est désormais virtuellement illimitée (à l'exception des limites imposées par le système d'exploitation). Il n'est plus possible de créer ni d'utiliser des segments de données.

Lors de la conversion d'une base comportant des segments, l'assistant de conversion regroupe le contenu de tous les segments au sein du nouveau fichier de données. Vous devez veiller à disposer de suffisamment de place sur votre disque pour le nouveau fichier de données.

Vous pouvez visualiser les segments d'une base dans la page "Informations sur les segments" de la boîte de dialogue de conversion (cf. paragraphe "Visualiser ou modifier les paramètres de conversion", page 23).

Si un segment est manquant au moment de la conversion, l'assistant de conversion vous permet de le désigner manuellement. Si le segment ne peut être localisé, la conversion ne peut pas être effectuée.

Conversion de bases avec champs multilingues

Lors de la conversion d'une base en version 11, les données textuelles sont converties en Unicode. Pour que la conversion soit correctement effectuée, 4D doit connaître le jeu de caractères source. Par défaut, le jeu de caractères correspondant à la langue du système courant est utilisé.
Dans les versions précédentes de 4D, il était possible d'associer un jeu de caractères spécifique à des objets de formulaires via la propriété **Configuration du clavier**. Dans ce cas, 4D v11 ne peut pas déduire de façon certaine le jeu de caractères source car la propriété était stockée avec le formulaire et la saisie pouvait être effectuée via des variables intermédiaires.

Il revient au développeur de déclarer explicitement **avant la conversion** les jeux de caractères spécifiques utilisés, via un fichier texte placé au même niveau que le fichier de données à convertir. Ce fichier devra avoir les caractéristiques suivantes :

- *Nom* : **multilang.txt**
- *Encodage* : **ANSI** ou **Mac roman** (ne pas utiliser Unicode)
- *Format* : **numéro_de_table séparateur numéro_de_champ séparateur numéro_de_sous_champ (facultatif) séparateur code_dialecte CRLF**
  - Le séparateur est ";"
  - Chaque ligne doit être terminée par un retour chariot (CR ou CRLF), les lignes vides et les espaces sont autorisés.
- *Exemple* :
  Soit une base avec 4 tables (TABLE_1 à TABLE_4).
  - Dans TABLE_3 il y a 4 champs Alpha (champ_1 à champ_4)
  - Dans TABLE_4 il y a 6 champs Entier long et une sous-table SOUS-CHAMP avec 2 champs Alpha
  - La propriété "Configuration du clavier" (dialecte) avait été modifiée de manière à pouvoir saisir du grec dans [TABLE_3]champ_3 et [TABLE_4][SOUSCHAMP]champ_1, et du russe dans [TABLE_3]champ_4 et [TABLE_4][SOUSCHAMP]champ_2
  - Le fichier *multilang.txt* devra donc contenir :
  3;3;1049
  3;4;1032

  4;7;1;1049
  4;7;2;1032

  Les codes de dialecte sont décrits dans le fichier *keyboardmapping.xml,* présent dans le sous-dossier *4D Extensions*.

Compatibilité des plug-ins

Les plug-ins version 2004 (plug-ins 4D ou plug-ins commercialisés par des sociétés tierces) sont compatibles avec 4D v11 et peuvent être utilisés dans cet environnement.

A noter que les plug-ins devront être mis à jour pour pouvoir tirer parti du nouveau moteur de 4D v11 (cf. paragraphe "Extension des capacités", page 33).

Sous Mac OS, les plug-ins devront être d'architecture Universal Binary pour que vous puissiez exécuter 4D dans ce mode et tirer parti de ses performances. Dans le cas contraire, vous pourrez cependant utiliser les plug-ins dans 4D en mode *Rosetta* (cf. paragraphe "Architecture Universal Binary (Mac OS)", page 38).

Compatibilité des applications 4D

Mécanismes non conservés

Dans 4D version 11, plusieurs mécanismes obsolètes ou inutiles ne sont plus pris en charge. Ils seront ignorés ou remplacés durant la conversion. Ces options et mécanismes sont les suivants :

- **composants "ancien mode"** : ces composants doivent être désinstallés à l'aide de 4D Insider avant toute conversion de base en version 11 (cf. paragraphe "Conversion de bases avec composants", page 26).
- **segments de données** : les segments de données seront regroupés en un seul fichier au moment de la conversion (cf. paragraphe "Conversion de bases en multi-segments", page 26).
- **champs Racine** (**sous-tables**) : les champs de type Racine (aussi appelés sous-tables ou sous-structures) ne sont plus pris en charge. Ces champs sont convertis en tables standard reliées à la table d'origine par un lien automatique et un champ est ajouté dans les nouvelles tables. Les anciens mécanismes de gestion des sous-enregistrements continuent toutefois de fonctionner tant qu'aucune modification n'est effectuée sur les tables, champs ou liens issus des sous-tables. Pour plus d'informations sur ce point, reportez-vous au paragraphe "Conversion des sous-tables", page 44.

Note Les sous-enregistrements ne sont plus pris en charge par les commandes DUPLIQUER ENREGISTREMENT, ENVOYER ENREGISTREMENT et RECEVOIR ENREGISTREMENT.

- **affectation de groupes d'utilisateurs aux opérations dans les tables** : il n'est plus possible d'affecter des groupes aux opérations sur les données des tables (chargement, enregistrement, ajout et suppression) dans l'éditeur de Structure (cf. paragraphe "Barre d'outils, barre d'information et nouvel Inspecteur", page 110). De même, il n'est plus possible de définir de groupe propriétaire de la table.
Ces contrôles sont désormais inopérants dans 4D v11. Dans le cadre de bases de données converties, ils sont ignorés.

Dans 4D v11, le contrôle des accès aux différentes parties de l'application peut s'effectuer au niveau de l'interface (formulaires, menus, méthodes), via le langage ou encore via des options spécifiques (voir par exemple le paragraphe "Contrôle des accès à la base de données 4D", page 213 relatif aux accès aux données via le SQL).

- **accès mode Utilisation** : ce pop up menu de la page Accès des Préférences a été supprimé, car le mode Utilisation n'existe plus en tant que tel (cf. paragraphe "Interface du mode Développement", page 75). Seul l'accès au mode Développement peut être restreint.
- **report de la sauvegarde en cas d'opération critique** : les paramètres de report de la sauvegarde automatique en cas d'opération critique ("Si transactions actives ou opérations d'index :"), auparavant disponibles dans la page Sauvegarde/Sauvegarde des Préférences, ont été supprimés. Ces opérations sont désormais gérées différemment.
- **accès direct aux ressources de 4D et aux ressources système** : l'accès via les commandes du thème "Ressources" aux ressources internes de 4D et du système est désormais impossible. Un mécanisme de substitution a été mis en place pour limiter l'impact de cette modification sur le fonctionnement des bases de données tirant parti de ces ressources. Pour plus d'informations, reportez-vous au paragraphe "Gestion des ressources", page 47.
- **macros 4D 2003/2004** : les macros utilisées avec les versions 2003 et 2004 de 4D ne sont pas compatibles avec 4D v11. Un mécanisme de conversion automatique est mis en place mais le respect de la norme xml peut nécessiter des adaptations ponctuelles. Pour plus d'informations, reportez-vous au paragraphe "Nouvelle architecture des macros", page 139.
- **personnalisation de l'icône de la fenêtre d'identification via une ressource** : l'icône de la boîte de dialogue d'identification de l'utilisateur (sélection du nom et saisie du mot de passe) pouvait être personnalisée à l'aide de la copie d'une ressource. Ce mécanisme ne fonctionne plus désormais et est remplacé par la simple copie d'un fichier (cf. paragraphe "Personnalisation de l'icône de la fenêtre d'identification", page 74).

- **"Choix du mode" pour les images sur fond** : il n'est plus possible de choisir le mode d'interaction entre les couleurs d'une image (format Image sur fond) et du fond via la fenêtre "Choix du mode". Ce mécanisme, mis en place à l'origine pour les images monochromes, n'est plus adapté aux images gérées en mode natif (cf. paragraphe "Optimisation des variables et champs image", page 160).
- **Actions standard Structure, Utilisation et Menus créés** : dans les bases de données converties, ces actions standard ne sont plus sélectionnables. Elles ne sont plus disponibles dans les nouvelles bases de données (cf. paragraphe "Actions standard dans les bases converties", page 183).

Options de compatibilité supprimées

Les options de compatibilité suivantes ont été supprimées des Préférences de 4D v11 (page **Application/Compatibilité**) :

- **Utiliser les formules-fichiers de la v5.x.x**
- **Compatibilité 6.8 pour le rendu du texte**

  Ces options émulaient des anciens mécanismes désormais obsolètes.
- **Ancien mécanisme du menu Edition** (option **v 6.8**)
  Cette option de l'éditeur de menus est désormais supprimée. Dans le cas des bases de données converties utilisant cette option, il sera nécessaire d'ajouter un menu **Edition** standard (cf. paragraphe "Menu Edition (compatibilité)", page 184).
- **Utiliser la méthode Debut de la v5.x.x**
  Cette ancienne option n'est désormais plus prise en charge dans 4D v11, les méthodes projet appelées *Debut* ou *Startup* ne sont plus exécutées automatiquement. Le code de démarrage doit être placé dans les méthodes base Sur démarrage et/ou Sur démarrage serveur.

Commandes supprimées

Les commandes suivantes n'existent plus dans 4D v11 car elles correspondaient à des technologies qui ne sont plus prises en charge.

- CHERCHER SUR CLE
- STOCKER ANCIEN
- TRIER SUR INDEX

Modifications de syntaxe

Ce paragraphe indique les évolutions de 4D v11 pouvant conduire à des erreurs de syntaxe dans les applications converties.

- Pour des raisons structurelles, le paramètre *table* devient **obligatoire** pour les commandes CHERCHER PAR FORMULE, CHERCHER PAR FORMULE DANS SELECTION, et APPLIQUER A SELECTION. Si ces commandes étaient utilisées sans ce paramètre, une erreur de syntaxe est générée par 4D v11.
- De manière générale, l'interpréteur du langage de 4D v11 est plus strict que celui des versions précédentes de 4D. En particulier, les retypages implicites, l'emploi de tableaux au lieu de variables et inversement pourront générer des erreurs de syntaxe en version 11, alors que ces pratiques étaient tolérées dans les versions précédentes du programme.

Note Le fonctionnement interne de nombreuses commandes a été modifié dans 4D v11, ce qui peut parfois nécessiter des ajustements dans les applications converties. Ces modifications sont décrites tout au long du chapitre "Langage", page 255 ; à noter en particulier le paragraphe "Modifications diverses", page 423.

- La variable système *Document* contient désormais toujours le chemin d'accès complet des documents (cf. paragraphe "Variable système Document", page 423).

Formats des documents

Le format interne des documents manipulés par les commandes STOCKER ENSEMBLE/CHARGER ENSEMBLE et ENVOYER ENREGISTREMENT/RECEVOIR ENREGISTREMENT a été modifié. La compatibilité ascendante est assurée (les commandes CHARGER ENSEMBLE et RECEVOIR ENREGISTREMENT de 4D v11 pourront relire les documents générés en version précédente).
En revanche, les documents générés avec 4D version 11 ne pourront pas être ouverts par des versions précédentes.

3 Architecture des bases de données 4D v11 SQL

Le moteur de bases de données de 4D v11 a été réécrit. Plus performant, plus puissant, ce moteur nouvelle génération bénéficie des technologies les plus avancées en matière de gestion des données et repousse les limites des précédentes versions.

L'architecture des bases de données 4D v11 a également été modifiée, aussi bien au niveau des fichiers que des composants.

Les dossiers contenant les fichiers des bases de données 4D v11 utilisent également une nouvelle architecture.

Extension des capacités

De nombreuses limitations des versions précédentes de 4D sont supprimées ou repoussées dans 4D v11.

- Le tableau suivant compare les nouvelles capacités du moteur de bases de données de 4D v11 avec celles des versions précédentes de 4D :

Capacités	**4D 200x**	**4D v11**
Taille fichier de données	127 segments de 2 Go chacun	**illimitée (pas de segmentation)**
Nombre de tables	255	**32767**
Nombre de champs par table	511	**32767**
Nombre d'enregistrements par table	16 millions	**1 milliard**
Nombre de clés d'index par table	16 millions	**128 milliards**

Capacités	4D 200x	4D v11
Longueur des champs alpha	80	**255**
Longueur des champs texte	32000 caractères	**2 Go de texte**[1]
Niveaux de transactions	1	**illimité**[2]

1. Pour plus de clarté, la constante existante MAXLARGTEXTE (valeur 32000) a été renommée MAXLONGTEXTEAVANTv11. Son usage est désormais inutile.
2. Pour des raisons de compatibilité, les bases de données converties restent limitées par défaut à 1 niveau de transaction (cf. ci-dessous).

Nouveaux types de champs

Deux nouveaux types de champs sont disponibles dans l'éditeur de structure de 4D v11 : Entier 64 bits et Float.

- **Entier 64 bits** : Entier sur 8 octets permettant de manipuler des valeurs comprises dans l'intervalle -2^63..(2^63)-1
- **Float** : Nombre à virgule flottante. Les nombres à virgule flottante permettent de stocker des valeurs réelles sans perte de précision.

Attention, dans la version actuelle de 4D v11, ces types de champs sont exploités uniquement par le moteur SQL de 4D (cf. paragraphe "Utiliser le moteur SQL de 4D", page 205). Si ces champs sont utilisés dans le langage de 4D, leurs valeurs sont converties en interne en nombres réels.

Suppression des tables et des champs

Dans 4D v11, il est possible de supprimer physiquement les tables et les champs.

Cette opération peut être effectuée via le moteur SQL de 4D ou via l'éditeur de structure.

Pour plus d'informations sur l'utilisation des instructions SQL dans 4D, reportez-vous au chapitre "Utiliser le moteur SQL de 4D", page 205.

Pour supprimer une table ou un champ depuis l'éditeur de structure, il suffit de sélectionner la table ou le champ à supprimer puis de choisir la commande **Effacer** dans le menu **Edition** ou la commande **Supprimer** dans le menu contextuel de la table.

Une boîte de dialogue d'alerte vous demande de confirmer l'opération :

Si vous cliquez sur le bouton **OK**, 4D effectue les opérations suivantes :

- La table ou le champ est définitivement supprimé(e) de la structure. Toutes les données associées à la table ou au champ sont définitivement supprimées du fichier de données.
- La méthode trigger éventuellement associée à la table est supprimée.
- Les formulaires table associés à la table sont transformés en formulaires projet (paragraphe "Formulaires projet", page 143) et sont placés dans la Corbeille de l'Explorateur.

Placer une table dans la corbeille

Vous pouvez également supprimer une table de façon non définitive en la plaçant dans la Corbeille (accessible via l'Explorateur). La table disparaît alors des éditeurs de 4D et son contenu devient inaccessible mais elle pourra être récupérée tant que la Corbeille n'aura pas été vidée. Cette fonction existait dans les versions précédentes de 4D.

Pour placer une table dans la Corbeille, choisissez la commande **Placer dans la corbeille** dans le menu contextuel ou sélectionnez la table et appuyez sur la touche **Suppr**. ou **Ret. Arr.** La boîte de dialogue suivante apparaît dans ce cas :

Numéros des tables et des champs supprimés

Lors de la suppression d'une table ou d'un champ, les autres tables et champs de la base ne sont pas renumérotés, afin de ne pas mettre en péril la stabilité de la base. Il est donc possible d'obtenir une base de données comportant trois tables numérotées 2, 4 et 5. Il est également possible d'obtenir une table avec quatre champs, numérotés 1,4, 6 et 8.

Ce principe rend caduc le décompte des tables et des champs utilisant les anciennes commandes Nombre de tables et Nombre de champs car ces commandes ne tiennent pas compte des éventuels "trous" dans la

numérotation. Pour cela, ces commandes ont été renommées et de nouvelles commandes ont été ajoutées dans 4D v11 (cf. paragraphe “Compter les tables et les champs”, page 399).

A noter que les numéros des tables et des champs supprimés sont réutilisés lors de la création ultérieure de nouvelles tables ou de nouveaux champs (ce principe est semblable à celui mis en oeuvre pour les numéros des enregistrements).

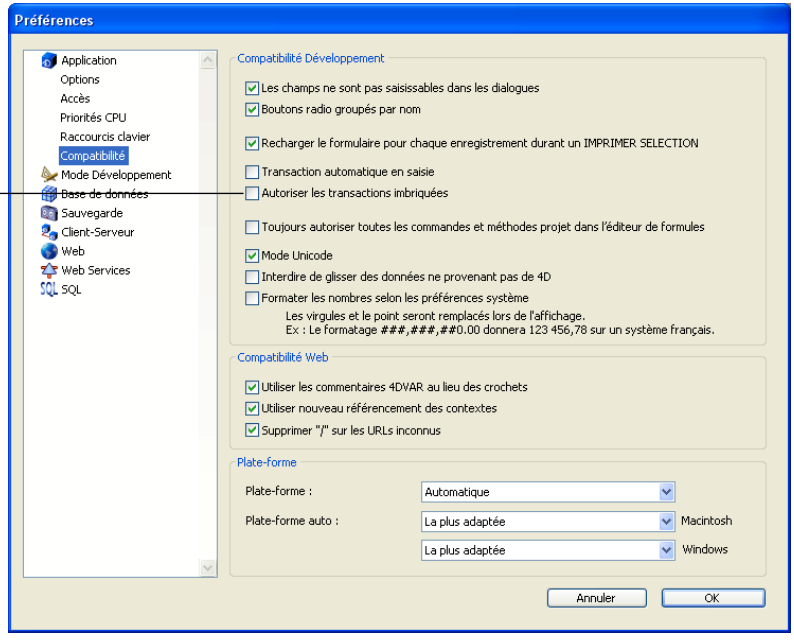
Compatibilité des transactions

4D v11 accepte désormais les transactions imbriquées sur un nombre de niveaux illimité.

Ce nouveau fonctionnement pouvant générer des dysfonctionnements dans les bases développées avec des versions précédentes de 4D, il est désactivé par défaut dans les bases converties (les transactions restent limitées à un seul niveau).

Si vous souhaitez bénéficier des transactions sur plusieurs niveaux dans une base convertie, vous devez l’indiquer explicitement en cochant l’option **Autoriser les transactions imbriquées** dans la page “Application/Compatibilité” des Préférences de l’application :

Option d’activation des transactions multi-niveaux dans les bases converties



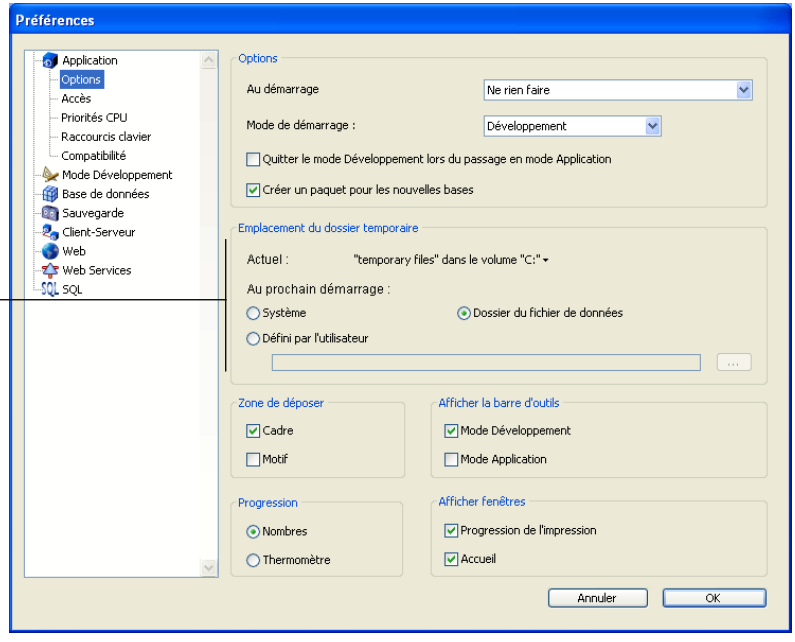
Cette option n'apparaît que pour les bases de données converties. Par défaut, elle n'est pas cochée. Elle est spécifique à chaque base de données.

Note L'option **Autoriser les transactions imbriquées** n'a pas d'effet sur les transactions effectuées dans le moteur SQL de 4D v11 (cf. chapitre "Utiliser le moteur SQL de 4D", page 205). Les transactions SQL sont toujours multi-niveaux.

Dossier temporaire

4D v11 propose de nouvelles options pour le stockage du "dossier temporaire". Ce dossier est utilisé par l'application, en cas de nécessité, pour écrire temporairement sur disque des données se trouvant en mémoire.

La définition de l'emplacement du dossier s'effectue dans la page Applications/Options des Préférences :



Zoine de définition du dossier temporaire

L'emplacement courant du dossier est indiqué sous forme de liste déroulante en haut de la zone. Cliquez sur la liste pour visualiser ou copier le chemin d'accès complet.

Trois options d'emplacement sont proposées :

- **Système** : lorsque cette option est sélectionnée, les fichiers temporaires 4D sont créés dans un dossier situé à l'emplacement spécifié par Windows et Mac OS. Vous pouvez lire l'emplacement courant défini par votre système à l'aide de la commande Dossier temporaire.
Les fichiers sont placés dans un sous-dossier dont le nom est construit à partir du nom de la base et d'un identifiant unique.
- **Dossier du fichier de données** (option par défaut) : lorsque cette option est sélectionnée, les fichiers temporaires 4D sont créés dans un dossier nommé "temporary files" situé au même niveau que le fichier de données de la base.
- **Défini par l'utilisateur** : permet de définir un emplacement personnalisé.

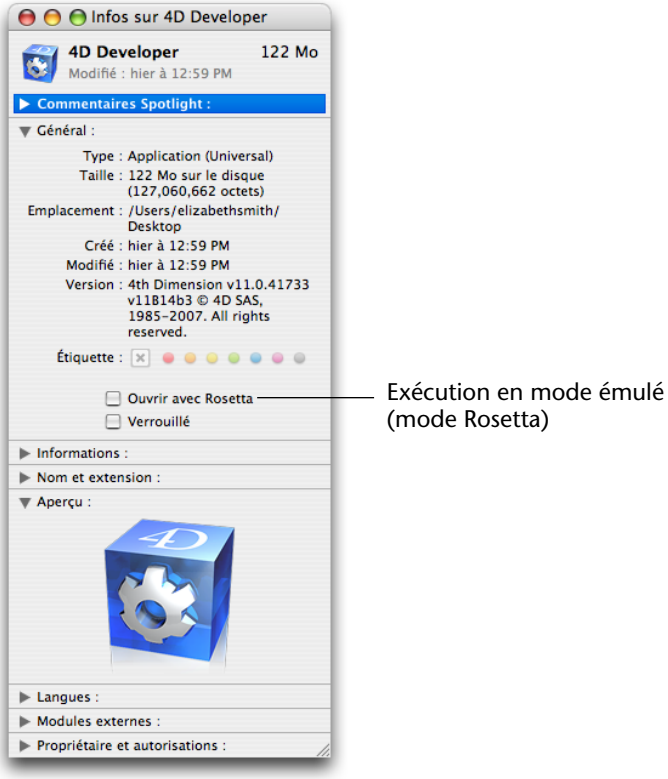
  En cas de modification de cette option, sa prise en compte nécessitera le redémarrage de la base.
  4D vérifie que le dossier sélectionné est accessible en écriture. Si ce n'est pas le cas, l'application essaiera les autres options jusqu'à ce qu'un dossier valide soit trouvé.
  Cette option est stockée dans les propriétés additionnelles ("extra properties") de la structure, accessibles lors de l'exportation xml de la définition de structure (cf. paragraphe "Exporter et importer des définitions de structures", page 131).

Architecture Universal Binary (Mac OS)

La version Mac OS de 4D v11 est d'architecture *Universal Binary*, ce qui signifie que l'application 4D est exécutée en mode natif sur les machines Apple récentes, à base de processeurs Intel ("Mac Intel"). Il en résulte un gain de performances important.

Par défaut, les applications 4D v11 sont exécutées en mode natif dans cet environnement. Toutefois, vous pouvez forcer l'exécution en mode émulé (mode *Rosetta*), notamment pour pouvoir utiliser des plug-ins d'architecture non Universal Binary. Pour cela, sélectionnez l'application 4D au niveau du Bureau et choisissez la commande **Lire**

les informations dans le menu **Fichier**. Une option de la fenêtre “Infos” vous permet de forcer l’exécution de l’application en mode émulé :



Gestion des index

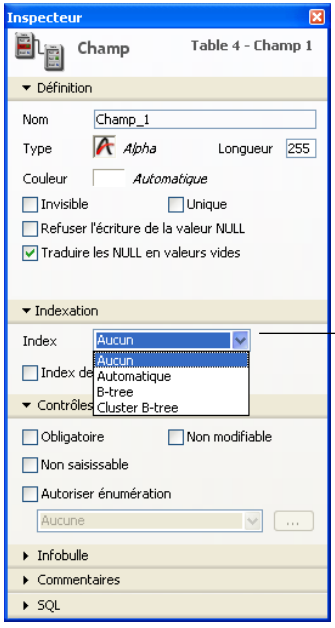
La gestion des index a été modifiée dans 4D v11. D’une part, il est désormais possible de choisir le mode de création des index et d’autre part, de nouveaux types d’index sont disponibles. Une nouvelle option de la boîte de dialogue d’import permet également de différer la reconstruction de l’index.

A noter que de nouvelles commandes de langage permettent de créer et de modifier les index : reportez-vous au paragraphe “Définition structure”, page 399.

Note Toutes les opérations possibles en mode indexé peuvent être réalisées en mode non indexé. Par exemple, il n’est plus obligatoire d’indexer chaque champ utilisé par un lien.

Sélection de l'architecture des index standard

4D v11 permet de choisir l'architecture des index standard au moment de leur création (index standard désigne un index générique, par opposition aux index de mots-clés ou composites, cf. paragraphes suivants). Dans l'éditeur de structure, un pop up menu permet de gérer les index standard :



Sélection de l'architecture de l'index

Vous disposez de quatre choix :

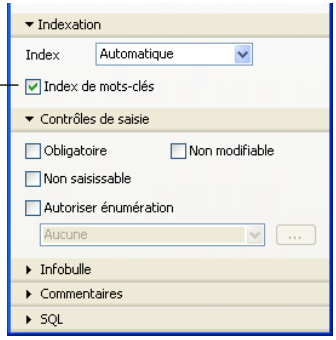
- **B-tree** : index de type B-Tree classique. Ce type d'index polyvalent est utilisé dans les versions précédentes de 4D.
- **Cluster B-tree** : index de type B-Tree utilisant des *clusters*. Cette architecture est plus efficace lorsque l'index contient peu de clés, c'est-à-dire lorsque les mêmes valeurs reviennent souvent dans les données.
- **Automatique** : 4D sélectionne l'architecture en fonction des données.
- **Aucun** : pas d'index ou suppression de l'index existant.

Note Il est également possible de créer des index via la nouvelle boîte de dialogue de création d'index (cf. paragraphe "Boîte de dialogue de propriété d'index", page 124).

Index de mots-clés

Il est désormais possible d'utiliser un nouveau type d'index avec les champs Alpha et Texte : *mots-clés*. Pour associer un index de mots-clés à un champ, il suffit de cocher l'option **Index de mots-clés** dans l'Inspecteur de l'éditeur de structure :

Sélection du nouveau type d'index



Lorsque vous cochez cette option, les textes stockés dans le champ concerné seront indexés mot par mot. Chaque mot sera indexé, même s'il ne comporte qu'un ou deux caractères. Ce type d'index permet d'accélérer de façon spectaculaire les recherches par mots-clés parmi les champs texte (pour plus d'informations sur les recherches par mots-clés, reportez-vous au paragraphe "Recherches par mots-clés", page 275). Cochez cette option si vous devez effectuer des recherches par mots-clés dans le champ.

Notes

• Il reste parfaitement possible de créer un index classique avec les champs Texte (lorsqu'ils sont stockés avec les enregistrements, cf. paragraphes suivants) et les champs Alpha. En fait, les deux types d'index peuvent être sélectionnés pour un même champ. 4D utilisera l'index le plus approprié en fonction du contexte.

• Il est possible de créer des index de mots-clés via la nouvelle boîte de dialogue de création d'index (cf. paragraphe "Boîte de dialogue de propriété d'index", page 124).

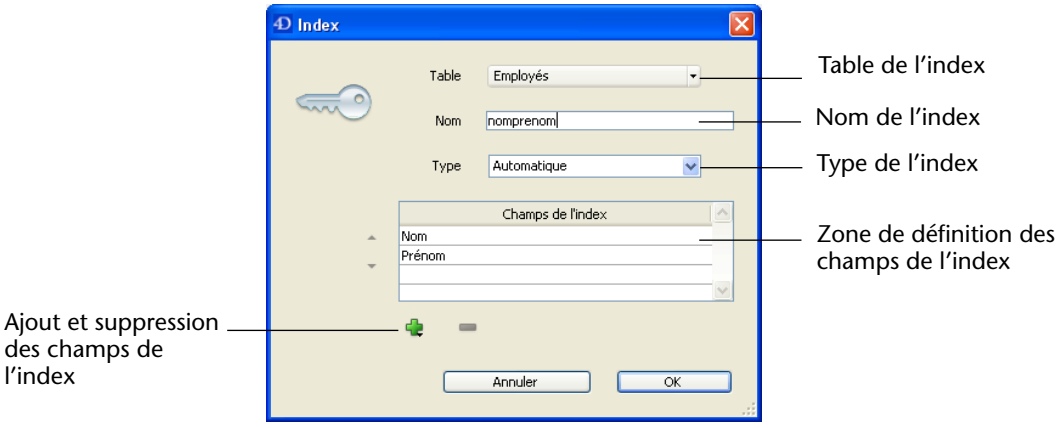
Index composites

Vous pouvez créer des index *composites*, c'est-à-dire des index stockant la valeur conjointe de deux ou plusieurs champs pour chaque entrée. L'exemple classique est l'index composé basé sur les champs Prénom+Nom. La recherche de "Pierre Martin" sera alors optimisée comparativement à une recherche standard (recherche des "Martin" puis recherche des "Pierre").

4D tire automatiquement parti des index composites lors des recherches ou des tris. Par exemple, si un index composite "Code

postal+Ville" existe, il sera utilisé en cas de recherche du type "*nom=dupond* et *ville=paris* et *code postal=750@*".

La création des index composites s'effectue uniquement via la nouvelle boîte de dialogue de création d'index, accessible depuis l'éditeur de structure de 4D :



Pour une description détaillée de cette boîte de dialogue, reportez-vous au paragraphe "Boîte de dialogue de propriété d'index", page 124.

Note La boîte de dialogue de création d'index permet également de créer des index classiques ou de mots-clés.

Commandes "par formule"

Les commandes CHERCHER PAR FORMULE, CHERCHER PAR FORMULE DANS SELECTION et TRIER PAR FORMULE tirent désormais parti des index, lorsqu'ils sont disponibles.

En environnement client/serveur, ces commandes sont maintenant exécutées sur le poste serveur. Par conséquent, les variables locales ne peuvent plus être utilisées dans ce contexte.

Par souci de compatibilité, dans les anciennes bases de données converties, le fonctionnement des versions précédentes de 4D est conservé (opérations séquentielles). Une case à cocher dans la boîte de dialogue des Préférences permet d'activer le nouveau fonctionnement.

Stockage des index sous forme de fichiers

Les index créés dans une base 4D v11 sont stockés sous la forme de fichiers séparés. Ces fichiers sont automatiquement placés à côté du fichier de structure. Ils ne doivent pas être déplacés ni renommés, sinon 4D doit les recréer.

Deux fichiers d'index sont créés :

- ***NomBase*.4DIndx** : contient les index des données.
- ***NomBase*.4DIndy** : contient l'index de la structure (utilisé notamment lors de recherches dans le développement).

L'un des intérêts majeurs de ce principe est la possibilité, en cas de corruption de l'index, de supprimer physiquement le fichier avant de lancer 4D afin que le fichier soit automatiquement recréé.

Reconstruire l'index après l'import

Une nouvelle option est disponible dans la boîte de dialogue d'import de données : **Reconstruire les index après l'import**. Lorsque cette option est cochée (par défaut), les index des champs dans lesquels des données ont été importées sont reconstruits à l'issue de l'import. Ce mécanisme permet d'accélérer les imports de gros volumes de données.

Si l'import concerne une quantité de données moins importante que celle déjà contenue dans le champ, il peut être intéressant de désélectionner cette option avant l'import. Dans ce cas, l'index est mis à jour au fur et à mesure de l'import et n'est pas reconstruit en totalité.

Stockage des champs BLOB, Image et Texte

Dans 4D v11, les données contenues dans les champs pouvant stocker de gros volumes (type BLOB, Image et Texte[1]) sont désormais stockées en-dehors des enregistrements. Ce principe permet d'accélérer l'exécution des bases, notamment lors des recherches : les éventuelles données volumineuses contenues dans ces champs ne sont plus systématiquement chargées en mémoire lorsque 4D accède aux enregistrements hôtes. Ces données ne sont effectivement chargées que lorsque c'est nécessaire, par exemple une fois que l'enregistrement recherché a été trouvé.
Ce nouveau fonctionnement est automatique et ne nécessite aucune modification du code existant — à l'exception du fait qu'il rend inutile le code éventuellement mis en place par les développeurs pour stocker les données dans une table séparée.

En effet, pour des raisons d'optimisation, la plupart des développeurs 4D utilisent, pour la gestion des champs BLOB, Image et Texte, une

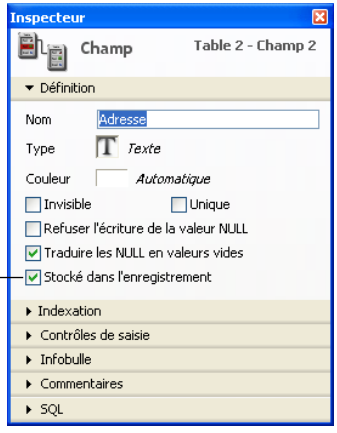
1. Une option permet de "forcer" le stockage des champs Texte dans les enregistrements, cf. paragraphe suivant.

table dédiée, reliée à la table principale par un lien de type un-vers-un. Cette astuce fonctionne parfaitement en version 11 mais n'est désormais plus nécessaire.

Option pour le stockage des champs Texte

Une option des propriétés des champs Texte permet de "forcer" le stockage des données texte dans les enregistrements (équivaut au fonctionnement des versions précédentes de 4D).

Cette possibilité est nécessaire notamment si vous souhaitez utiliser un index "classique" avec un champ Texte (pour plus d'informations sur les index des champs Texte, reportez-vous au paragraphe "Index de mots-clés", page 41). En effet, les index classiques sont incompatibles avec le stockage hors enregistrement des champs Texte. Dans ce cas, vous devez cocher l'option **Stocké dans l'enregistrement**, située dans la zone "Définition" de la palette Inspecteur pour le champ sélectionné :



Option de stockage dans l'enregistrement

Par compatibilité, cette option est cochée par défaut pour les champs Texte des bases de données converties.

Elle est désélectionnée par défaut pour les champs Texte créés en version 11.

Conversion des sous-tables

Les sous-tables ne sont plus prises en charge dans 4D à compter de la version 11. Dans les nouvelles bases, la création de sous-tables est impossible.

Par compatibilité, dans les bases converties, les sous-tables existantes continueront de fonctionner normalement. Un traitement spécifique leur est appliqué au moment de la conversion afin de les transformer en tables standard avec liens automatiques spéciaux. Ce paragraphe décrit le principe de cette conversion et le fonctionnement des anciennes sous-tables dans 4D v11 SQL.

Note Seules les sous-tables de premier niveau sont prises en charge par l'assistant de conversion. Si la base d'origine contient des sous-tables sur des niveaux supplémentaires, elles sont ignorées et l'erreur 1012 est consignée dans le fichier d'historique *DataConversion_Log.log*.

Mécanisme de conversion

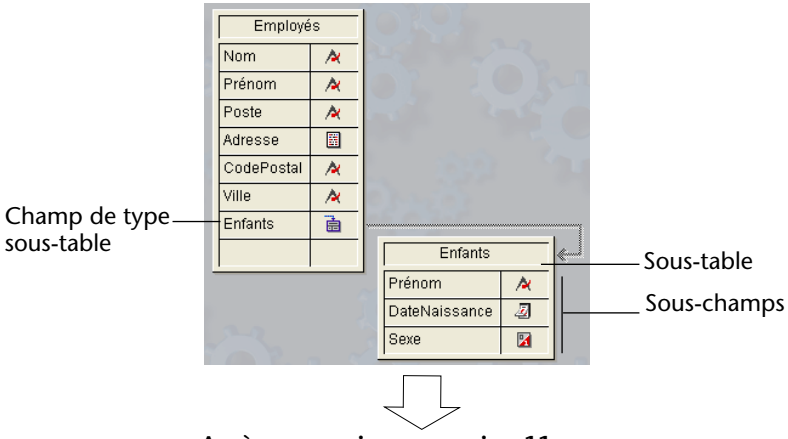
Lors de la conversion d'une base en version 11, les sous-tables existantes sont automatiquement transformées en tables standard reliées aux tables d'origine par un lien automatique. La sous-table devient une table "N" et la table d'origine la table "1".

Dans la table d'origine, l'ex-champ de type sous-table est transformé en champ spécial de type "Lien sous-table".
La sous-table, transformée en table standard, est renommée "*Table_Champ*", où *Table* représente le nom de la table d'origine et *Champ* le nom de l'ancien champ sous-table dans la table d'origine. Ce nom est tronqué s'il dépasse les 31 caractères autorisés.

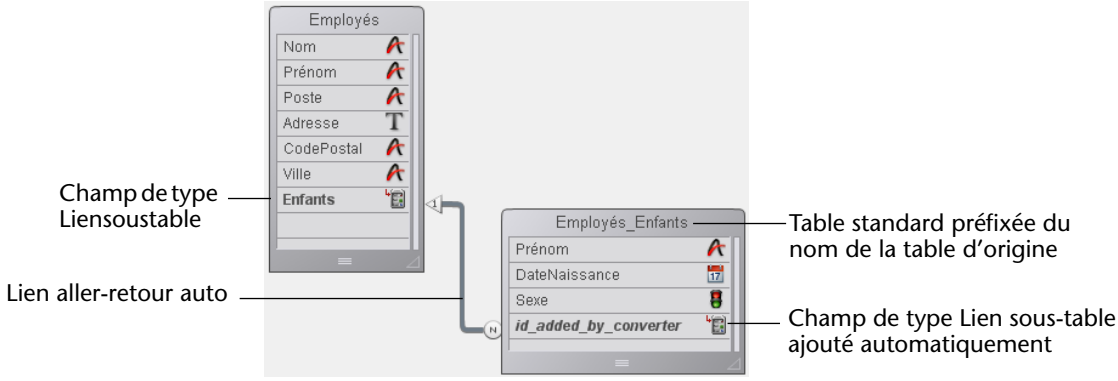
Dans chaque nouvelle table issue de la transformation d'une sous-table, un champ spécial de type "Lien sous-table" est ajouté. Ce champ

est utilisé pour le lien automatique vers la table d'origine. Il est nommé "<*Table*>_Relation" où *Table* représente le nom de la table d'origine.

Sous-table (version de 4D < v11)



Après conversion en version 11



Note Le type de champ "Lien sous-table" ne peut pas être assigné manuellement, seule la conversion d'une base avec sous-tables peut le produire.

Les liens nouvellement créés sont automatiquement nommés "Auto_*Champ_Table*" et "Auto_*Table_Champ*".

Note Pour plus d'informations sur les noms des liens, reportez-vous au paragraphe "Propriétés des liens", page 128.

Après la conversion, les fonctions spécifiques des sous-tables (automatismes, commandes du langage, etc.) sont intégralement maintenues dans l'application. A noter que les autres fonctions

relationnelles basées sur l'emploi de liens automatiques seront également utilisables avec les sous-tables converties. Dans l'optique d'évolutions futures de la base, il peut être avantageux de remplacer l'emploi des sous-tables converties par des mécanismes standard.

Transformer une sous-table convertie en table liée standard

Une fois que les sous-tables ont été converties, vous pouvez souhaitez les transformer en tables liées standard. Pour cela, il suffit de supprimer le lien automatique spécial qui la relie à la table d'origine.

La suppression de ce lien entraîne automatiquement la désactivation définitive des fonctions et automatismes spécifiques associés à la sous-table. Lorsque vous demandez la suppression de ce lien, une boîte de dialogue d'alerte apparaît. Elle rappelle les conséquences de cette action et permet de l'annuler en cas d'erreur.

Une fois le lien supprimé, les deux champs spéciaux de type "Lien sous-table" sont transformés en champs standard de type Entier long.

Gestion des ressources

La gestion des ressources a été modifiée dans 4D v11.

Conformément aux orientations définies par Apple et mises en oeuvres dans les versions de Mac OS les plus récentes, le concept de ressources au sens strict est désormais obsolète et est progressivement abandonné. De nouveaux mécanismes sont mis en place pour prendre en charge les besoins précédemment couverts par les ressources : fichiers XLIFF pour la traduction des chaînes de caractères, fichiers images .png... De fait, les fichiers de ressources disparaissent au profit de fichiers de type standard.

4D v11 accompagne cette évolution et propose de nouveaux outils pour la gestion des traductions des bases de données, tout en maintenant la compatibilité avec les systèmes existants.

Compatibilité des ressources existantes

Pour le maintien de la compatibilité et afin de permettre l'adaptation progressive des applications existantes, les anciens mécanismes des ressources continuent de fonctionner dans 4D v11, à quelques différences près :

- Lorsqu'ils sont présents, les fichiers de ressources sont toujours pris en charge par 4D et le principe de la "chaîne des fichiers de ressources"

(ouverture successive de plusieurs fichiers de ressources) reste valide. La "chaîne des fichiers de ressources" comprend notamment les fichiers .rsr ou .4dr des bases de données converties (ouverts automatiquement) et les fichiers de ressources personnalisés ouverts via les commandes du thème "Ressources".

- Toutefois, pour cause d'évolution de l'architecture interne, il n'est plus possible d'accéder directement via les commandes du thème "Ressources" ou les références dynamiques aux ressources de l'application 4D ni à celles du système.
  Certains développeurs ont pu tirer parti de ressources internes de 4D pour leurs interfaces (par exemple les ressources contenant les noms des mois ou ceux des commandes du langage). Cette pratique, auparavant déconseillée, est désormais proscrite. Dans la plupart des cas, il est possible d'utiliser d'autres moyens que les ressources internes de 4D (constantes, commandes du langage...).
  Afin de limiter l'impact de cette modification sur les bases existantes, un système de substitution a été mis en place, basé sur l'externalisation des ressources les plus utilisées. Il est cependant fortement conseillé de faire évoluer les bases de données converties et de supprimer les appels aux ressources internes de 4D.

Note Si vous souhaitez utiliser certaines ressources internes de 4D, vous pouvez les dupliquer dans un fichier de ressources utilisateur et charger explicitement ce fichier via les commandes du langage.

- Les bases de données créées avec 4D v11 ne comportent plus par défaut de fichiers .RSR (ressources de la structure) et .4DR (ressources des données).

Nouveaux principes de gestion des ressources

Dans 4D v11, la notion de "ressources" doit désormais être entendue au sens large comme "fichiers utilisés pour l'interface des applications".

La nouvelle architecture des ressources s'appuie sur un dossier, nommé **Resources**, impérativement situé à côté du fichier de structure de la base (.4db ou .4dc).

Vous devez placer dans ce dossier tous les fichiers nécessaires à la traduction ou la personnalisation de l'interface de l'application (fichiers image, texte, XLIFF...).

Il peut également contenir les éventuels fichiers de resources personnalisés d'"ancienne génération" de la base (fichiers .rsr). Attention, ces fichiers ne sont pas automatiquement inclus dans la chaîne des ressources, ils devront être ouverts via les commandes standard de manipulation des ressources de 4D.

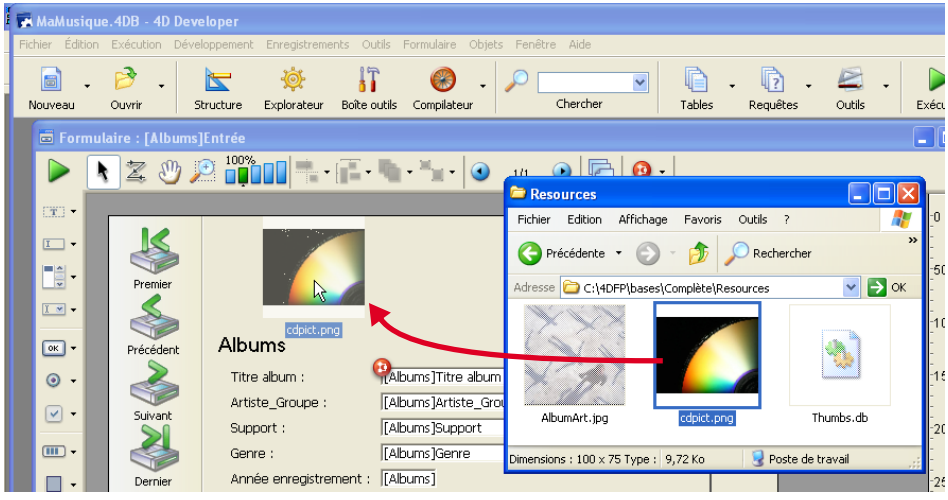
4D utilise des mécanismes automatiques pour l'exploitation du contenu de ce dossier, notamment pour la gestion des fichiers XLIFF (cf. paragraphe "Prise en charge du standard XLIFF", page 101).

Référencement automatique des images

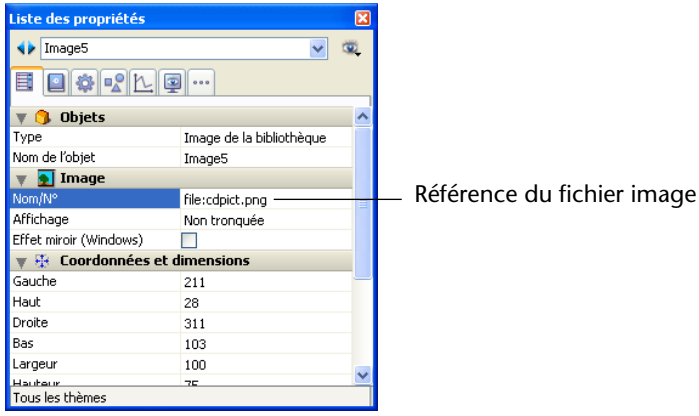
Vous pouvez utiliser le dossier **Resources** pour stocker les images statiques utilisées dans vos formulaires et les manipuler par référence (il n'est plus nécessaire de placer ces images dans la bibliothèque). L'affichage de ces images est alors optimisé.

Vous pouvez notamment faire appel à des images .png (bitmap) ou .svg (vectorielles). Pour plus d'informations sur la gestion native des images dans 4D, reportez-vous au paragraphe "Prise en charge de formats natifs", page 161.

L'insertion de ces images dans les formulaires peut s'effectuer par simple glisser-déposer depuis le dossier **Resources** :



4D insère automatiquement la référence de l'image dans le formulaire sous la forme "file:{*cheminaccès+*}nomfichier" :



Il est possible de placer les images dans des sous-dossiers du dossier Resources. En particulier, vous pouvez utiliser le mécanisme du dossier *.lproj* pour les images en différentes langues.

Nouvelle architecture des composants

L'architecture des composants 4D a été modifiée en profondeur dans 4D v11, dans le but de les rendre plus simples à développer et à installer, tout en leur conférant un niveau de sécurité maximal.

Les composants "ancienne génération" ne sont plus pris en charge dans 4D v11. Pour plus d'informations, reportez-vous au paragraphe "Conversion de bases avec composants", page 26.

Présentation

Un composant 4D est un ensemble d'objets 4D représentant une ou plusieurs fonctionnalité(s), qu'il est possible d'installer dans différentes bases.

Dans les versions précédentes de 4D, les composants étaient créés, générés et installés via l'application 4D Insider. Désormais, dans un but de simplification, la création et l'installation des composants 4D s'effectuent directement depuis 4D v11.

Dans 4D v11, les composants sont gérés comme des plug-ins. Les nouveaux principes sont les suivants :

- Un composant est un simple fichier de structure (compilé ou non compilé) d'architecture standard ou sous forme de *package* (cf. paragraphe "Dossier .4dbase", page 69).
- Pour installer un composant dans une base, il suffit de le copier dans le dossier "Components" de la base, placé à côté du fichier de structure. Il est possible d'utiliser un raccourci (Windows) ou un alias (Mac OS). Pour le désinstaller, il suffit de le supprimer du dossier.
- A la différence des versions précédentes, il n'est pas possible d'inclure de tables dans les composants 4D v11.

Ces principes sont détaillés dans les paragraphes suivants.

Nouveaux concepts (définitions)

Les nouveaux mécanismes de gestion des composants dans 4D v11 nécessitent l'introduction des concepts et de la terminologie suivants :

- **Base matrice** : base de données 4D utilisée pour développer le composant. La base matrice est une base standard, sans attribut spécifique. Une base matrice constitue un seul composant.
  La base matrice est destinée à être copiée, compilée ou non, dans le dossier Components de la base devant utiliser le composant (la *base hôte*, cf. ci-dessous).
- **Base hôte** : base dans laquelle est installée un composant (dans le dossier Components).
- **Composant** : base matrice, compilée ou non, copiée dans le dossier Components de la base hôte et dont le contenu est utilisé dans la base hôte.
- **Formulaire projet** : formulaire non relié à une table. Les formulaires projets sont une nouveauté de 4D v11, particulièrement adaptée à la génération de composants. Pour plus d'informations sur les formulaires projet, reportez-vous au paragraphe "Formulaires projet", page 143.

Note Dans 4D v11, il n'est plus obligatoire qu'une base de données contienne au moins une table.

- **Formulaire table** (aussi appelé formulaire "classique") : formulaire associé à une table. Ce type de formulaire ne peut pas être utilisé dans un composant.

Il est à noter qu'une base peut donc être à la fois "matrice" et "hôte", c'est-à-dire qu'une base matrice peut elle-même utiliser un ou plusieurs composants. En revanche, une base utilisée comme composant ne peut pas elle-même utiliser un composant : un seul niveau de composant est chargé.

Protection des composants : la compilation

Par défaut, toutes les méthodes projet d'une base matrice installée comme composant sont virtuellement visibles depuis la base hôte. En particulier :

- Les méthodes projet partagées sont accessibles dans la page "Méthodes" de l'Explorateur et peuvent être appelées dans les méthodes de la base hôte (cf. paragraphe "Partage des méthodes projet", page 61). Leur contenu peut être sélectionné et copié dans la zone de prévisualisation de l'Explorateur. Elles peuvent également être visualisées dans le débogueur.
  Il n'est toutefois pas possible de les ouvrir dans l'éditeur de méthodes ni de les modifier.
- Les autres méthodes projet de la base matrice n'apparaissent pas dans l'Explorateur mais peuvent également être visualisées dans le débogueur de la base hôte.

Pour protéger efficacement les méthodes projet d'un composant, il suffit simplement de **compiler** la base matrice et de la fournir sous forme de fichier .4dc (base compilée ne contenant pas le code interprété). Lorsqu'une base matrice compilée est installée comme composant :

- Les méthodes projet partagées sont accessibles dans la page "Méthodes" de l'Explorateur et peuvent être appelées dans les méthodes de la base hôte. En revanche, leur contenu n'apparaît pas dans la zone de prévisualisation ni dans le débogueur.
- Les autres méthodes projet de la base matrice n'apparaissent jamais.

Installation d'un composant

Pour installer un composant dans une base de données 4D v11, il suffit de copier le fichier de structure de la base matrice dans le dossier **Components** de la base hôte. Vous pouvez utiliser des alias (Mac OS) ou des raccourcis (Windows).

Ce principe est géré de manière transparente en client-serveur.

Dossier Components

Le dossier Components doit être placé à côté du fichier de structure de la base hôte. Dans le cas de bases de données hôtes compilées et fusionnées à 4D Volume Desktop, le dossier Components doit se trouver à côté du fichier exécutable (sous Windows) et dans le dossier Contents du package (sous Mac OS).

4D recherchera dans le dossier Components les bases matrices du type **.4db** (base matrice interprétée), **.4dc** (base matrice compilée) ou **.4dbase** (base matrice de type paquet, cf. paragraphe “Dossier .4dbase”, page 69). Les autres éléments, notamment les fichiers de données ou les fichiers de structure utilisateur (.4DA), sont ignorés.
Vous pouvez utiliser des alias ou des raccourcis vers ces bases matrices. Cette possibilité est particulièrement utile en phase de développement du composant car toute modification effectuée dans la base matrice est immédiatement reportée dans toutes les bases hôtes.

Vous pouvez placer dans le dossier Components un raccourci (Windows) ou un alias (Mac OS) vers une base qui est elle-même utilisée comme base hôte. Par le jeu des alias, une base hôte peut donc devenir un composant et inversement. Dans ce cas, rappelez-vous qu’un seul niveau de composants est chargé. Les composants d’une base hôte utilisée comme composant ne seront pas chargés.

Le dossier Components peut contenir tout fichier ou dossier personnalisé nécessaire au fonctionnement du composant (xliff, images...). En revanche, il ne peut pas contenir de plug-ins ou de sous-dossiers Components. Si ces éléments sont présents, ils sont ignorés par 4D.

Les plug-ins utilisés par les composants doivent être installés dans la base hôte ou dans 4D.

Interprété / compilé / Unicode

Une base hôte exécutée en mode interprété peut utiliser indifféremment des composants interprétés ou compilés, en mode Unicode ou non (cf. paragraphe “Prise en charge de l’Unicode”, page 67). Il est possible d’installer des composants interprétés et compilés dans la même base hôte. Toutefois, si plusieurs composants compilés sont présents, ils doivent être exécutés dans le même mode Unicode.

Une base hôte exécutée en mode compilé ne peut pas utiliser de composant interprété. Dans ce cas, seuls des composants compilés peuvent être employés. De même, le mode Unicode doit être identique pour la base hôte et les composants.

Le tableau suivant résume les possibilités d'usage des composants :

		Composants interprétés		Composants compilés	
		Unicode	*Non Unicode*	*Unicode*	*Non Unicode*
Base hôte interprétée	*Unicode*	√		√ (*)	
	Non Unicode				
Base hôte compilée	*Unicode*	-	-	√	-
	Non Unicode	-	-	-	√

(*) Si plusieurs composants compilés sont installés, ils doivent fonctionner dans le même mode Unicode.

Notes

- Une base hôte interprétée contenant des composants interprétés peut être compilée si elle ne fait pas appel à des méthodes du composant interprété. Dans le cas contraire, une boîte de dialogue d'alerte apparaît lorsque vous sélectionnez la commande de menu **Compilateur** et la compilation est impossible.
- De façon générale, une méthode interprétée peut appeler une méthode compilée et non l'inverse, hormis via l'utilisation des commandes EXECUTER METHODE et EXECUTER FORMULE.

Pour plus d'informations sur les échanges inter-composants et base hôte-composants, reportez-vous au paragraphe "Interaction entre les composants et les bases hôtes", page 59.

Mac OS / Windows

Un composant interprété développé sous Mac OS peut être installé dans un environnement Windows et inversement.
En revanche, la plate-forme d'exécution des composants compilés doit correspondre à la plate-forme de compilation, ou bien ils doivent avoir été compilés pour les deux plates-formes.

Client-Serveur

Les composants installés dans la base serveur sont automatiquement transférés sur les postes clients via un mécanisme proche de celui des plug-ins.

En revanche, il est déconseillé de modifier un composant en client/serveur car les modifications seront stockées en local, le composant ne sera pas mis à jour sur le poste serveur.

Chargement des composants au démarrage

Les composants sont chargés à l'ouverture de la base hôte.

Si un composant contient du code compilé et du code interprété qui ne correspondent pas, un message d'erreur est affiché et le composant n'est pas chargé dans la base hôte.

Si un composant est manquant au démarrage, la base hôte s'ouvre normalement. Ce principe permet de mettre en place des applications utilisant des composants optionnels. Il est possible de vérifier la présence de composants à l'aide de la nouvelle commande LISTE COMPOSANTS.

Conflits de noms (masquage des méthodes)

A la différence des autres objets partagés (cf. paragraphe "Objets partagés et objets non partagés", page 60), les méthodes projet partagées ont une existence "physique" dans la base, elles ne sont pas créées par l'exécution de code.

Par conséquent, un conflit de nom peut se produire lorsqu'une méthode projet partagée du composant a le même nom qu'une méthode projet de la base hôte. Dans ce cas, lorsque du code est exécuté dans le contexte de la base hôte, c'est la méthode de la base hôte qui est appelée. Ce principe permet de "masquer" une méthode du composant avec une méthode personnalisée (par exemple pour obtenir une fonctionnalité différente).
Bien entendu, lorsque le code est exécuté dans le contexte du composant, c'est la méthode du composant qui est appelée.

Un masquage est signalé par un *warning* en cas de compilation de la base hôte.

Note Si deux composants partagent des méthodes du même nom, une erreur est générée au moment de la compilation de la base hôte

Développement d'un composant

Un composant se présentant désormais sous la forme d'une base de données 4D, le développement d'un composant est semblable au développement d'une fonctionnalité dans une base.

Il existe toutefois des restrictions et des règles spécifiques liées à la nature des composants.

Objets utilisables et non utilisables

Un composant peut faire appel à la plupart des objets de 4D : méthodes projet, formulaires projet, barres de menus, énumérations, images de la bibliothèque, etc.

Seuls les objets suivants ne peuvent pas être utilisés par un composant :

- tables et champs,
- formulaires table et leurs méthodes formulaire associées (en revanche un composant peut appeler un formulaire table de la base hôte),
- formulaires utilisateurs,
- méthodes base et triggers.

Il n'est pas nécessaire de supprimer ces éléments s'ils existent dans les bases matrices. Lorsqu'un objet "non utilisable" est présent, il est simplement ignoré une fois le composant installé.

Note Les utilisateurs et groupes ainsi que les éventuels droits d'accès définis dans la base matrice sont ignorés dans la base hôte.

Seules les méthodes projet "partagées" par le composant seront visibles et sélectionnables en mode Développement dans la base hôte. A l'inverse, les méthodes projet "partagées" de la base hôte peuvent être appelées par le composant. Pour plus d'informations sur ce point, reportez-vous au paragraphe "Partage des méthodes projet", page 61.

Les autres objets du composant (formulaires projet, énumérations, menus, etc.) pourront être utilisés par le composant mais ne seront pas accessibles en tant qu'objets de développement depuis la base hôte. A noter que certains objets sont *cloisonnés* et d'autres sont *partagés* entre la base hôte et le(s) composant(s). Pour plus d'informations sur ce point, reportez-vous au paragraphe "Objets partagés et objets non partagés", page 60.

Un composant peut utiliser les plug-ins installés dans l'application 4D ou dans la base hôte. Il n'est pas possible d'installer des plug-ins dans le dossier du composant (cf. paragraphe "Installation d'un composant", page 52).

Les méthodes base et les paramètres génériques des bases matrices (dossier Web, préférences, etc.) ne sont jamais pris en compte.

Commandes non utilisables

Les commandes suivantes ne sont pas compatibles avec une utilisation dans le cadre d'un composant car elles modifient le fichier de structure — ouvert en lecture. Leur exécution dans un composant provoque l'erreur -10511, "La commande *NomCommande* ne peut pas être appelée depuis un composant" :

- APPELER SUR EVENEMENT

- Méthode appelee sur evenement
- ECRIRE IMAGE DANS BIBLIOTHEQUE
- SUPPRIMER IMAGE DANS BIBLIOTHEQUE
- STOCKER LISTE
- TABLEAU VERS ENUMERATION
- MODIFIER FORMULAIRE
- CREER FORMULAIRE UTILISATEUR
- SUPPRIMER FORMULAIRE UTILISATEUR
- CHANGER MOT DE PASSE
- CHANGER PRIVILEGES
- Ecrire proprietes groupe
- Ecrire proprietes utilisateur
- SUPPRIMER UTILISATEUR
- CHANGER LICENCES
- BLOB VERS UTILISATEURS
- ECRIRE ACCES PLUGIN

Note La commande Table du formulaire courant retourne Nil lorsqu'elle est appelée dans le contexte d'un formulaire projet. Par conséquent, elle ne peut pas être utilisée dans un composant.

Utilisation de formulaires

Seuls les "formulaires projet" (formulaires non associés à une table en particulier) peuvent être exploités directement dans un composant. Tous les formulaires projet présents dans la base matrice peuvent être utilisés par le composant.

Pour créer un formulaire projet dans la base matrice, affichez la page "Formulaires" de l'Explorateur, sélectionnez l'élément **Formulaires projet** puis cliquez sur le bouton d'ajout . Un formulaire projet dispose des mêmes propriétés que les formulaires standard : il peut comporter une barre de menus associée, un méthode formulaire, etc.

Dans une base hôte, il n'est pas possible de visualiser ni d'appeler explicitement depuis le mode Développement un formulaire projet provenant d'un composant. Ces formulaires n'apparaissent pas dans la page "Formulaires" de l'Explorateur ni dans l'éditeur de méthodes. Ils ne peuvent être appelés que via les méthodes projet du composant.

Pour plus d'informations sur les formulaires projet, reportez-vous au paragraphe "Formulaires projet", page 143.

Un composant peut faire appel à des formulaires table de la base hôte. A noter qu'il est nécessaire dans ce cas d'utiliser des pointeurs plutôt que des noms de table entre [] pour désigner les formulaires dans le code du composant.

Note Si un composant utilise la commande AJOUTER ENREGISTREMENT, le formulaire Entrée courant de la base hôte sera affiché, dans le contexte de la base hôte. Par conséquent, si le formulaire comporte des variables, le composant n'y aura pas accès (cf. paragraphe "Interaction entre les composants et les bases hôtes", page 59).

Utilisation de ressources

Les composants peuvent utiliser des ressources (ressources Mac OS "classiques" ou fichiers de type XLIFF).

Conformément aux nouveaux principe de gestion des ressources (cf. paragraphe "Gestion des ressources", page 47), les fichiers de ressources des composants doivent être placés dans un dossier Resources, situé à côté du fichier .4db ou .4dc du composant. Si le composant est d'architecture .4dbase (architecture conseillée), le dossier Resources doit être placé à l'intérieur de ce dossier.

Les mécanismes automatiques sont opérationnels : les fichiers XLIFF présents dans le dossier Resources d'un composant seront chargés par ce composant. Un composant utilisera aussi automatiquement les ressources Mac OS "classiques" situées dans le fichier .rsr à côté du fichier .4db ou .4dc (compatibilité).

Les fichiers de ressources "classiques" situés dans le dossier Resources devront être explicitement chargés dans le composant à l'aide des commandes du thème "Ressources".

Dans une base hôte contenant un ou plusieurs composant(s), chaque composant ainsi que la base hôte dispose de sa propre "chaîne de ressources". Les ressources sont cloisonnées entre les différentes bases : il n'est pas possible d'accéder aux ressources du composant A depuis le composant B ou la base hôte (cf. paragraphe "Objets partagés et objets non partagés", page 60).

Aide en ligne des composants

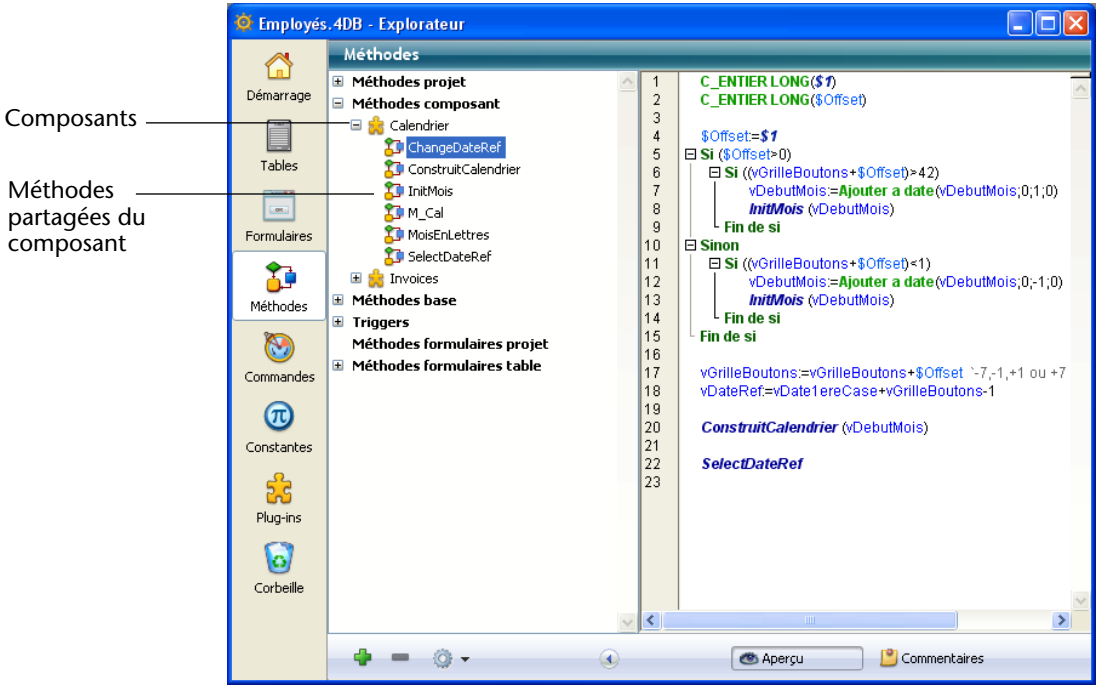
Un mécanisme spécifique a été mis en place afin de permettre aux développeurs d'ajouter des aides en ligne à leurs composants. Le principe est semblable à celui proposé pour les bases de données 4D :

- l'aide du composant doit être fournie sous le forme d'un fichier suffixé *.htm*, *.html* ou (Windows uniquement) *.chm,*
- le fichier d'aide doit être placé à côté du fichier de structure du composant,
- l'aide est alors automatiquement chargée dans le menu **Aide** de l'application avec le libellé "**Aide de...**" suivi du nom du fichier d'aide.

Interaction entre les composants et les bases hôtes

Visualisation des composants

Lorsqu'un composant a été installé dans une base hôte, son nom apparaît dans la page Méthodes de l'Explorateur de la base hôte, dans le thème **Méthodes composant**. Les méthodes partagées des composants sont listées sous forme hiérarchique et, si le composant est interprété, leur contenu peut être visualisé dans la zone de prévisualisation.



Note Pour plus d'informations sur la définition des méthodes partagées, reportez-vous au paragraphe "Partage des méthodes projet", page 61.

Objets partagés et objets non partagés

Certains types d'objets définis par un composant évoluent dans un espace d'exécution qui leur est propre, ce qui élimine les possibilités de conflit avec les objets existants de la base hôte et ceux des autres composants. Ces objets sont appelés "non partagés" ou "cloisonnés". Par exemple, les variables sont cloisonnées, ce qui signifie qu'une variable <>*Mavar* de type Entier long créée et manipulée par un composant peut coexister avec une variable <>*Mavar* de type Texte créée et manipulée par la base hôte (ou un autre composant).

D'autres objets partagent le même espace d'exécution entre la base hôte et les composants. L'emploi de ces objets nécessite davantage de précautions mais permet à la base hôte et aux composants de communiquer. Ces objets sont appelés "partagés" ou "non cloisonnés".

Par exemple, les ensembles sont partagés, ce qui signifie que si un composant crée l'ensemble *monEnsemble*, il sera supprimé si la base hôte exécute l'instruction EFFACER ENSEMBLE(monEnsemble).

- Les **objets non partagés** (cloisonnés) sont les suivants :
  - Feuilles de style
  - Info-bulles
  - Enumérations
  - Images de la bibliothèque
  - Menus et barres de menus créés via l'éditeur de menus
  - Méthodes projet sans la propriété "Partager la méthode"
  - Sémaphores
  - Process
  - Variables (locales, process et interprocess)
  - Variables système (OK, Document, etc.)
  - Formulaires projet et formulaires table
  - Ressources et références aux fichiers de ressources ouverts
- Les **objets partagés** (non cloisonnés) sont les suivants :
  - Ensembles
  - Sélections temporaires

- Listes hiérarchiques utilisant une référence (créées via les commandes Nouvelle liste, Charger liste, Copier liste ou BLOB vers liste)
- Menus et barres de menus utilisant la référence retournée par la nouvelle commande Creer menu
- Méthodes projet ayant la propriété “Partager la méthode”
- Références de structure XML
- Références de fichiers ouverts (hormis fichiers de ressources)
- Pointeurs

Note Bien entendu, les objets non utilisables présents dans une base matrice sont ignorés par la base hôte (cf. paragraphe “Objets utilisables et non utilisables”, page 55).

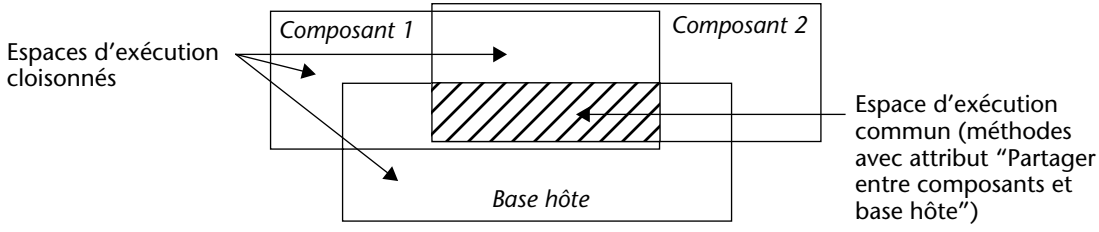
Partage des méthodes projet

Toutes les méthodes projet d’une base matrice sont par définition incluses dans le composant (la base est le composant), ce qui signifie qu’elles peuvent être appelées et exécutées par le composant.

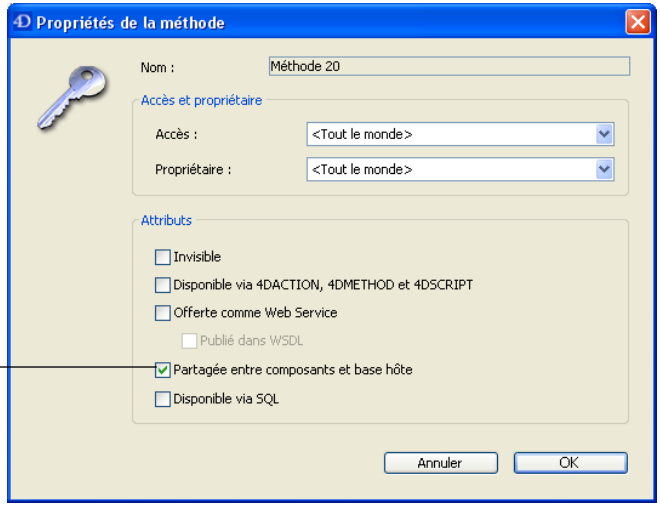
En revanche, par défaut ces méthodes projet ne seront ni visibles ni appelables par la base hôte. Vous devez explicitement désigner dans la base matrice les méthodes que vous souhaitez partager avec la base hôte. Ces méthodes projet seront visibles dans la page Méthodes de l’Explorateur (cf. paragraphe “Visualisation des composants”, page 59) et pourront être appelées dans le code la base hôte (mais elles ne pourront pas être modifiées dans l’éditeur de méthodes de la base hôte). Ces méthodes constituent les **points d’entrée** dans le composant.

A l’inverse, pour des raisons de sécurité, par défaut un composant ne peut pas exécuter de méthode projet appartenant à la base hôte. Dans certains cas, vous pourrez avoir besoin d’autoriser un composant à accéder à des méthodes projet de votre base hôte. Pour cela, vous devez explicitement désigner les méthodes projet de la base hôte que vous souhaitez rendre accessibles aux composants.

Partage des méthodes

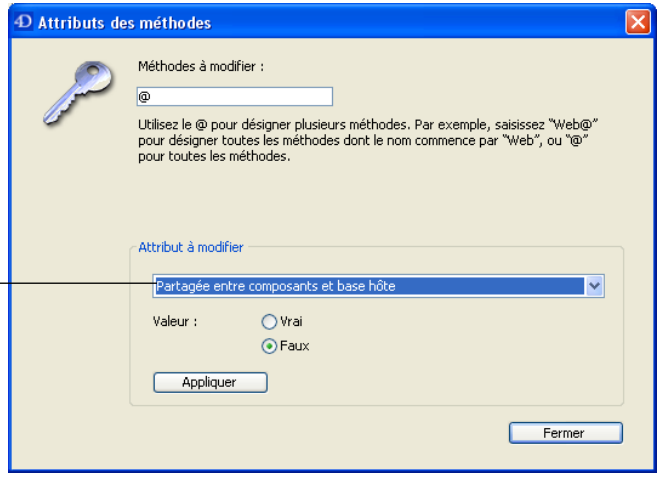


Ce paramétrage est effectué via la nouvelle propriété **Partager entre composants et base hôte** dans la boîte de dialogue des propriétés des méthodes :



Nouvelle option d'accessibilité de la méthode

Il est également possible d'appliquer cette propriété à plusieurs méthodes à la fois via la boîte de dialogue d'attributs globaux (accessible depuis le menu contextuel de l'Explorateur) :



Nouvelle option d'accessibilité de la méthode

L'effet de cette option est défini par le contexte d'utilisation de la base : si la base est utilisée comme composant, la méthode sera accessible dans la base hôte et visible dans l'Explorateur. Si la base est une base hôte, la méthode sera utilisable par les composants.

Passage de variables

Les composants et les bases hôtes ne partagent pas de variable locale, process ou interprocess. Le seul moyen d'accéder aux variables du composant depuis la base hôte et inversement est d'utiliser des pointeurs.

▼ Exemple utilisant un tableau :

- Dans la base hôte :
  TABLEAU ENTIER(MonTab;10)
  UneMéthode(->MonTab)
- Dans le composant, la méthode projet *UneMéthode* contient :
  AJOUTER A TABLEAU($1->; 2)

▼ Exemples utilisant des variables :

C_TEXTE(mavariable)
methode1_du_composant(->mavariable)

C_POINTEUR($p)
$p:=*methode2_du_composant(...)*

L'utilisation de pointeurs pour faire communiquer les composants et la base hôte nécessite de prendre en compte les spécificités suivantes :

- La commande Pointeur vers ne retournera pas un pointeur vers une variable de la base hôte si elle est appelée depuis un composant et inversement.
- La nouvelle architecture des composants autorise la coexistence, au sein d'une même base interprétée, de composants interprétés et compilés (à l'inverse, seuls des composants compilés peuvent être utilisés dans une base compilée).
  L'usage de pointeurs dans ce cas doit respecter le principe suivant : l'interpréteur peut dépointer un pointeur construit en mode compilé mais à l'inverse, en mode compilé, il n'est pas possible de dépointer un pointeur construit en mode interprété.
  Illustrons ce principe par l'exemple suivant : soient deux composants, C (compilé) et I (interprété) installés dans la même base hôte.
  - si le composant C définit la variable *mavarC*, le composant I peut accéder à la valeur de cette variable en utilisant le pointeur *->mavarC*.
  - si le composant I définit la variable *mavarI*, le composant C ne peut pas accéder à cette variable en utilisant le pointeur *->mavarI*. Cette syntaxe provoque une erreur d'exécution.

- La comparaison de pointeurs via la commande RESOUDRE POINTEUR est déconseillée avec les composants car le principe de cloisonnement des variables autorise la coexistence de variables de même nom mais au contenu radicalement différente dans un composant et la base hôte (ou un autre composant). Le type de la variable peut même être différent dans les deux contextes.
Si les pointeurs *monptr1* et *monptr2* pointent chacun sur une variable, la comparaison suivante produira un résultat erroné :

```
RESOUDRE POINTEUR(monptr1;vNomVar1;vnumtable1;vnumchamp1)
RESOUDRE POINTEUR(monptr2;vNomVar2;vnumtable2;vnumchamp2)
Si(vNomVar1=vNomVar2)
   ` Ce test retourne Vrai alors que les variables sont différentes
```

Dans ce cas, il est nécessaire d'utiliser la comparaison de pointeurs :

```
Si(monptr1=monptr2)
   ` Ce test retourne Faux
```

Note La gestion des pointeurs sur les variables process a été modifée dans 4D v11 (cf. paragraphe "Pointeurs sur variables process", page 395).

Accès aux tables de la base hôte

Bien que les composants ne puissent pas utiliser de tables, les commandes suivantes peuvent être appelées dans le contexte d'un composant :

- TABLE PAR DEFAUT
- PAS DE TABLE PAR DEFAUT
- Table par defaut courante

En effet, ces commandes sont utiles lorsqu'un composant doit exploiter les tables de la base hôte. Les pointeurs permettent à la base hôte et au composant de communiquer dans ce cas.

Par exemple, voici une méthode pouvant être appelée depuis un composant :

```
C_ENTIER LONG($1) `Numéro de table de la base hôte
$pointeurtable:=Table($1)
TABLE PAR DEFAUT($pointeurtable->)
CREER ENREGISTREMENT `Utilise la table par défaut de la base hôte
$pointeurchamp:=Champ($1;1)
$pointeurchamp->:="valeur"
STOCKER ENREGISTREMENT
LIBERER ENREGISTREMENT
```

Portée des commandes du langage

Hormis les commandes interdites (cf. paragraphe “Commandes non utilisables”, page 56), un composant peut utiliser toute commande du langage 4D.

Lorsqu’elles sont appelées depuis un composant, les commandes s’exécutent dans le contexte du composant, à l’exception de la nouvelle commande EXECUTER METHODE qui utilise le contexte de la méthode désignée par la commande. A noter également que les commandes de lecture du thème “Utilisateurs et groupes” sont utilisables depuis un composant mais lisent les utilisateurs et les groupes de la base hôte (un composant n’a pas d’utilisateurs et groupes propres).

Les commandes FIXER PARAMETRE BASE et Lire parametre base constituent aussi une exception à ce principe : leur portée est globale à la base. Lorsque ces commandes sont appelées depuis un composant, elles s’appliquent à la base hôte.

Par ailleurs, les commandes Fichier structure et Dossier 4D ont été adaptées afin de permettre leur utilisation dans le cadre des composants (cf. routine Fichier structure, page 357 et routine Dossier 4D, page 358).

Note La nouvelle commande LISTE COMPOSANTS permet de connaître la liste des composants chargés par la base hôte (cf. routine LISTE COMPOSANTS, page 356).

Débogage

Lorsque vous utilisez des composants non compilés, leur code apparaît dans le débogueur standard de la base hôte.

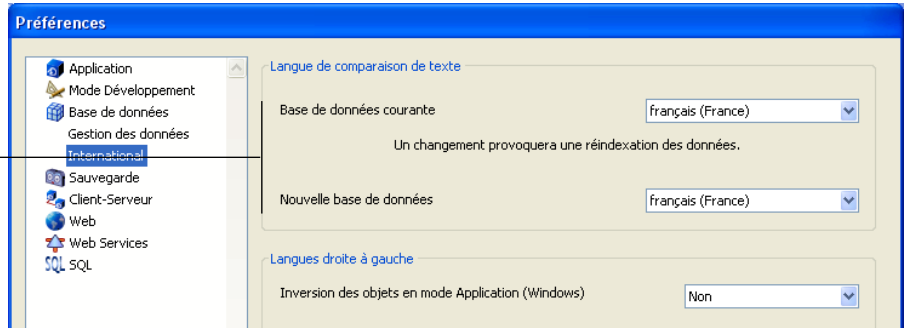
Le débogueur respecte les espaces d’exécution des objets cloisonnés. Si vous affichez la valeur de la variable *var1* de la base hôte dans la fenêtre d’évaluation puis exécutez du code appartenant au composant contenant également une variable *var1*, la valeur affichée ne sera pas mise à jour. Vous devez afficher une autre instance de la variable dans la fenêtre d’évaluation pour obtenir sa valeur dans le contexte courant.

Gestion des langues

La prise en charge de la langue utilisée pour le traitement et la comparaison des chaînes de caractères a été enrichie dans 4D v11 SQL.

Le paramétrage de la langue s'effectue dans la boîte de dialogue des préférences, onglet **Base de données/International** :

Options de sélection de la langue



Par défaut, 4D utilise la langue du système. Le choix d'une langue influe sur le tri et la recherche des textes ainsi que le passage en minuscules/majuscules mais n'a pas d'incidence sur la traduction des libellés ou sur les formats de dates, d'heure ou monétaires qui restent, eux, dans la langue du système.

Comme dans les versions précédentes de 4D, il est possible de choisir une langue pour la base de données ouverte (besoin ponctuel) et la langue par défaut pour les nouvelles bases (besoin permanent). Le choix de langues est toutefois plus vaste que dans les versions précédentes car il est basé sur la librairie ICU et non plus sur les ressources TRIC.

A la différence des versions précédentes, une base 4D peut désormais fonctionner dans une langue différente de celle du système. A l'ouverture d'une base, le moteur de 4D détecte la langue utilisée par le fichier de données et la fournit au langage (interpréteur ou mode compilé). Les comparaisons de texte, qu'elles soient effectuées par le moteur de base de données ou par le langage, sont donc toujours effectuées dans la même langue.

Après sélection d'une langue pour la base de données courante et validation de la boîte de dialogue des préférences, le code de la langue est stocké dans la structure et les données puis les index sont immédiatement reconstruits. Il est nécessaire de relancer la base pour

que les comparaisons effectuées par le langage prennent en compte ce paramétrage.

Lors de la création d'un nouveau fichier de données, 4D utilise la langue préalablement définie dans les préférences. En cas d'ouverture d'un fichier de données qui n'est pas dans la même langue que la structure, la langue du fichier de données est utilisée et le code de langue est recopié dans la structure.

Prise en charge de l'Unicode

La prise en charge du jeu de caractères Unicode a été étendue dans 4D v11. Désormais, le moteur de la base de données et le langage manipulent nativement les chaînes de caractères en Unicode. Cette nouveauté accélère les traitements basés sur les chaînes de caractères et facilite l'internationalisation des applications 4D.

Pour des raisons de compatibilité, 4D v11 peut toujours fonctionner dans le mode précédent (basé sur le jeu de caractères ASCII Mac) via une nouvelle option des Préférences.

Qu'est-ce que l'Unicode ?

L'Unicode un jeu de caractères standard unifié qui gère pratiquement toutes les langues usuelles de la planète. Un jeu de caractères est une table de correspondance *caractère/valeur numérique*, par exemple "a"->1, "b"->2, "5"->15, "œ"->662, etc.

Dans les versions précédentes de 4D, la gestion multi-langue reposait sur le principe "une langue = un jeu de caractères". 4D sélectionnait au démarrage le jeu de caractères correspondant à la langue du système. Ce principe avait pour conséquence l'impossibilité pour 4D de gérer plusieurs langues simultanément.

Alors qu'en *us-ascii* la valeur numérique de base est typiquement comprise entre 1 et 127, en Unicode la borne haute va au-delà de 65000, ce qui permet de représenter quasiment tous les caractères de toutes les langues.

Il existe différentes manières de coder les valeurs numériques Unicode : UTF-16 les code sur des entiers de 16-bits, UTF-32 sur des entiers de 32-bits et UTF-8 sur des entiers de 8-bits. Il existe même un UTF-7.

4D v11 utilise principalement UTF-16 (comme Windows et Mac OS). Parfois, essentiellement pour des besoins liés au Web, 4D utilise UTF-8

qui a l'avantage de la compacité et de la lisibilité pour les caractères usuels (a-z,0-9).

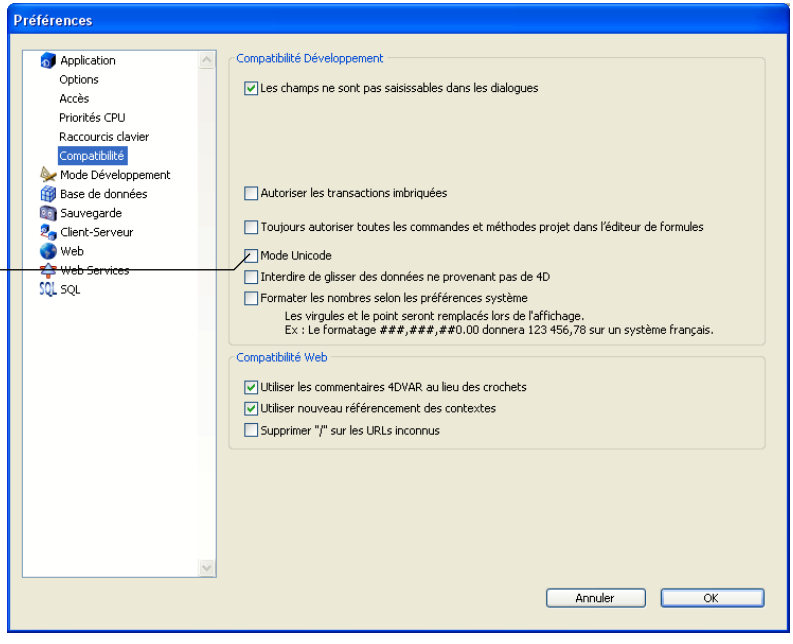
Il est à noter que les valeurs Unicode de 1 à 127 correspondent exactement au jeu de caractères *us-ascii* (Mac et Windows). Ces caractères étant les plus courants (a-z, 0-9, etc...), le passage en Unicode n'a pas d'incidence directe dans la plupart des cas.

Compatibilité de l'Unicode avec les bases 4D

Le passage des applications 4D en jeu de caractères Unicode est transparent pour l'utilisateur, le programme se chargeant en interne des conversions si nécessaire.

Toutefois, certaines instructions du langage, utilisant notamment les commandes travaillant avec les chaînes de caractères, pourront nécessiter des adaptations. Par exemple, l'instruction Caractere(200) ne retournera pas la même valeur en Unicode et en Ascii. Les spécificités de la prise en charge d'Unicode dans le langage 4D sont détaillées dans le paragraphe "Modifications liées à l'Unicode", page 258.

Afin de préserver la compatibilité du code existant, la nouvelle option **Mode Unicode** placée dans la page Application/Compatibilité des Préférences permet d'activer ou de désactiver l'utilisation d'Unicode dans la base 4D :



Activation d'Unicode dans la base

Par défaut, cette option est désélectionnée dans les bases de données converties depuis une version précédente de 4D. Elle est cochée par défaut pour les bases créées avec 4D v11 et suivantes.

Cette option est spécifique à chaque base de données. Il est donc possible de faire cohabiter une base Unicode avec des composants non Unicode (ou inversement) en mode interprété. Pour plus d'informations sur les composants dans 4D v11, reportez-vous au paragraphe "Nouvelle architecture des composants", page 50.

Note Il est également possible de configurer le mode Unicode à l'aide des commandes Lire parametre base et FIXER PARAMETRE BASE. Pour plus d'informations sur ce point, reportez-vous au paragraphe "FIXER PARAMETRE BASE, Lire parametre base", page 359.

Propriété "Configuration du clavier"

Lorsque le mode Unicode est activé, la propriété "Configuration du clavier" des objets de formulaire est ignorée.

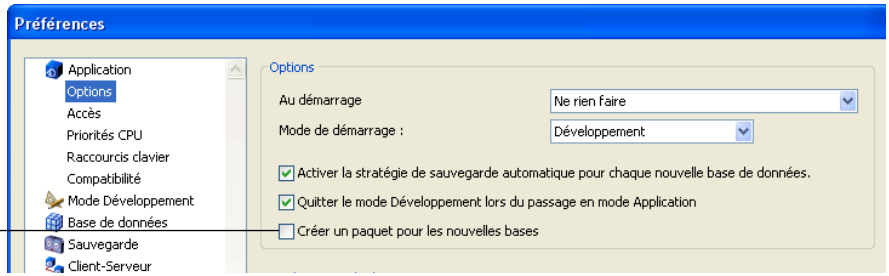
Dossier .4dbase

Par défaut, les bases de données 4D créées en version 11 sont automatiquement placées dans un dossier suffixé **.4dbase**. Par exemple, une base nommée "Factures" sera créée dans le dossier [*Factures.4dbase*]. Ce dossier stocke tous les éléments nécessaires au fonctionnement de la base.

Grâce à ce principe, sous Mac OS les dossiers des bases apparaissent sous forme de paquets (*packages*). Il est possible de double-cliquer sur le paquet afin de lancer directement 4D, la base et le fichier de données courant. Il est également possible de glisser-déposer le paquet sur l'icône de l'application 4D. Enfin, ce principe permet de placer la base dans un outil de gestion de versions (tel que *CVS* ou *Subversion*).

Sous Windows, ce fonctionnement n'a pas d'incidence particulière.

Vous pouvez désactiver ce fonctionnement par défaut en désélectionnant l'option **Créer un paquet pour les nouvelles bases** dans les préférences, page "Application/Options" :



Option de création de "packages"

4 Interface et navigation en mode Développement

L'interface générale de l'environnement de développement de 4D v11 a été en grande partie repensée pour offrir davantage de simplicité et d'ergonomie. Les modes Structure et Utilisation ont été fusionnés et de nombreuses nouvelles fonctions sont disponibles.

Ce chapitre décrit les nouveautés concernant :

- la création et l'ouverture des bases de données 4D,
- le mode Développement et les menus de 4D,
- l'Explorateur,
- le compilateur,
- l'exécution des méthodes base,
- le déplacement d'objets en mode Développement,
- le rechercher / remplacer dans toute la base,
- la gestion des raccourcis clavier en mode Développement,
- l'éditeur de feuilles de style,
- la traduction des bases de données,
- le nouvel opérateur "contient mot-clé" pour les recherches.

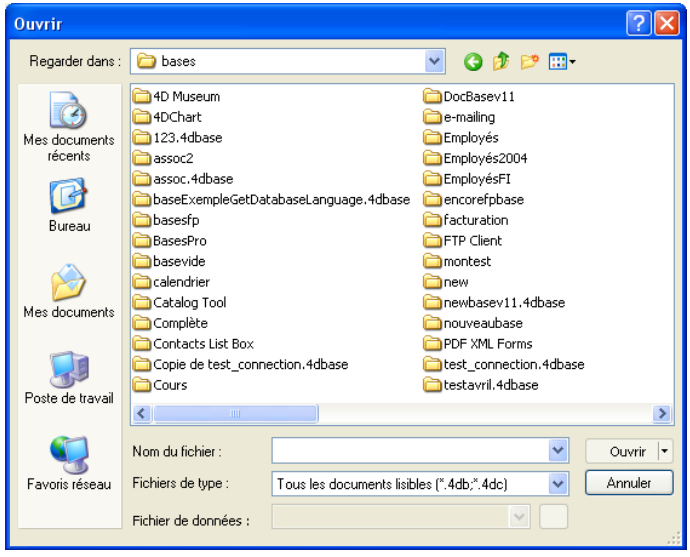
Ouverture et création des bases de données

L'ouverture et la création de bases de données ont été modifiées dans 4D v11. Ces opérations sont désormais effectuées via des boîtes de dialogue système.

Boîte de dialogue d'ouverture

Les principes d'ouverture et de création des bases de 4D v11 ont été simplifiés.
La boîte de dialogue de bienvenue disparaît au profit de boîtes de dialogue système. La notion de "bases favorites", en doublon avec celle de "bases récentes", disparaît. Les informations liées aux fichiers de sauvegarde et les commandes de vérification sont reportées dans le nouveau Centre de sécurité et de maintenance (cf. chapitre "Centre de Sécurité et de Maintenance", page 185).

Désormais, lorsque vous double-cliquez sur l'icône de l'application 4D, par défaut une boîte de dialogue standard d'ouverture de documents s'affiche :



Boîte de dialogue d'ouverture de bases (Windows)

Notes

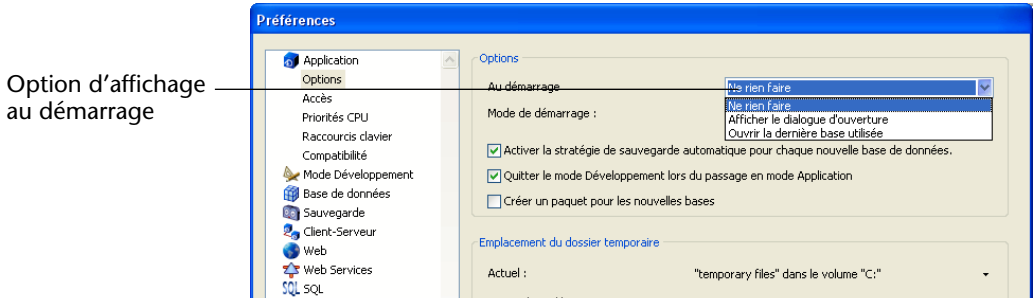
- Si vous cliquez sur **Annuler** dans cette boîte de dialogue après avoir double-cliqué sur l'icône de l'application, 4D s'exécute sans base courante. Vous pouvez alors sélectionner une commande de création ou d'ouverture dans le menu **Fichier** ou la commande **Centre de sécurité et de maintenance** dans le menu **Aide**.
- Vous pouvez configurer l'affichage à l'ouverture de la base via les préférences (cf. paragraphe "Préférences d'ouverture", page 73).

Cette boîte de dialogue comporte une zone d'options, permettant de sélectionner les options d'ouverture standard des bases 4D :

- Menu associé au bouton **Ouvrir** : ce menu permet de définir le mode d'ouverture de la base. Il propose les options **Base compilée** (si du code compilé est présent), **Base interprétée** (mode par défaut) et **Centre de sécurité et de maintenance** (mode sécurisé, permettant d'ouvrir une base de données endommagées afin d'effectuer d'éventuelles réparations). Pour plus d'informations sur cette option, reportez-vous au chapitre "Centre de Sécurité et de Maintenance", page 185.
- **Fichier de données** : ce menu permet de définir le fichier de données à utiliser avec la base. Par défaut, l'option **Fichier de données courant** est sélectionnés. La base s'ouvre avec ce fichier si vous cliquez sur le bouton **Ouvrir**.
  Pour changer de fichier de données, choisissez l'option **Choisir un autre fichier de données** ou **Créer un nouveau fichier de données** puis cliquez sur le bouton **Ouvrir**. Dans ce cas, la boîte de dialogue standard d'ouverture ou de création de fichier de données apparaît, vous permettant d'ouvrir ou de créer un fichier de données.

Préférences d'ouverture

Une nouvelle préférence, placée dans la page **Application/Options**, permet de configurer l'affichage proposé par défaut par 4D au démarrage, lorsque l'utilisateur lance uniquement l'application :



Les options proposées sont les suivantes :

- **Ne rien faire** : seule la fenêtre de l'application apparaît, vide.
- **Afficher le dialogue d'ouverture** (option par défaut) : 4D affiche la boîte de dialogue standard d'ouverture de documents.
- **Ouvrir la dernière base utilisée** : 4D ouvre directement la dernière base utilisée, aucune boîte de dialogue d'ouverture n'apparaît.

Note Pour forcer l'affichage de la boîte de dialogue d'ouverture lorsque l'option **Ouvrir la dernière base utilisée** est sélectionnée, maintenez enfoncée la touche **Alt** (Windows) ou **Option** (Mac OS) pendant le lancement de la base.

Menu Fichier

Une fois l'application 4D lancée, de nouvelles commandes du menu **Fichier** vous permettent de gérer l'ouverture et la création des bases :

- **Nouveau > Base de données à partir d'un modèle...** : cette commande permet d'accéder à l'assistant de création de bases de données. Dans les versions précédentes de 4D, cet assistant était accessible via le bouton **Utiliser un modèle** dans la boîte de dialogue d'ouverture. Cet assistant permet de générer une base de données opérationnelle à l'aide de modèles prédéfinis, dans différents thèmes.
- **Ouvrir bases récentes** > ***liste des bases récemment ouvertes*** : cette commande affiche un sous-menu listant toutes les bases ouvertes sur le poste. La commande **Effacer le menu** permet de réinitialiser le contenu de ce menu.
- **Fermer base** : cette commande permet de refermer la base de données courante. L'application 4D reste lancée mais il n'y a plus de base ouverte. Vous pouvez alors sélectionner une commande de création ou d'ouverture dans le menu **Fichier** ou la commande **Centre de sécurité et de maintenance** dans le menu **Aide**.

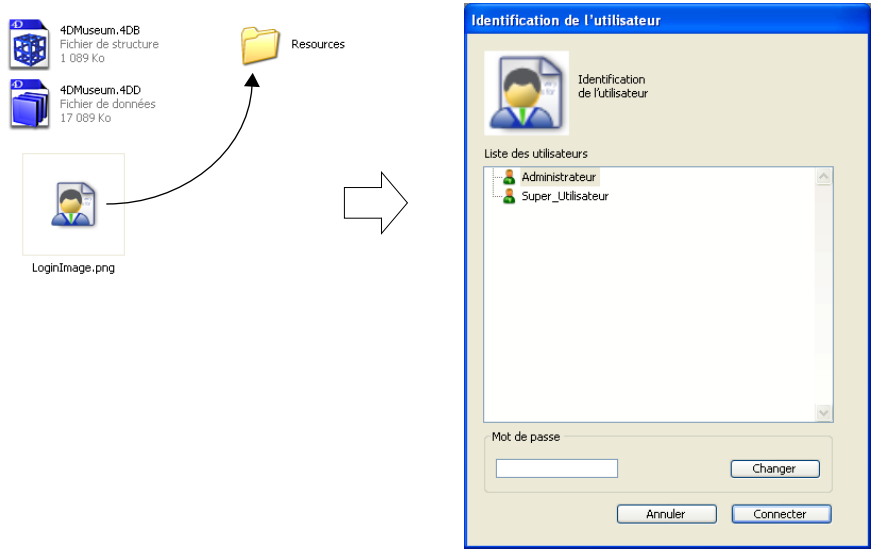
Note Les bases de données 4D créées en version 11 sont automatiquement placées dans un dossier suffixé **.4dbase**. Pour plus d'informations, reportez-vous au paragraphe "Dossier .4dbase", page 69.

Personnalisation de l'icône de la fenêtre d'identification

Il est possible de personnaliser l'icône affichée dans la boîte de dialogue de connexion à une base de données. Par défaut, cette icône représente le logo de 4D.

Pour remplacer cette icône par celle de votre choix, il vous suffit de placer un fichier nommé **LoginImage.png** dans le dossier **Resources** de la base (situé à côté du fichier de structure de la base).
Le fichier personnalisé doit être de type "png" et sa taille doit être de 80x80 pixels.

A l'ouverture de la base, 4D chargera cette image à la place de l'image par défaut :



Interface du mode Développement

L'interface globale de l'atelier de développement de 4D a été modifiée. En premier lieu, la distinction stricte entre les modes Structure et Utilisation a été supprimée. En outre, le mode Menus créés a été renommé. Il n'existe plus désormais que deux modes : Développement et Application.

Mode Développement

Le nouveau mode Développement regroupe les fonctions auparavant disponibles dans les modes Structure et Utilisation. Les éditeurs utilisés pour le développement de l'application (formulaires, menus, etc.) et les fenêtres permettant l'accès aux enregistrements des tables sont accessibles dans une seule et unique barre de menus. Le passage d'un type d'environnement à l'autre ne s'effectue plus explicitement via un menu **Mode** mais découle implicitement des fonctions que vous activez. Par exemple, si vous vous trouvez dans l'éditeur de formulaires et choisissez la commande **Nouvel enregistrement** dans le menu **Enregistrements**, la fenêtre principale de saisie des enregistrements passe au premier plan, affichant le formulaire d'entrée de la table courante.

Dans 4D v11, tous les menus sont désormais disponibles en permanence — dans les versions précédentes de 4D, la barre de menus et la barre d'outils étaient modifiées en fonction du mode courant. Désormais, seuls les menus spécifiques des éditeurs sont ajoutés dans la barre courante. La barre de menus du mode Développement contient en permanence les menus **Fichier**, **Edition**, **Exécution**, **Développement**, **Enregistrements**, **Outils** et **Fenêtres**, donnant accès à toutes les parties de 4D (cf. paragraphe "Réorganisation des menus", page 76).

Mode Application

Le nouveau mode Application est strictement équivalent au mode Menus créés. Pour y accéder, il suffit de sélectionner la commande **Tester l'application** dans le menu **Exécution**. Cette commande est accessible dès qu'une barre de menus personnalisée au moins a été créée dans l'application.

Actions standard

Deux nouvelles actions standard permettent de gérer la navigation entre le mode Application et le mode Développement : **Retour au mode Développement** et **Application**.
Dans les bases de données créées en version 11 et suivantes, les actions standard Utilisation et Structure ne sont plus disponibles.
Dans les bases de données converties, ces actions standard sont conservées, mais il n'est plus possible de les affecter à de nouvelles commandes de menus. Pour plus d'informations, reportez-vous au paragraphe "Actions standard", page 183.

Réorganisation des menus

Dans le cadre de la fusion des anciens modes Structure et Utilisation, les menus de la barre de 4D v11 ont été réorganisés. Désormais, tous les menus de la barre (et les icônes associées dans la barre d'outils) sont disponibles en permanence dans le mode Développement.

Fichier	Description	Mode version précédente de 4D
Nouveau>Base de données	*Idem version précédente*	*Tous*
Nouveau>Base de données à partir d'un modèle	Affiche la boîte de dialogue de sélection de modèle (cf. paragraphe "Menu Fichier", page 74)	*Nouveauté 4D v11*
Nouveau>Base de données à partir d'une définition de structure	Permet de créer une base d'après un fichier xml de définition de structure (cf. paragraphe "Créer une base de données à partir d'une définition de structure", page 134)	*Nouveauté 4D v11*

Nouveau>Fichier de données	*Idem version précédente*	Utilisation
Nouveau>Bibliothèque d'objets	*Idem version précédente*	Structure
Nouveau>Table/Formulaire/ Méthode	*Idem version précédente*	Structure
Ouvrir>Base de données	*Idem version précédente*	*Tous*
Ouvrir>Fichier de données	*Idem version précédente*	Utilisation
Ouvrir>Bibliothèque d'objets	*Idem version précédente*	Structure
Ouvrir>Formulaire/Méthode	*Idem version précédente*	Structure
Ouvrir bases récentes	Liste les bases récemment ouvertes (cf. paragraphe "Menu Fichier", page 74)	*Nouveauté 4D v11*
Fermer base	Referme la base courante (cf. paragraphe "Boîte de dialogue d'ouverture", page 72)	*Nouveauté 4D v11*
Fermer (objet)	*Idem version précédente*	Structure
Tout fermer	*Idem version précédente*	Structure
Sauvegarder (objet)	*Idem version précédente*	Structure
Tout enregistrer	*Idem version précédente*	Structure
Ecrire cache données	*Idem version précédente*	Utilisation
Version enregistrée	*Idem version précédente*	Structure
Importer>Du fichier	*Idem version précédente*	Utilisation
Importer>De la source ODBC	*Idem version précédente*	Utilisation
Exporter>Données vers le fichier	*Idem version précédente*	Utilisation
Exporter>Données vers la source ODBC	*Idem version précédente*	Utilisation
Exporter>Définition de structure vers le fichier XML	Permet de créer un fichier de définition de la base (cf. paragraphe "Exporter une définition de structure", page 132)	*Nouveauté 4D v11*
Exporter>Définition de structure vers le fichier HTML	Permet de créer un fichier de définition de la base (cf. paragraphe "Exporter une définition de structure", page 132)	*Nouveauté 4D v11*
Sauvegarde	*Idem version précédente*	Utilisation
Vérifier fichier d'historique	*Idem version précédente*	Utilisation
Format d'impression	*Idem version précédente*	*Tous*
Imprimer	*Idem version précédente*	*Tous*
Quitter	*Idem version précédente*	*Tous*

Edition	Description	Mode version précédente
Annuler/Répéter/Couper/Copier/ Coller/Effacer/Tout sélectionner	*Idem version précédente*	*Tous*
Dupliquer	Duplique l'objet sélectionné	*Nouveauté 4D v11*
Chercher dans le développement	*Idem version précédente*	Structure
Chercher>... (commandes de recherche)	*Idem version précédente*	Structure
Afficher le Presse-papiers	*Idem version précédente*	*Tous*
Préférences	*Idem version précédente*	*Tous*

Exécution	Description	Mode version précédente
Tester l'application	Exécute le mode Application sans redémarrage de la base (*équivaut la commande Menus créés dans le menu Mode de la version précédente*)	Utilisation/Structure
Méthode	*Idem version précédente*	*Tous*
Explorateur d'exécution	*Idem version précédente*	*Tous*
Démarrer (Arrêter) le serveur Web (*commande en bascule*)	*Idem version précédente*	*Tous*
Tester le serveur Web	*Idem version précédente*	*Tous*
Démarrer (Arrêter) le serveur SQL (*commande en bascule*)	Démarre ou stoppe le serveur SQL intégré de 4D (cf. paragraphe "Démarrer et stopper le serveur SQL", page 211)	*Nouveauté 4D v11*
Redémarrer en interprété/ Redémarrer en compilé	Redémarre la base et exécute le mode Application (si disponible) en interprété ou en compilé[1]	*Nouveauté 4D v11*

1. Ces commandes diffèrent de la commande en bascule En interprété/En compilé de la version précédente de 4D car elles provoquent l'ouverture de la base en mode Application.

Développement	**Description**	**Mode version précédente**
Ce menu correspond pour l'essentiel au menu Structure de la version précédente, il comprend une seule nouvelle commande		Structure
Lancer la compilation	Exécute directement la compilation de la base avec les paramètres courants, sans passer par la boîte de dialogue du compilateur	*Nouveauté 4D v11*

Enregistrements	**Description**	**Mode version précédente**
Afficher la table courante (Nom_table)	Affiche les enregistrements de la table courante dans le formulaire de sortie courant (le nom de la table courante est indiqué entre parenthèses). S'il n'y a pas de table courante, la commande est grisée	*Nouveauté 4D v11*
Dernières tables utilisées>... (Noms des tables utilisées)	Le sous-menu liste les 15 dernières tables utilisées (c'est-à-dire affichées) dans la base. La sélection d'un nom de table provoque l'affichage des enregistrements de cette table dans le formulaire de sortie courant	*Nouveauté 4D v11*
Liste des tables	*Idem version précédente*	Utilisation
Nouvel enregistrement Nouvel enregistrement en liste Modifier l'enregistrement	*Idem version précédente*	Utilisation
Suppprimer la sélection	Supprime du fichier de données les enregistrements surlignés dans la fenêtre (= commande Effacer)	*Nouveauté 4D v11*
Tout montrer Sous sélection Chercher>... Trier Appliquer une formule	*Idem version précédente*	Utilisation

Outils	**Description**	**Mode version précédente**
Ce menu correspond strictement au menu Outils de la version précédente		Utilisation

Barre d'outils

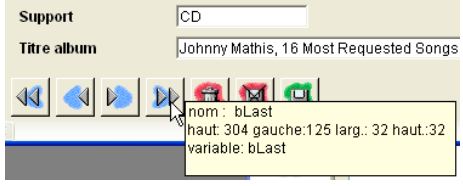
A l'instar de la barre de menus, la barre d'outils de 4D v11 est désormais invariable dans tous les environnements (hors mode Application). Elle permet un accès rapide aux principales fonctions de 4D :

Le nouveau bouton Tables équivaut à la nouvelle commande Dernières tables utilisées du menu Enregistrements (cf. ci-dessus)

Le nouveau bouton CSM affiche la fenêtre du Centre de Sécurité et de Maintenance (cf. chapitre "Centre de Sécurité et de Maintenance", page 185).

Informations sur les objets des formulaires

Une combinaison de touches permet au développeur d'obtenir à tout moment diverses informations utiles sur les objets des formulaires (nom, coordonnées, etc.). Ces informations sont affichées sous forme d'info-bulles qui apparaissent lorsque vous maintenez les touches **Ctrl+Majuscule** (Windows) ou **Commande+Majuscule** et placez le curseur au-dessus d'un objet :

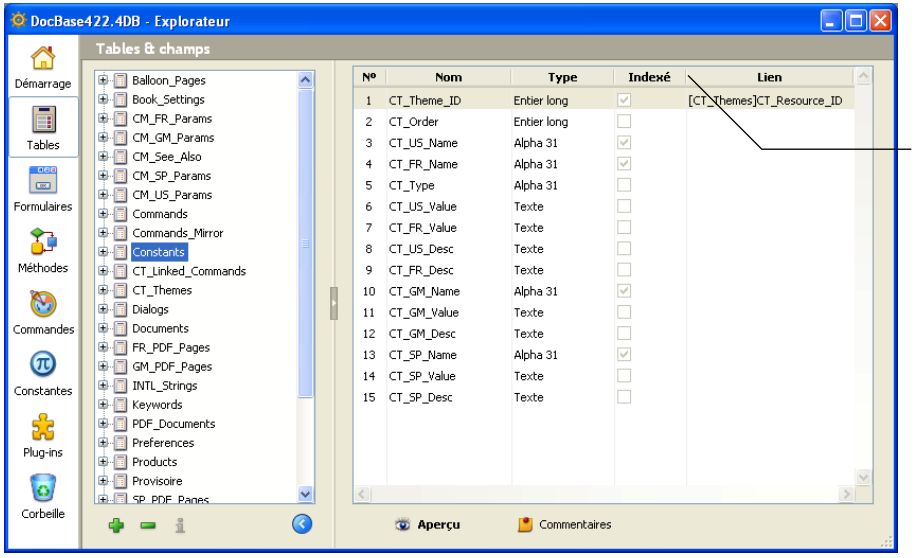


Cette aide à la mise au point des programmes est disponible pour chaque objet visualisé dans un formulaire lorsque l'environnement de développement est ouvert.

Explorateur

Visualisation des tables et des champs

Lorsqu'une table est sélectionnée dans la page **Tables** ou **Démarrage**, la zone d'aperçu de l'Explorateur affiche désormais la description de la table sous forme de liste :

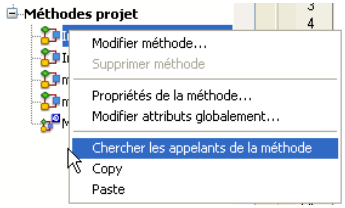


Cette zone est uniquement informative, il n'est pas possible de modifier les valeurs.
En plus des numéros, noms et types des champs de la table sélectionnée, le tableau liste les index associés à la table (quel que soit leur type) ainsi que les liens partant de la table. Pour chaque champ d'où part un lien (champ N), le champ d'arrivée du lien (champ 1) est indiqué dans la colonne Lien.

Rechercher les appelants

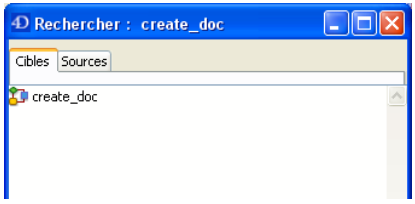
Une nouvelle fonction de l'Explorateur permet de rechercher la liste des objets référençant une méthode projet (autres méthodes ou menus). Cette fonction est accessible via la nouvelle commande

Chercher les appelants de la méthode dans le menu contextuel de l'Explorateur :



Note La commande **Chercher les appelants** est également disponible dans le menu contextuel de l'éditeur de méthodes (cf. paragraphe "Chercher les appelants", page 138).

Cette commande provoque l'affichage d'une nouvelle fenêtre de résultat. Les objets trouvés sont affichés dans la liste **Source** :



Pour plus d'informations sur cette fenêtre, reportez-vous au paragraphe "Nouvelle fenêtre de résultat", page 95.

Copier, coller ou dupliquer des méthodes projet

Il est désormais possible de copier, de coller et de dupliquer des méthodes projet directement depuis la liste des méthodes de l'Explorateur. Ces commandes permettent de dupliquer une méthode au sein de la même base de données ou de recopier une méthode d'une base à une autre.

Pour dupliquer une méthode, cliquez sur son nom dans la zone de liste avec le bouton droit de la souris et choisissez la commande **Dupliquer** dans le menu contextuel.

Pour copier une méthode, cliquez sur son nom dans la zone de liste avec le bouton droit de la souris et choisissez la commande **Copier** dans le menu contextuel. Pour coller la méthode, choisissez la commande **Coller** dans le même menu contextuel.

En cas de duplication ou si une méthode de même nom existe déjà à l'endroit ou vous collez la méthode, un numéro est accolé au nom de la méthode ajoutée, par exemple *Maméthode1*. Ce numéro est incrémenté si nécessaire (*Maméthode2, Maméthode3*, etc.).

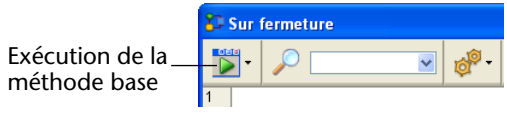
Page Méthodes

La page Méthodes de l'Explorateur a été réorganisée afin de permettre la prise en compte les méthodes des formulaires projet (cf. paragraphe "Formulaires projet", page 143). Elle est désormais constituée des éléments suivants :

- **Méthodes base** (inchangé)
- **Méthodes formulaires table** : affiche la liste des formulaires table de la base, classée par tables (correspond à la partie Méthodes formulaires de l'élément **Méthodes formulaire & triggers** de la version précédente).
- **Méthodes formulaires projet** : affiche la liste des formulaires projet de la base (nouveauté 4D v11).
- **Méthodes projet** (inchangé).
- **Triggers** : affiche la liste des tables de la base (correspond à la partie Triggers de l'élément **Méthodes formulaire & triggers** de la version précédente).

Exécution des méthodes base

Il est désormais possible d'éxécuter certaines méthodes bases directement depuis l'éditeur de méthodes, afin de pouvoir contrôler leur fonctionnement sans devoir quitter ou relancer la base :



Les méthodes base exécutables sont :

- Sur ouverture
- Sur fermeture
- Sur démarrage serveur
- Sur arrêt serveur

Déplacement d'objets en mode Développement

Dans 4D v11, vous disposez de la possibilité de déplacer des objets entre deux bases de données. Vous pouvez ainsi recopier dans une base les tables, formulaires, méthodes... que vous avez créés pour une autre base. Cette fonction facilite et accélère le développement de vos bases.

Note Dans les versions précédentes, cette fonction était prise en charge par le programme 4D Insider.

Le déplacement d'objets ne se limite pas aux objets individuels, il peut concerner un objet ainsi que tous ceux que cet objet référence, c'est-à-dire ses objets dépendants. Vous déplacez ainsi des fonctionnalités entières. Par exemple, si vous avez créé une boîte de dialogue de recherche personnalisée, vous pouvez déplacer le formulaire utilisé pour la boîte de dialogue ainsi que toutes les méthodes, images et autres objets qu'il utilise. Vous pouvez copier le formulaire dans une autre base ou dans une bibliothèque regroupant des fonctionnalités couramment utilisées dans vos bases.

Certains objets sont également indissociables : ils sont obligatoirement déplacés avec leurs objets "parents". La liste des objets indissociables est fournie ci-dessous.

Objets déplaçables

Les objets peuvent être déplacés à partir de la Boîte à outils, de l'Explorateur et de l'éditeur de formulaires. En outre, il est possible d'effectuer un déplacement depuis la fenêtre de résultat d'une recherche globale.

Pour des raisons de cohérence structurelle, la copie de certains objets entraîne la copie des objets qui leur sont indissociables. Par exemple, la copie d'un formulaire entraînera la copie de la méthode formulaire et des méthodes objet qui lui sont éventuellement attachées. Ces objets indissociables, quant à eux, ne peuvent être directement déplacés.

Voici la liste des objets déplaçables ainsi que leurs objets indissociables :

Objets déplaçables	Objets indissociables
Boîte à outils	
Enumérations	
Feuilles de style	

Objets déplaçables	Objets indissociables
Formats/Filtres	
Images de la bibliothèque d'images	
Infobulles	
Explorateur	
Formulaires projet	Méthodes formulaire
Formulaires table	Méthodes formulaire
Méthodes projet	
Dossiers / Sous-dossiers	
Tables	Champs, triggers
Editeur de formulaires	
Tous les objets d'un formulaire (boutons, variables, etc.). En cas de déplacement d'un formulaire, tous les objets qu'il contient sont déplacés.	Méthodes objet

Principes des déplacements

Le déplacement d'objets peut être effectué indifféremment via les fonctions standard de glisser-déposer ou de copier-coller.

Pour effectuer un déplacement entre deux bases de données par glisser-déposer, vous devez dupliquer votre version de 4D ou utiliser 4D et un 4D Client.

Dans le cas de déplacement inter-bases, les objets déplacés peuvent être collés ou déposés dans le même environnement que celui de départ (Boîte à outils, Explorateur, etc.) ou dans d'autres zones de l'application. 4D effectuera l'action appropriée en fonction du contexte, si cela a un sens. Par exemple, il est possible de déposer un formulaire dans une fenêtre de l'éditeur de méthodes, dans ce cas le nom du formulaire est inséré dans la méthode.

Lors du déplacement, si un objet de même type et de même nom existe déjà dans la base de destination, par défaut l'objet existant sera remplacé par l'objet déplacé. La boîte de dialogue de déplacement s'affiche dans ce cas ; elle indique les objets qui seront remplacés et vous permet de modifier cette action.

Les mécanismes suivants sont à noter :

- **Vues et plan** : Les objets de formulaire déplacés conservent leurs propriétés d'emplacement dans l'éditeur, notamment leur position dans les vues ou dans les plans du formulaire.

- **Formulaires hérités** : Les formulaires hérités ne sont pas déplacés avec les formulaires sources, toutefois leur référence est conservée. En outre, les formulaires hérités sont considérés comme objets dépendants et il est possible d'utiliser comme formulaire hérité un autre formulaire (existant) au moment du déplacement (cf. paragraphe "Objets dépendants des formulaires", page 86).
- **Droits d'accès** : Les formulaires et méthodes projet déplacés ne conservent pas leurs éventuels droits d'accès d'origine. La valeur par défaut ("Sans restriction") leur est automatiquement assignée.
- **Dossiers** : Lorsque vous déplacez un dossier depuis la page Démarrage de l'Explorateur, l'opération inclut le dossier et la totalité de son contenu (tables, formulaires et méthodes projet), ce qui peut représenter un volume de données important. Lors de ce type de déplacement, une boîte de dialogue d'alerte apparaît afin de préciser ce fonctionnement.

Note Il n'est pas possible de faire glisser des objets depuis la page **Corbeille** de l'Explorateur.

Objets dépendants des formulaires

Un formulaire (table ou projet) peut référencer divers autres objets comme des énumérations, des images, etc. : ces objets sont appelés *objets dépendants*.

Dans certains cas, vous pourrez avoir besoin de déplacer tous les objets dépendants, dans d'autres cas vous souhaiterez ne déplacer qu'une partie, voire aucun de ces objets.

4D vous permet de contrôler le déplacement des objets dépendants des formulaires via des Préférences de déplacement ainsi que via le Dialogue de déplacement.

Les Préférences de déplacement définissent les principes à appliquer pour le déplacement des objets dépendants. Vous pouvez choisir diverses options par défaut pour chaque type d'objet. Pour plus d'informations, reportez-vous au paragraphe "Préférences de déplacement", page 90.

Dialogue de déplacement

Lorsque vous déplacez une sélection d'objets (par glisser-déposer ou copier-coller) entre deux bases 4D ou entre une base de données et une bibliothèque d'objets, vous avez la possiblité d'afficher une boîte de dialogue listant tous les objets déplacés ainsi que les actions qui leur seront associés dans la base de destination.

Cette boîte de dialogue est nommée "Dialogue de déplacement". Elle s'affiche lorsqu'au moins un des cas suivants est vrai :

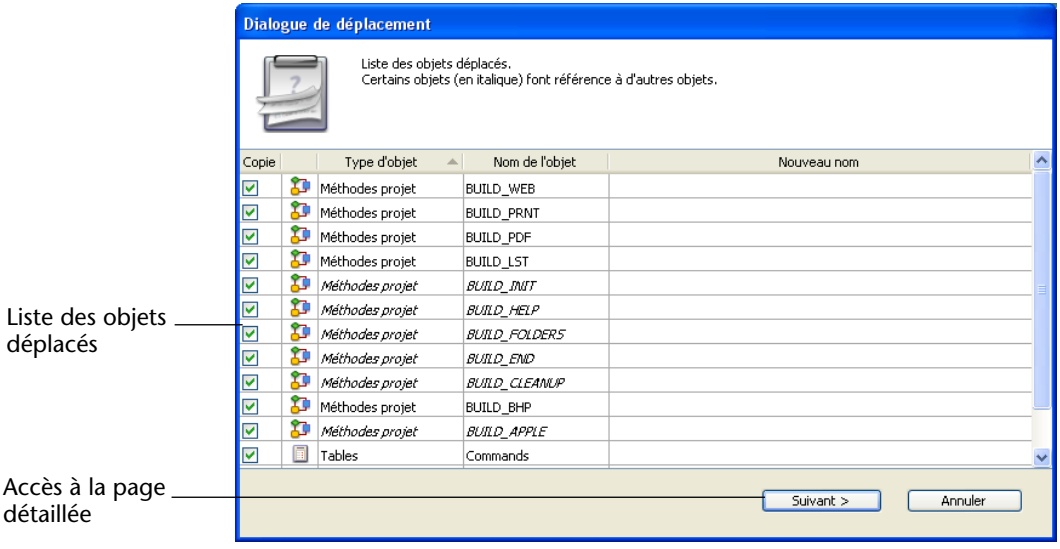
- L'option "Toujours afficher" est sélectionnée dans les préférences de déplacement (cf. paragraphe "Préférences de déplacement", page 90).
- Au moins un objet déplacé est en conflit de nom avec un objet de la base de destination.
- L'action par défaut **Utiliser un autre objet** a été sélectionnée pour au moins un type d'objet dépendant déplacé.

En-dehors de ces cas, si le déplacement ne soulève pas de conflit, le dialogue de déplacement n'apparaît pas et les objets sont directement copiés.

Cette boîte de dialogue vous permet de visualiser et/ou de modifier les paramétrages du déplacement en fonction du contexte. Elle comporte deux pages : la **page principale** et la **page détaillée**. Ces pages sont accessibles via les boutons **Suivant>** et **<Précédent**.

Page principale

La page principale affiche la liste des objets déplacés :



Les objets en conflit de nom apparaissent en caractères **gras** et les objets dépendants en caractères *italiques*. Les différentes colonnes indiquent le type et le nom de chaque objet ainsi que son "nouveau" nom, c'est-à-dire son nom par défaut dans la base de destination. Vous pourrez modifier ce nom si vous le souhaitez dans la page détaillée (cf.

paragraphe suivant).
L'option de la colonne "Copier" indique si l'objet sera copié ou non dans la base de destination. Cette option est active : vous pouvez la désélectionner pour résoudre rapidement les conflits de copie. A noter que s'il s'agit d'un objet dépendant, la référence (le nom) sera conservée dans la base de destination. Vous pouvez sélectionner / désélectionner alternativement toutes les cases de la liste en effectuant **Alt**+clic (Windows) ou **Option**+clic (Mac OS) sur une case. Vous pouvez également utiliser le menu contextuel de la boîte de dialogue.

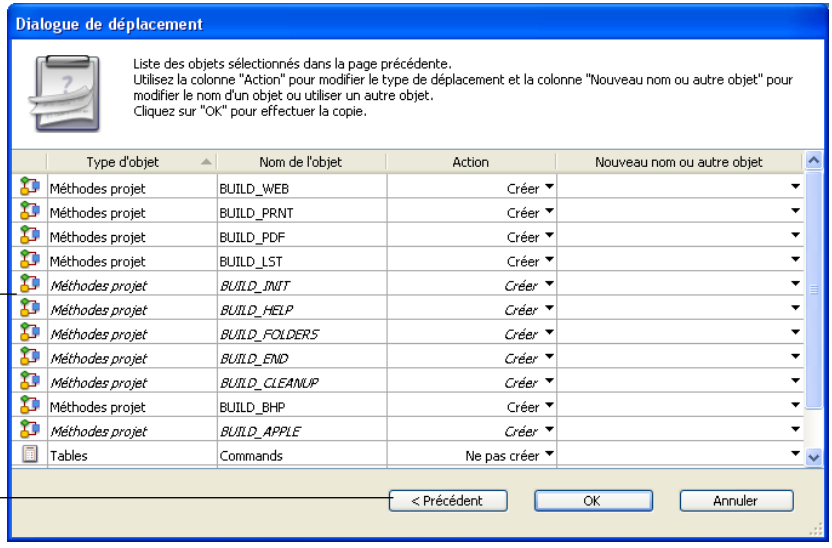
Si les paramétrages par défaut du déplacement vous conviennent, vous pouvez cliquer directement sur le bouton **OK** afin d'exécuter le déplacement.

Note Si au moins un objet a été associé à l'action **Utiliser un autre objet**, vous ne pouvez pas valider la copie sans avoir désigné l'objet cible de la base de destination (ou désélectionné l'objet).

Si vous souhaitez modifier certaines actions, cliquez sur le bouton **Suivant>** pour afficher la page détaillée. Si vous souhaitez annuler le déplacement, cliquez sur le bouton **Annuler**.

Page détaillée

La page détaillée liste les objets à copier (non désélectionnés dans la page principale) et permet de modifier les paramètres du déplacement :



Vous pouvez utiliser le pop up menu de la colonne "Action" pour modifier les actions effectuées sur les objets. Les actions proposées dans le menu dépendent du type d'objet sélectionné. Ces actions sont décrites ci-dessous.
Il est possible de modifier en une seule opération l'action assignée à plusieurs lignes. Pour cela, il suffit de sélectionner les lignes à modifier puis de choisir une nouvelle action dans la colonne "Action" de l'une des lignes sélectionnées. La modification est alors reportée dans toutes les lignes de la sélection où elle est applicable. Si l'action est incompatible avec une ligne, la ligne n'est pas modifiée et une boîte de dialogue d'alerte vous le signale.

La colonne "Nouveau nom ou autre objet" affiche le nom qui sera attribué à l'objet une fois copié dans la base de destination. Vous pouvez modifier ce nom (attention dans ce cas à ne pas utiliser un nom existant dans la base de destination, ce qui créerait un nouveau conflit de nom).
Dans le cas d'objets dépendants, cette colonne vous permet également de désigner un autre objet de la base de destination (lorsque l'action **Utiliser un autre objet** est sélectionnée). Par exemple, dans le cas du déplacement d'un formulaire table, vous pouvez désigner une table de la base de destination comme table d'appartenance, au lieu de créer la table.

Actions possibles

Les actions alternatives possibles sont les suivantes :

- **Ne pas créer** : l'objet n'est pas copié. Dans le cas d'un objet dépendant, sa référence (son nom) est conservée si un objet de même nom existe déjà dans la base (dans ce cas, il est utilisé par l'objet principal). Si aucun objet de même nom n'est disponible, la référence est supprimée.
- **Remplacer** : cette option est proposée lorsqu'un objet de même nom et de même type existe déjà dans la base. Dans ce cas, l'objet de la base de destination est remplacé par celui de la base de départ.
- **Créer** : l'objet dépendant est copié dans la base de destination avec ses propriétés (option proposée lorsqu'il n'y a pas de conflit de nom).
- **Créer et renommer** : cette option est proposée lorsqu'un objet de même nom et de même type existe déjà dans la base. Par défaut, l'objet est renommé par l'ajout d'un numéro. Dans ce cas, vous pouvez renommer l'objet dans la colonne "Nouveau nom ou autre objet". Bien entendu, les références de l'objet sont mises à jour dans la base de destination.

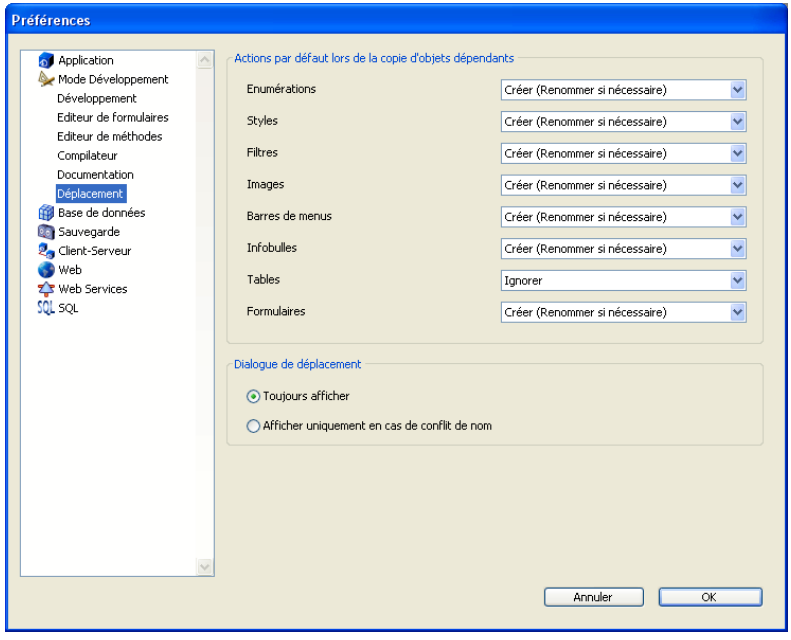
- **Utiliser un autre objet** : cette option n'est disponible qu'avec les objets dépendants. Elle permet d'utiliser en tant que référence un autre objet, déjà présent dans la base de destination. Dans ce cas, la colonne "Nouveau nom ou autre objet" contient la liste des autres objets que vous pouvez utiliser.
- **Utiliser la table de même nom** : cette option est proposée lorsqu'une table de même nom existe déjà dans la base. Dans ce cas, la colonne "Nouveau nom ou autre objet" contient la liste des tables que vous pouvez utiliser à la place de la table dépendante.

Si les objets dépendants référencent eux-mêmes d'autres objets, la liste est mise à jour en fonction de vos paramétrages.

Si les paramétrages du déplacement vous conviennent, vous pouvez cliquer sur le bouton **OK** afin d'exécuter le déplacement. Cliquez sur le bouton **<Précédent** pour retourner à la page principale. Si vous souhaitez annuler le déplacement, cliquez sur le bouton **Annuler**.

Préférences de déplacement

Vous pouvez pré-configurer les déplacements d'objets dans la base via la page Mode Développement/Déplacement des Préférences :



Ces préférences s'appliquent au déposer/coller d'objets dans la base, c'est-à-dire lorsque la base est utilisée comme base de destination.

Actions par défaut lors de la copie d'objets dépendants

La partie supérieure de la boîte de dialogue permet de configurer le déplacement des objets dépendants, c'est-à-dire les objets liés aux formulaires déplacés (cf. paragraphe "Objets dépendants des formulaires", page 86). Vous pouvez définir une action pour chaque type d'objet dépendant.

Ces actions par défaut seront automatiquement appliquées si le déplacement ne provoque pas de conflit et si l'option **Afficher uniquement en cas de conflit de nom** est sélectionnée (cf. paragraphe suivant). Dans le cas contraire, elles seront sélectionnées par défaut dans la boîte de dialogue de déplacement.

Les actions **Ignorer**, **Créer (renommer si nécessaire)**, **Créer (remplacer si nécessaire)** ainsi que **Utiliser un autre objet** sont proposées pour chaque type d'objet. D'autres actions contextuelles plus spécifiques sont proposées dans la boîte de dialogue de déplacement lorsqu'elle est affichée. Pour plus d'informations sur ces actions, reportez-vous au paragraphe "Actions possibles", page 89.

Voici la description de ces options :

- **Ignorer** : un objet dépendant de ce type n'est jamais copié dans la base de destination. Dans le dialogue de déplacement, l'action **Ne pas créer** est proposée par défaut.
- **Créer (renommer si nécessaire)** : un objet dépendant de ce type est toujours copié dans la base de destination. Dans le dialogue de déplacement, l'action **Créer** est proposée par défaut si l'objet n'existe pas déjà dans la base de destination.
  En cas de conflit de nom avec un objet de la base de destination, l'objet copié est renommé par ajout du suffixe "_n", suivant le principe appliqué pour les objets principaux (cf. paragraphe "Principes des déplacements", page 85). Dans ce cas, l'action **Renommer** est proposée par défaut dans le dialogue de déplacement.
- **Créer (remplacer si nécessaire)** : un objet dépendant de ce type est toujours copié dans la base de destination. Dans le dialogue de déplacement, l'action **Créer** est proposée par défaut si l'objet n'existe pas déjà dans la base de destination.
  En cas de conflit de nom avec un objet de la base de destination, l'objet copié remplace l'objet existant. Dans ce cas, l'action **Remplacer** est proposée par défaut dans le dialogue de déplacement.

- **Utiliser un autre objet** : cette option provoque l'affichage systématique du dialogue de déplacement, même si l'option "Afficher uniquement en cas de conflit de nom est sélectionnée". Lors du déplacement, vous devez désigner un objet de la base de destination à utiliser au lieu de l'objet dépendant copié.

Note Ces options sont prises en compte pour les objets dépendants uniquement. Pour les objets déplacés, l'action par défaut est du type **Créer (remplacer si nécessaire)**.

Affichage du dialogue de déplacement

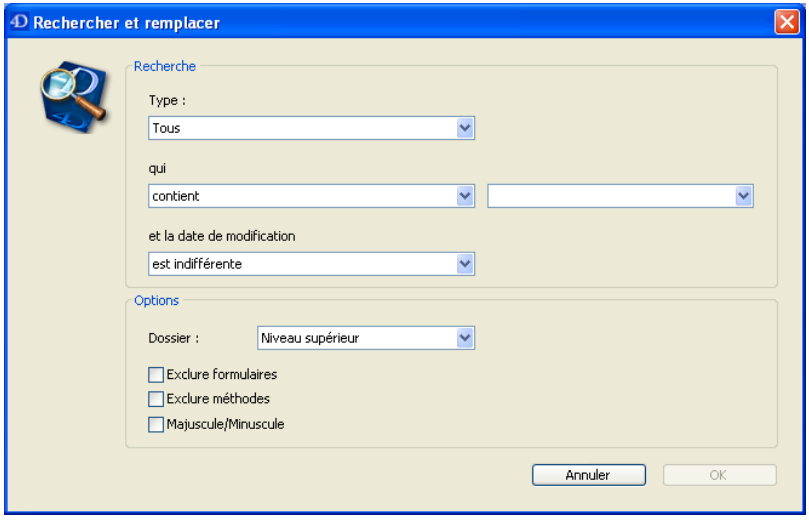
Cette option permet de configurer l'affichage du dialogue de déplacement.
Lorsque le bouton radio **Toujours afficher** est sélectionné, le dialogue apparaît à chaque déplacement, ce qui permet de contrôler précisément les objets déplacés. Si le bouton radio **Afficher uniquement en cas de conflit de nom** est sélectionné, le dialogue n'apparaît que lorsqu'un objet déplacé (principal ou objet dépendant) est en conflit de nom avec un objet de la base de destination.

Rechercher et remplacer dans la base

La fonction de recherche dans le développement a été étendue. D'une part, de nouveaux critères de recherche sont disponibles et, d'autre part, il est désormais possible de renommer ou de préfixer les éléments trouvés.

Nouveaux critères de recherche

Lorsque vous sélectionnez la commande **Chercher dans le développement** dans le menu **Edition** de 4D, la boîte de dialogue suivante apparaît :



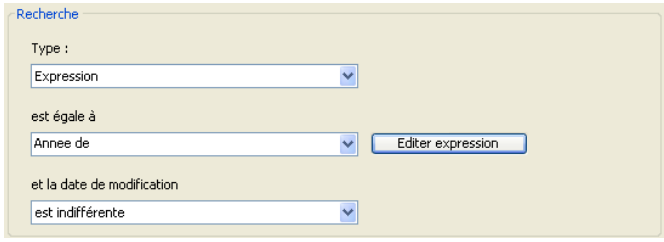
Un nouveau type d'objet et deux nouveaux critères de recherche vous sont proposés. Une recherche standard pourra donc être basée sur un, deux ou trois critères.

- **Type** : un nouveau type d'objet, **Méthode projet**, est disponible dans le pop up menu de sélection. Cette fonction vous permet d'effectuer des recherches parmi les noms des méthodes projet de la base.
Lorsque ce type est sélectionné, l'option "Exclure méthodes" est inactivée. En outre, dans ce cas la fenêtre de résultat comporte alors les listes "Cibles" et "Sources" (cf. paragraphe "Pages Cibles et Sources", page 97).

- **Mode de recherche** : ce nouveau menu permet de définir le mode de comparaison à utiliser ("contient", "est exactement", "commence par" ou "se termine par").
Si vous ne souhaitez pas utiliser le nom comme critère de recherche, choisissez l'option "est indifférent". Dans ce cas, la recherche sera effectuée uniquement sur le critère du type d'objet, de sa date de modification, ou sur une combinaison des deux.

Note A la différence des versions précédentes, l'arrobas (@) peut désormais être utilisé comme "joker", par exemple pour définir des recherches du type "commence par et se termine par" (ex : "m@base").

Si le type d'objet recherché est une expression, ce menu est remplacé par la zone de chaîne recherchée et le bouton **Editer expression...** apparaît à droite de la ligne :



- **Date de modification** : ce nouveau menu permet d'utiliser la date de modification comme critère de recherche. Ce critère est valide pour les recherches parmi les méthodes et les formulaires uniquement. Il peut être utilisé seul ou combiné à un ou aux deux autre(s) critère(s).
Si vous ne souhaitez pas utiliser la date comme critère de recherche, choisissez l'option "est indifférente". Dans ce cas, la recherche sera effectuée uniquement sur le critère du type d'objet, de son nom, ou sur la combinaison des deux.

Note L'option "Nom d'objet entier", présente dans les versions précédentes de 4D, est remplacée par le menu de mode de recherche.

Option Dossier

- **Dossier** : cette nouvelle option permet de restreindre la recherche à un niveau de dossier. Par défaut, la recherche a lieu dans tous les dossiers.

Portée des recherches

Les recherches effectuées dans le développement avec 4D v11 incluent davantage d'objets. Désormais, en fonction du type de chaîne de caractères désigné, la recherche sera effectuée parmi les éléments suivants :

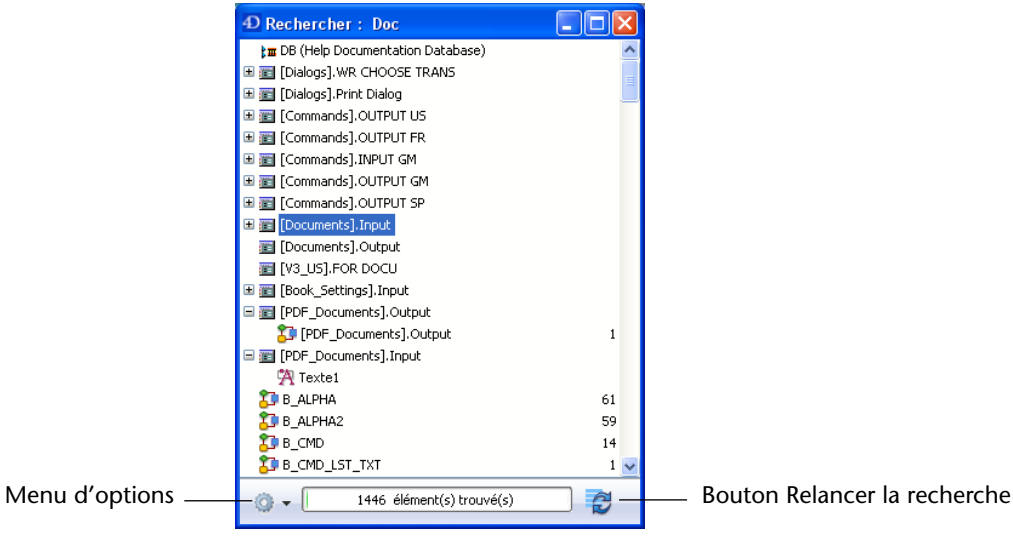
- formulaires (peuvent être exclus)
- méthodes de tout type
- menus et commandes de menus
- énumérations
- tables et champs
- commentaires
- images de la bibliothèque
- textes statiques

- messages d'aide
- formats/filtres (nouveauté version 11)
- feuilles de style (nouveauté version 11)
- commandes de plug-ins (nouveauté version 11)
- commandes (nouveauté version 11)
- constantes (nouveauté version 11)
- dossiers (nouveauté version 11)

Les résultats d'une recherche sont désormais regroupés par type d'objet, puis par ordre alphabétique dans la fenêtre de résultat.

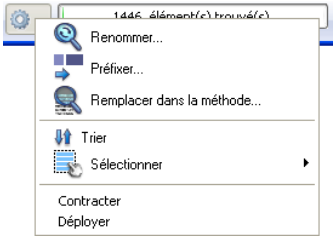
Nouvelle fenêtre de résultat

La fenêtre de résultat comporte désormais un menu d'options permettant d'effectuer diverses opérations sur les objets qu'elle contient :

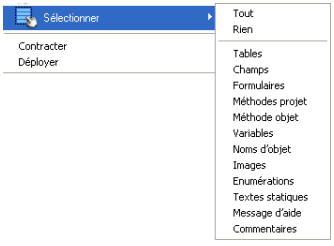


Les commandes du menu d'options sont activées ou inactivées en fonction de la sélection d'objets. Par exemple, si la sélection contient

des commandes 4D, les fonctions de nommage ou de préfixage sont inactivées :



- **Renommer** : cette commande permet de renommer l'objet sélectionné. Cette opération est détaillée dans le paragraphe suivant.
- **Préfixer** : cette commande permet de préfixer les noms de tous les objets sélectionnés dans la liste. Cette opération est détaillée dans le paragraphe suivant.
- **Remplacer dans la méthode** : ce bouton permet d'effectuer un rechercher-remplacer parmi une ou plusieurs méthodes. Cette opération est détaillée dans le paragraphe suivant.
- **Trier** : cette commande trie le contenu de la fenêtre par ordre alphabétique. Maintenez la touche **Maj** enfoncée pendant la sélection de cette commande pour effectuer un tri dans l'ordre alphabétique inverse.
- **Sélectionner >** : ce sous-menu permet d'effectuer des sélections automatiques parmi les objets présents dans la fenêtre (vous pouvez sélectionner manuellement un ou plusieurs objets à l'aide des combinaisons de touches standard **Maj+clic** ou **Ctrl/Commande+clic**). Il est nécessaire d'effectuer une sélection parmi les objets de la liste afin de pouvoir renommer ou préfixer la sélection.



Choisissez la commande **Tout** pour sélectionner le contenu de la fenêtre. Vous pouvez également utiliser le raccourci standard **Ctrl+A** (Windows) ou **Commande+A** (Mac OS) ou encore le menu contextuel de la liste.
La sélection peut également être effectuée sur la base du type d'objet. Pour cela, choisissez un type d'objet dans le sous-menu.

- **Contracter/Déployer** : ces commandes permettent respectivement de contracter ou de déployer tous les éléments hiérarchiques contenus dans la liste de résultat.

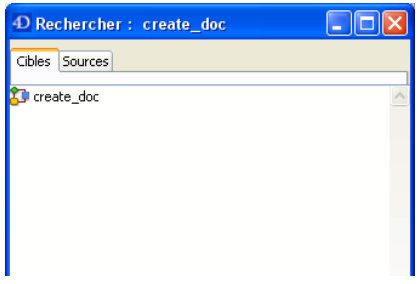
- **Relancer la recherche** : ce bouton permet de rééditer la recherche en conservant les mêmes critères et options. Cette fonction peut être utile par exemple pour vérifier que tous les remplacements souhaités ont été effectués.

  La fenêtre comporte également un menu contextuel permettant d'effectuer diverses opérations standard :

  Lorsque plus d'une occurrence d'un objet ou d'une chaîne a été trouvée à l'intérieur d'une méthode, la liste indique leur nombre en regard du nom de l'objet :

Pages Cibles et Sources

Lorsque vous avez effectué une recherche parmi les noms de méthodes projet uniquement (cf. paragraphe "Nouveaux critères de recherche", page 93), la fenêtre de résultat comporte deux listes, accessibles via des onglets : **Cibles** et **Sources** :



- La liste **Cibles** affiche les méthodes projet trouvées par la recherche, c'est-à-dire dont le nom et/ou la date de modification correspond aux critères de recherche.
- La liste **Sources** affiche les objets (méthodes, formulaires...) référençant les méthodes projet trouvées par la recherche.

Note Les objets appelants d'une méthode projet peuvent être recherchés directement depuis l'Explorateur ou l'éditeur de méthodes. Pour plus d'informations, reportez-vous au paragraphe "Rechercher les appelants", page 81.

Préfixer et renommer

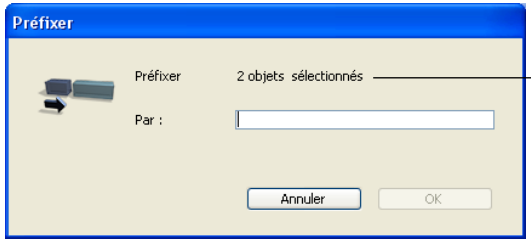
Dans 4D v11, la fenêtre de résultat d'une recherche permet de préfixer et de renommer une sélection d'objets, ou encore de remplacer une chaîne dans une sélection de méthodes.

Préfixer

Le préfixage s'avère très utile lorsque vous souhaitez améliorer la lisibilité de votre code ou avoir un moyen rapide de repérer les familles d'objets de votre base.

La commande de préfixage du menu d'option n'est active que si la sélection ne contient que des objets pouvant être préfixés. Les objets préfixables sont les **variables**, les **méthodes projet** et les **formulaires** (table et projet).

Lorsque vous sélectionnez cette commande, la boîte de dialogue suivante apparaît :



La zone de saisie vous permet d'indiquer la chaîne de caractères à utiliser pour préfixer les objets sélectionnés.

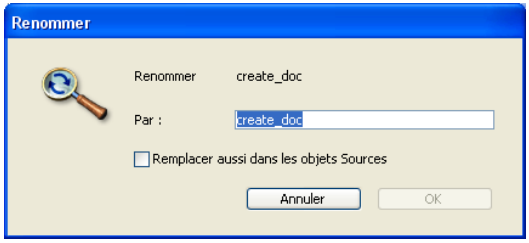
- **Remplacer aussi dans les objets Source** (option apparaissant lorsque vous effectuez un préfixage depuis la liste Cible) : cochez cette option si vous souhaitez que le préfixage soit également effectué dans les objets qui référencent les objets sélectionnés, par exemple les méthodes appelant la méthode à préfixer. Cette option permet de conserver le fonctionnement du code après préfixage.

Note Les noms d'objets étant limités à 31 caractères, il se peut que l'ajout d'un préfixe rende le nom d'un objet trop long. Dans ce cas, un message d'alerte vous en informe, vous laissant la possibilité d'annuler l'opération.

Renommer

Cette opération consiste à renommer un objet sélectionné. La commande **Renommer...** du menu d'options n'est active que si la sélection ne contient qu'**un seul objet** et que cet objet peut être renommé. Les objets renommables sont les **variables**, les **méthodes projet** et les **formulaires** (table et projet).

Lorsque vous sélectionnez cette commande, la boîte de dialogue suivante apparaît :



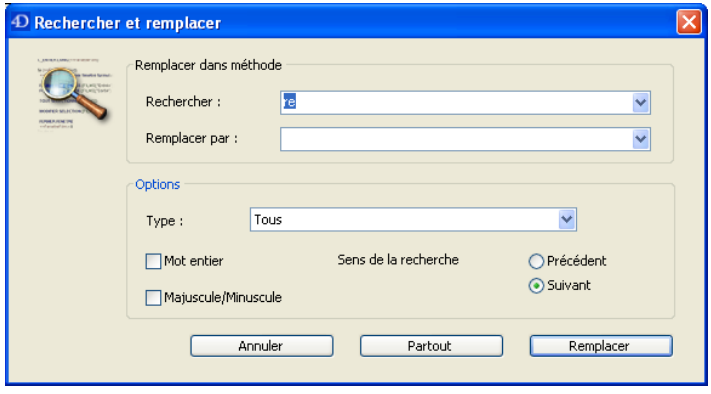
La zone de saisie vous permet d'indiquer le nouveau nom de l'objet sélectionné.

- **Remplacer aussi dans les objets Source** (option apparaissant lorsque vous effectuez un préfixage depuis la liste Cible) : cochez cette option si vous souhaitez que l'objet soit également renommé dans les objets qui le référencent. Cette option permet de préserver le fonctionnement du code après renommage.

Remplacer dans la méthode

Cette opération consiste à effectuer un "rechercher et remplacer" global parmi une ou plusieurs méthodes. La commande **Remplacer dans la méthode...** du menu d'options n'est active que si la sélection contient une ou plusieurs méthodes.

Lorsque vous sélectionnez cette commande, la boîte de dialogue standard de recherche et remplacement dans les méthodes apparaît :



Cette boîte de dialogue permet de rechercher et de remplacer une chaîne de caractères à l'intérieur des méthodes.

Note Dans 4D v11, cette boîte de dialogue comporte un menu d'option supplémentaire, **Type**. Pour plus d'informations, reportez-vous au paragraphe "Option de remplacement sélectif", page 138.

Gestion des raccourcis clavier

Dans 4D v11, les raccourcis clavier du mode Développement ont été modifiés de manière à être plus conformes aux usages et aux normes d'interface. Par exemple, le raccourci **Ctrl+P** (Windows) ou **Commande+P** (Mac OS), traditionnellement utilisé dans 4D pour ouvrir une méthode sélectionnée, est désormais associé à la commande **Imprimer**, à l'image de la plupart des applications.

Note **Ctrl+K** / **Commande+K** permet désormais d'ouvrir une méthode.

Vous pouvez toutefois personnaliser les raccourcis clavier par défaut afin de les adapter à vos habitudes de travail. Tous les raccourcis du mode Développement sont référencés dans un fichier xml nommé **4DShortcuts.xml**, situé dans le dossier **4D Extensions**. Pour définir des raccourcis personnalisés, il suffit de dupliquer ce fichier et de modifier la copie à l'aide d'un éditeur xml (veillez à ne pas affecter le même raccourci à plusieurs actions). Une fois les modifications effectuées, placez la copie dans le dossier de préférences courant de 4D. Le fichier **4DShortcuts.xml** présent dans le dossier de préférences sera alors chargé par 4D à la place du fichier par défaut.

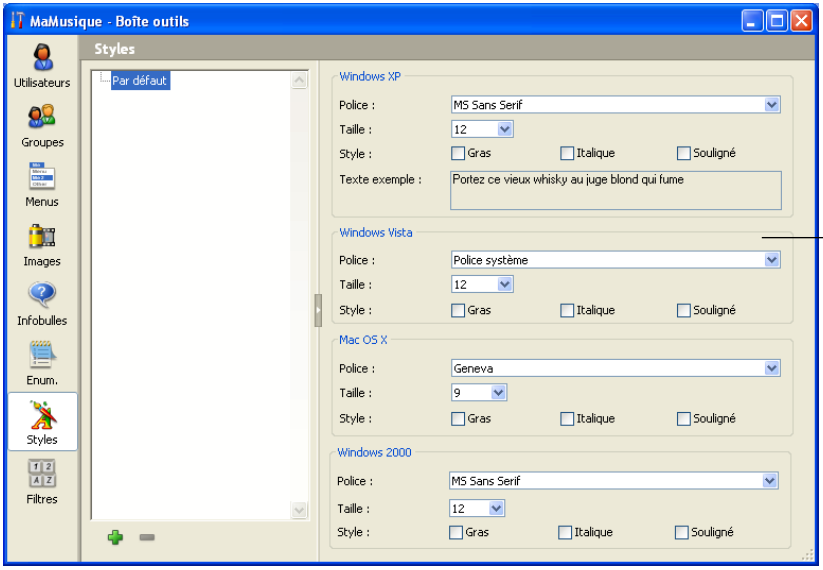
ATTENTION : *Ne modifiez pas directement le fichier 4DShortcuts.xml dans le dossier 4D Extensions.*

Feuilles de style

Prise en charge de Windows® Vista™

4D v11 prend en charge les spécificités d'interface de Windows® Vista™, la nouvelle version du système d'exploitation de Microsoft.

Pour cela, un nouvel ensemble d'attributs "Windows Vista" est disponible dans l'éditeur de feuilles de style :



Zone de définition de la feuille de style pour la plate-forme Windows Vista

Prise en charge du standard XLIFF

4D v11 prend désormais en charge le standard XLIFF pour l'adaptation linguistique (*localization*) des textes et libellés de l'interface. Cette technologie est utilisée en interne pour les applications 4D. Les développeurs 4D ou de plug-ins peuvent tirer parti de cette nouveauté dans leurs propres applications et plug-ins personnalisés.

Par compatibilité, le précédent système, basé sur les ressources, est toujours pris en charge dans 4D v11. A noter toutefois que l'accès direct aux ressources internes de 4D est désormais impossible (cf. paragraphe "Gestion des ressources", page 47).

Qu'est-ce que le XLIFF ?

XLIFF (*XML Localization Interchange File Format*) est un standard dédié aux processus de traduction et d'adaptation linguistique. Il permet d'établir une correspondance entre une langue source et une langue cible à l'intérieur d'un fichier XML.

Le standard XLIFF constitue de fait une alternative aux systèmes basés sur les ressources. Divers outils, dont de nombreux logiciels gratuits, permettent la gestion des fichiers XLIFF.

Pour plus d'informations sur le standard XLIFF, référez-vous à la spécification officielle XLIFF 1.1 qui peut être consultée à l'adresse suivante :

http://www.oasis-open.org/committees/xliff/documents/xliff-specification.htm

Attention : Le standard XLIFF tient compte de la casse des caractères.

Appeler des chaînes XLIFF depuis une base 4D

Le XLIFF peut en principe être utilisé partout où une référence de ressource pouvait être appelée dans les versions précédentes de 4D :

- Champ "Nom de l'objet" dans la liste des propriétés de l'éditeur de formulaires
- Champ "Nom de la fenêtre" dans la boîte de dialogue "Propriétés du formulaire" accessible depuis l'Explorateur
- Objets de texte statique, y compris les libellés des boutons, cases à cocher, etc.
- Editeur de menus
- Messages d'aide

Note Il n'est pas possible d'afficher de références XLIFF dans l'éditeur d'infobulles. Pour utiliser des infobulles basées sur le XLIFF, saisissez directement les références XLIFF dans le champ "Message d'aide" de la Liste des propriétés.

La syntaxe des chaînes basées sur les ressources est ":xxxx,yyy" où xxxx représente le numéro de la ressource STR# et yyy le numéro d'élément. Par exemple, ":10115,3" signifie que 4D doit utiliser le 3e élément de la ressource STR# numéro 10115.

Dans 4D v11, vous pouvez utiliser des références XLIFF dans les mêmes conditions. Trois syntaxes peuvent être employées :

- Syntaxe **:10115,3**
  Cette syntaxe est identique à celle que vous utilisiez pour appeler des ressources standard dans 4D version 2004 et précédentes.
  Lorsque cette syntaxe est utilisée avec des fichiers XLIFF, la première valeur (10115 dans cet exemple) désigne l'attribut **id** de l'élément **group**.
  La seconde valeur (3 dans cet exemple) désigne l'attribut **id** de l'élément **trans-unit**.
  Cette syntaxe préserve la compatibilité avec les systèmes existants et permet de passer d'un système basé sur les ressources à un système

XLIFF sans qu'il soit nécessaire de retoucher la base. En pratique, 4D recherchera dans un premier temps les valeurs correspondant à 10115,3 dans tous les fichiers XLIFF ouverts ; si cette valeur n'est pas trouvée, le programme poursuivra la recherche dans tous les fichiers de ressources ouverts. Avec ce principe, vous pouvez progressivement mettre en place un système basé sur le XLIFF dans une application sans devoir modifier les références dynamiques existantes : il vous suffit de copier un fichier XLIFF au bon emplacement (voir ci-dessous) et il sera pris en compte par 4D.
Vous pouvez ainsi conserver dans 4D v11 un ancien système basé sur les ressources tout en installant un nouveau système XLIFF. En fait, les deux systèmes peuvent être utilisés simultanément. Dans tous les cas, les fichiers XLIFF chargés seront prioritaires sur les fichiers de ressources si la même référence est présente aux deux emplacements.

- Syntaxe **:xliff:OKButton**
  Cette syntaxe alternative est utilisable uniquement avec les fichiers XLIFF chargés. Dans ce cas, la valeur référencée (*OKButton* dans l'exemple) désigne l'attribut **resname** de l'élément **trans-unit**.

- Syntaxe **Monlibellé** (c'est-à-dire, texte ou libellé statique "en clair")
  Cette syntaxe a pour but de faciliter la traduction des formulaires. A la différence des autres syntaxes, elle n'est utilisable que dans le cadre des formulaires. Le principe est d'encadrer dans le fichier XLIFF les éléments **trans-unit** par deux éléments **group** dont les attributs **resname** contiennent respectivement le nom de la table et du formulaire.

  - Exemple (*formulaire table*)
    Pour traduire le libellé de bouton "Sauvegarder" (champ "Titre" dans la liste des propriétés) dans le formulaire "Form1" de la table [Clients], il suffit que le fichier XLIFF contienne les lignes :

```
<group resname="[Clients]">
  <group resname="Form1">
    <trans-unit resname="Sauvegarder">
      ...
    </trans-unit>
  </group>
</group>
```

- Exemple (*formulaire projet*)
  Dans le cadre d'un formulaire projet, le nom de la table doit être remplacé par le libellé [ProjectForm], ce qui donne :

```
<group resname="[ProjectForm]">
  <group resname="Form1">
    <trans-unit resname="Sauvegarder">
      ...
    </trans-unit>
  </group>
</group>
```

Notes

• Si vous utilisez les différentes syntaxes dans votre base, l'ordre de priorité appliqué pour rechercher une traduction valide dans les fichiers XLIFF sera :
1. syntaxe ":10115,3"
2. syntaxe ":xliff:OKButton"
3. syntaxe "Monlibellé".

• Certaines commandes du thème "Ressources" ont été modifiées afin de pouvoir tirer parti des documents XLIFF. Pour plus d'informations, reportez-vous au paragraphe "Ressources", page 396.

Installer des fichiers XLIFF personnalisés

La mise en place d'une architecture XLIFF au sein d'une application personnalisée nécessite simplement la création d'un ou plusieurs fichier(s) XLIFF valide(s) et leur copie dans le dossier de la base (le *progiciel* sous Mac OS).

Sur les deux plates-formes (Mac OS et Windows), les fichiers XLIFF personnalisés doivent être stockés dans le dossier *Resources* de la base. Le chemin d'accès complet à utiliser est le suivant :

Windows:
***MaBase*\Resources*Lang*.lproj*MonLocEn*.xlf**

Mac OS:
***MaBase*:Resources:*Lang*.lproj:*MonLocEn*.xlf**

où :

- *MaBase* est le dossier contenant les fichiers de la base
- *Lang*.lproj est un dossier contenant les fichiers XLIFF pour le langage *Lang*. Le nom du dossier doit être conforme au standard international (reportez-vous au paragraphe ".lproj", page 105). Par exemple, pour les versions anglaises le dossier doit être nommé *en.lproj*.

4D chargera automatiquement les fichiers XLIFF du dossier de la langue courante de la base. Pour définir la langue courante de la base, 4D recherche successivement dans le dossier *Resources* de la base une langue correspondant à (dans l'ordre de priorité suivant) :
1. la langue du système (sous Mac OS, plusieurs langues peuvent être définies avec un ordre de préférence, 4D utilise ce pramétrage).
2. la langue de l'application 4D.
3. l'anglais
4. Si aucune de ces recherches n'aboutit, la première langue trouvée dans le dossier *Resources* est chargée.

La nouvelle commande Lire langue courante base permet de connaître la langue courante définie pour la base.

Si une variante de langue non disponible dans les fichiers XLIFF est utilisée, le langage le plus proche sera chargé.

Le nom des fichiers XLIFF est libre, ils doivent simplement comporter l'extension ".xlf". Vous pouvez placer plusieurs fichiers XLIFF dans le même dossier de langue, ils seront chargés dans l'ordre alphabétique.

Nom de dossier .lproj

Le nom du dossier .lproj doit respecter l'une des normes décrites ci-dessous. 4D recherchera un dossier valide sur la base de chacune de ces normes, dans l'ordre suivant :

1 RFC 3066 bis

Cette RFC est encore à l'état d'ébauche. 4D prend en charge la partie suivante de cette RFC : une langue est décrite par un code pays (ISO639-1) + un signe moins + code régional (ISO3166)
Par exemple, "fr-ca" (donc *fr-ca.lproj*) désigne la langue "français canadien".

2 ISO639-1

Cette norme définit chaque langue sur deux lettres. Par exemple, "fr" (donc *fr.lproj*) désigne le français.
Réf : http://www.loc.gov/standards/iso639-2/php/code_list.php

3 Nom "Legacy"

Dans cette convention, le nom de la langue est écrit en anglais et en toutes lettres. Par exemple, "french" (donc *french.lproj*) désigne le français.

Note Les deux premières normes sont prises en charge uniquement à partir de la version 10.4 de Mac OS. Avec les versions précédentes de cet OS, seul le nom “Legacy” peut être utilisé.

Un tableau récapitulatif des codes de langue pris en charge par 4D est fourni à l’annexe A, “Codes de langue”, page 431.

Si plusieurs définitions de langue existent, 4D utilise la traduction la plus précise. Par exemple, si le paramétrage de langue de l’OS est “français canadien”, 4D tente d’utiliser d’abord les traductions “fr-ca” puis, s’il ne les trouve pas, les traductions “fr”.

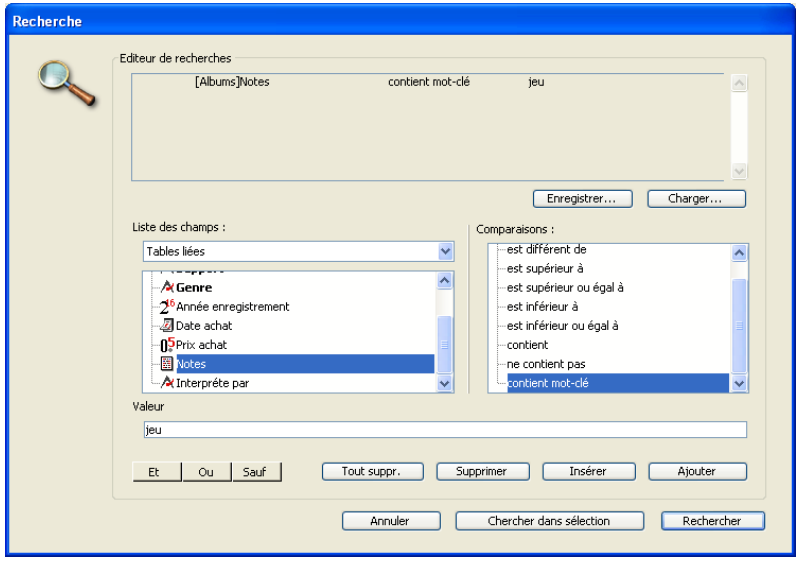
Note Le même principe s’applique à l’intérieur des fichiers XLIFF pour le marqueur “target-language”. Il faut faire attention à bien définir cet attribut au sein des fichiers XLIFF car un fichier situé dans le dossier “fr-ca.lproj” ayant un “target-language=fr” sera considéré comme une traduction en “fr” et non “fr-ca”.

Rechargement des fichiers XLIFF personnalisés

Les fichiers XLIFF sont rechargés dynamiquement, ce qui vous permet de contrôler visuellement l’intégration des libellés traduits dans l’interface de l’application. Le rechargement a lieu lorsque 4D passe au premier plan, si la date ou l’heure de modification a changé depuis la dernière sauvegarde. Le formulaire courant est rechargé simultanément.

Requêtes sur les enregistrements

Un nouvel opérateur de comparaison est disponible dans l'éditeur de recherches standard de 4D : "contient mot-clé" :



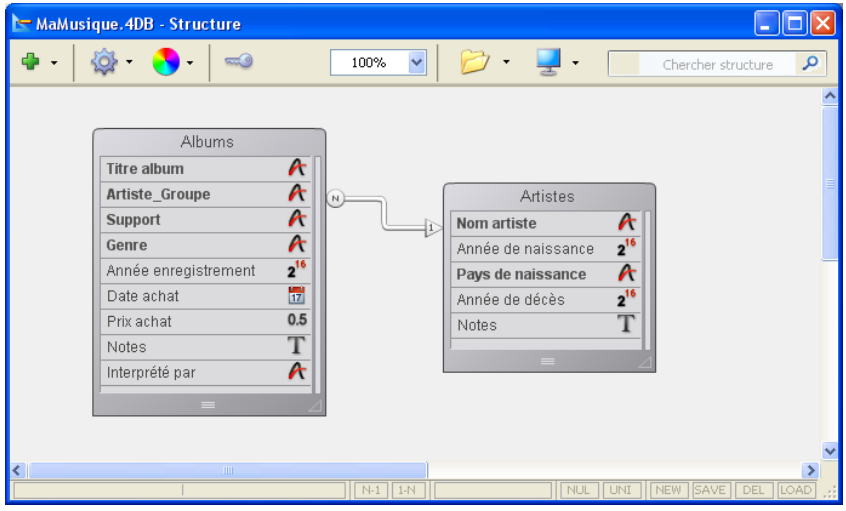
Cet opérateur permet d'effectuer des recherches par mots-clés avec des champs de type Alpha ou Texte. Puissante et rapide, la recherche par mot-clés permet de sélectionner les enregistrements dont le champ défini comme critère contient le mot-clé défini.

La recherche par mots-clés peut également être effectuée par programmation. Pour plus d'informations sur ce type de recherche, reportez-vous au paragraphe "Recherches par mots-clés", page 275.

5 Editeur de structure

De nombreuses nouveautés et améliorations ont été apportées à l'éditeur de structure dans 4D v11 SQL. Ces nouveautés concernent tous les aspects de l'éditeur : interface, fonctions, édition, etc.

Nouvel éditeur de structure de 4D v11



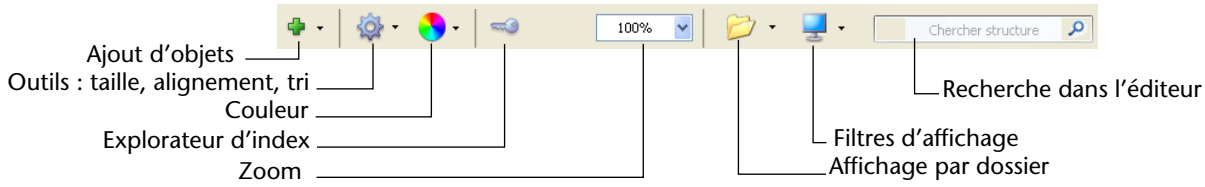
L'apparence générale de l'éditeur de structure a été modifiée afin de mieux l'intégrer à l'ensemble des éditeurs de 4D : tables, liens, Inspecteur ont été redessinés et leur ergonomie revue.

En outre, il est désormais possible d'exporter la définition de structure sous forme de fichier texte et de créer des structures à la volée sur la base d'un fichier de définition.

Barre d'outils, barre d'information et nouvel Inspecteur

L'éditeur dispose désormais d'une barre d'outils permettant l'accès à plusieurs fonctions d'ajout, de navigation et de visualisation :

Barre d'outils de l'éditeur de structure



Les fonctions de la barre d'outils sont décrites dans les paragraphes suivants.

La partie inférieure de la fenêtre de l'éditeur est une barre d'information affichant des données en fonction de la zone survolée par le curseur de la souris : table, champ ou lien.

Information sur une table

Nom et numéro de la table

Triggers de la table :
NEW = Sur sauvegarde nouvel enregistrement
SAVE = Sur sauvegarde enregistrement
DEL = Sur suppression enregistrement
LOAD = Sur chargement enregistrement

Information sur un champ

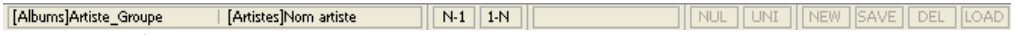
Nom et numéro de la table

Nom et numéro du champ

Type de champ

Attributs du champ :
NUL = Traduire les NULL en valeurs vide
UNI = Unique

Information sur un lien

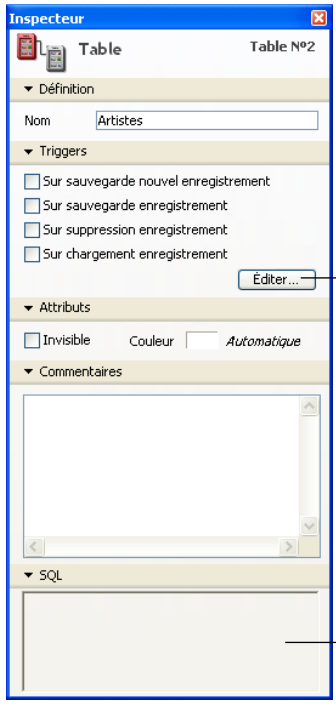


Champ de départ (champ N)

Champ d'arrivée (champ 1)

Attributs du lien :
N-1 = Lien aller automatique
1-N = Lien retour automatique

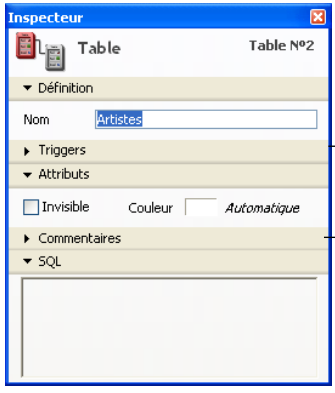
La palette Inspecteur a également été repensée de manière à afficher toutes les informations sur une seule page :



Ce nouveau bouton permet d'afficher directement la méthode trigger de la table

La nouvelle zone SQL est décrite dans le paragraphe "Zone SQL", page 121

Les diverses zones de la palette sont escamotables : vous pouvez cliquer sur les icônes en forme de triangle pour afficher ou masquer la zone correspondante :



Zones contractées

Les raccourcis clavier suivants peuvent être utilisés :

- **Maj+clic** sur le titre d'une zone contractée déploie la zone et contracte toutes les autres.
- **Alt** (Windows) ou **Option** (Mac OS) + **clic** sur le titre d'une zone contractée déploie toutes les zones.
- **Alt** (Windows) ou **Option** (Mac OS) + **clic** sur le titre d'une zone déployée contracte toutes les zones.

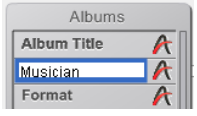
Lors de la fermeture de la fenêtre, 4D mémorise la position de la palette et le statut déployé/contracté des zones.

Note de compatibilité L'Inspecteur de table ne permet plus d'attribuer des groupes d'accès aux actions sur les tables (charger, enregistrer, ajouter et supprimer) ni de définir des propriétaires via l'Inspecteur de table. Pour plus d'informations, reportez-vous au paragraphe "Mécanismes non conservés", page 28.

Edition directe en mode graphique

De nombreuses fonctions (traditionnelles ou nouvelles) de l'éditeur de structure sont désormais directement accessibles en mode graphique :

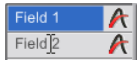
- **Modication des libellés** : vous pouvez modifier directement les libellés des tables et des champs depuis le mode graphique. Pour cela, il suffit de cliquer deux fois sur le libellé afin de le passer en édition :



A noter que, suivant la zone du libellé survolée par la souris, le curseur prend la forme d'une flèche (indiquant le mode "sélection", dans lequel vous tracez un lien par exemple) ou se transforme en **I** (indiquant le mode "édition").

Mode sélection

Mode édition

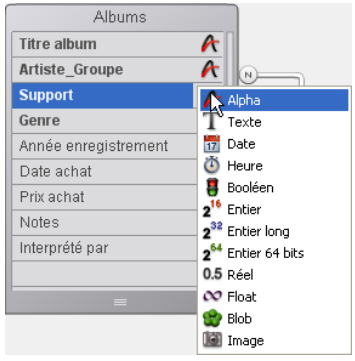


- **Ajout de champs** : vous pouvez ajouter des champs directement en mode graphique. Pour cela, **double-cliquez** dans la zone vide de la table ou appuyez sur la touche **Retour chariot** alors que la table est sélectionnée) : un nouveau champ Alpha est ajouté à la fin de la table et son libellé par défaut est saisissable.

Vous pouvez également créer un champ à l'aide du bouton d'ajout de la barre d'outils de l'éditeur.

Note Il est désormais possible de supprimer un champ d'une table en le sélectionnant puis en appuyant sur la touche **Ret. Arr** (ou la commande **Supprimer** du menu contextuel de l'éditeur). Pour plus d'informations sur ce point, reportez-vous au paragraphe "Suppression des tables et des champs", page 34.

- **Modification des types de champs** : vous pouvez modifier directement le type d'un champ via le pop up menu de type situé à droite du nom du champ :



- **Sélection simultanée de plusieurs objets de même type** : vous pouvez désormais sélectionner plusieurs tables, champs ou liens afin, par exemple, d'afficher ou de modifier leurs propriétés communes dans la palette de l'Inspecteur.
  La sélection de plusieurs objets s'effectue via les commandes standard :
  - **Maj+clic** pour sélectionner plusieurs objets contigüs.
  - **Ctrl+clic** (Windows) ou **Commande+clic** (Mac OS) pour sélectionner plusieurs objets non contigüs.

Note Vous pouvez sélectionner des champs de tables différentes.

  - Tracé d'un rectangle de sélection (sélection de tables uniquement).
  - Commande **Tout sélectionner** du menu **Edition** ou du menu contextuel (sélection de tables uniquement).

- **Tabulation** : vous pouvez appuyer sur **Tabulation** ou **Maj+Tabulation** pour sélectionner successivement chaque table. Les tables sont sélectionnées en fonction de leur position à l'écran et non leur ordre de création.

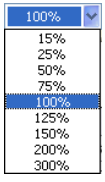
- **Copier et coller** : l'éditeur de structure vous permet de copier et de coller des tables et des champs.
  Pour copier un ou plusieurs objets, il suffit de les sélectionner et de choisir la commande "copier" standard (menu **Edition** de 4D, menu contextuel ou raccourci **Ctrl/Commande+c**).
  Pour coller une table, choisissez la commande **Coller** dans le menu **Edition** ou dans le menu contextuel de l'éditeur. La table est collée avec tous ses champs. Elle est renommée par défaut "Copie(N)_de_*NomTable*", où *NomTable* représente le nom de la table d'origine et *N* le nombre de copies de la table.
  Pour coller un champ, sélectionnez la table dans lequel vous souhaitez l'insérer et choisissez la commande "coller" standard (menu **Edition** de 4D, menu contextuel ou raccourci **Ctrl/Commande+v**). Le champ est collé avec toutes ses propriétés. Si un champ de même nom existait déjà dans la table, le champ collé est renommé par défaut "Copie(N)_de_*NomChamp*", où *NomChamp* représente le nom du champ d'origine et *N* le nombre de copies du champ.

Interface et visualisation

Afin d'apporter un plus grand confort de travail, de nouvelles fonctions de navigation et de visualisation des tables, des champs et des liens sont disponibles dans l'éditeur de structure.

Zoom

Vous pouvez désormais modifier l'échelle d'affichage de la structure de la base à l'aide de l'outil **Zoom** dans la barre d'outils de l'éditeur de structure :



100% est la valeur de zoom par défaut à l'ouverture de la base. Si une sélection existe dans la fenêtre, le zoom est centré sur cette sélection.

Le paramétrage courant du zoom est spécifique à chaque utilisateur. Il est mémorisé à la fermeture de la fenêtre.

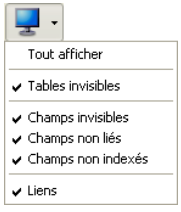
Défilement

De nouvelles fonctions facilitent la navigation parmi les structures de grande taille :

- La molette de la souris permet de faire défiler verticalement le contenu de la fenêtre de l'éditeur. Cette action provoque le défilement des champs d'une table lorsque le curseur est placé au-dessus de la table.
- La combinaison **Maj**+molette de la souris permet de faire défiler horizontalement le contenu de la fenêtre.
- Appuyer sur la touche **Maj** active l'outil "main" et permet de faire glisser le contenu de la fenêtre en cliquant dans des zones vides.

Filtrage des objets par type

Le nouveau bouton **Affichage** de la barre d'outils de l'éditeur de structure est associé à un menu vous permettant de filtrer les objets à afficher dans la fenêtre en fonction de leur type :



Par défaut, tous les objets sont affichés.

Cette fonction autorise différents types de représentations ou vues — de la plus simple à la plus complexe — et constitue un véritable outil d'analyse de la structure en fonction du niveau d'information sélectionné.

Le paramétrage d'affichage est appliqué à l'ensemble des tables et des champs. Ce paramétrage est spécifique à chaque utilisateur. Il est mémorisé lors de la fermeture de la fenêtre.

Le fait de masquer certains types d'objet ne modifie pas la position des tables dans la fenêtre.

- Lorsque plusieurs types de champs sont cochés, l'opérateur logique OU est utilisé pour déterminer les objets à afficher. Par exemple, si les options **Champs invisibles** et **Champs non indexés** sont cochées, tous les champs non indexés (visibles ou non) et tous les champs invisibles (indexés ou non) sont affichés.
- Les tables sont prioritaires sur les champs : si une table n'est pas affichée, les champs qu'elle contient ne sont pas affichés.

- Si vous ajoutez un objet dont le type est masqué (table, champ ou lien), l'objet apparaît dans l'éditeur, vous devez sélectionner de nouveau l'option correspondante dans le menu du bouton **Affichage** afin de le masquer.

Filtrage des tables par dossier

Vous pouvez masquer des ensembles de tables dans l'éditeur de structure sur la base des dossiers définis dans la fenêtre de l'Explorateur.

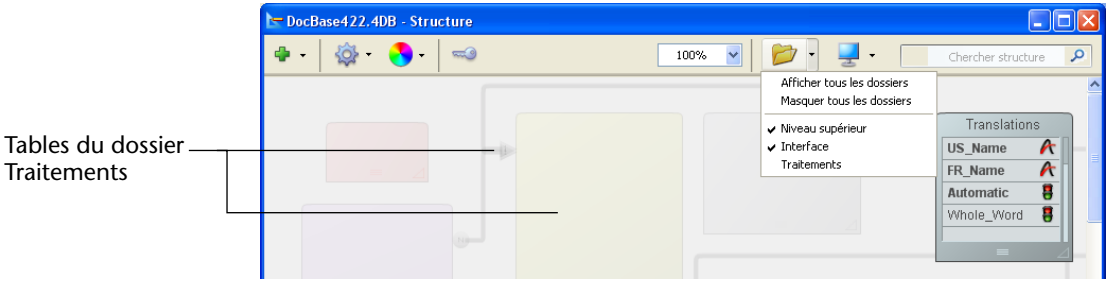
Ce paramétrage est effectué via le nouveau bouton **Dossiers** de la barre d'outils de l'éditeur de structure :

Le menu associé au bouton affiche des commandes de gestion de l'affichage ainsi que la liste des dossiers définis dans la base. Une coche apparaît en regard de chaque dossier affiché. Vous pouvez modifier l'affichage courant en sélectionnant ou en désélectionnant un dossier dans ce menu.

Les commandes **Afficher tous les dossiers/Masquer tous les dossiers** permettent d'afficher/de masquer toutes les tables des dossiers de la base.

Chaque clic sur le bouton  permet d'inverser l'affichage des tables : les tables affichées sont masquées et inversement.

Lorsque des tables sont masquées par cette commande, par défaut seul leur contour apparaît dans la fenêtre de structure. Seules les tables non masquées sont entièrement visibles :



Affichage des tables

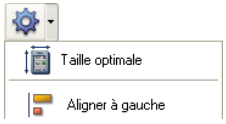
Une nouvelle option vous permet de définir l'apparence des tables masquées (sélection par dossier) dans l'éditeur de structure : **atténuées** ou **invisibles**. Cette option est accessible dans la page Mode Développement/Développement des Préférences.

Note Cette préférence nécessite la fermeture puis la réouverture de la fenêtre de structure pour être prise en compte.

Redimensionnement des tables

Deux nouvelles fonctions vous permettent de modifier automatiquement la taille de la ou des table(s) sélectionnée(s) :

- **Taille optimale** : cette fonction redimensionne la ou les tables sélectionnées de manière à ce que leur taille corresponde exactement à celle des champs qu'elles contiennent (sans ligne vide).
  Pour appliquer la commande **Taille optimale** à une ou plusieurs tables, vous pouvez :
  - Sélectionner la commande **Taille optimale** dans le menu du bouton d'outils de l'éditeur (cette commande est désactivée si aucune table n'est sélectionnée).

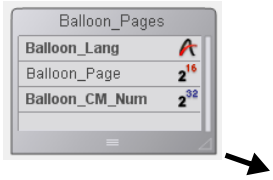
    

  - Sélectionner la commande **Taille optimale** dans le menu contextuel de l'éditeur (clic dans la zone de titre d'une table).
  - Utiliser le raccourci **Maj**+double-clic dans la zone de titre de la table à redimensionner. Si cette combinaison est répétée, la séquence suivante est appliquée : taille d'origine -> taille optimale - > contraction.
- **Contraction** : cette fonction redimensionne la ou les tables sélectionnées de manière à ce que seule leur zone de nom soit visible. Cette fonction est particulièrement utile avec les structures de grande taille.
  Pour appliquer la commande **Contraction** à une ou plusieurs tables, vous pouvez :
  - Sélectionner la commande **Contracter** dans le menu contextuel de l'éditeur (clic dans la zone de titre d'une table).
  - Utiliser le raccourci **Maj**+double-clic dans la zone de titre de la table à redimensionner. Si cette combinaison est répétée, la séquence suivante est appliquée : taille d'origine -> taille optimale - > contraction.

Notes • Un double-clic standard dans la zone de titre d'une table ouvre l'Inspecteur.
• **Alt** (Windows) ou **Option** (Mac OS) + double-clic ouvre le trigger de la table dans l'éditeur de méthodes.

• **Ctrl** (Windows) ou **Command** (Mac OS) + double-clic ouvre la page Formulaires de l'Explorateur.

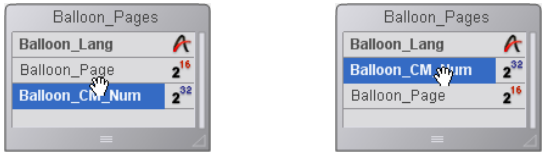
A noter qu'il est désormais possible de modifier manuellement la largeur des tables dans l'éditeur de structure, en cliquant sur leur angle inférieur droit :



Ordre des champs

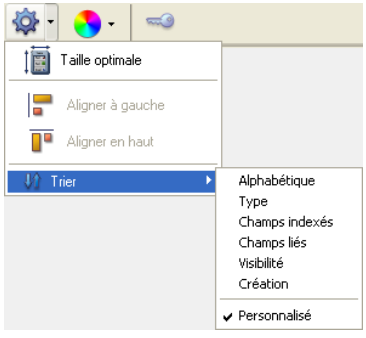
4D v11 vous permet de modifier l'ordre dans lequel apparaissent les champs dans les tables. Pour cela, vous disposez de deux possibilités :

- réorganiser manuellement le contenu d'une table par simple glisser-déposer de ses champs. Pour cela, il suffit d'appuyer sur la touche **Alt** (Windows) ou **Option** (Mac OS) — le curseur se transforme en main — et de déplacer chaque champ où vous le souhaitez :



Le tri personnalisé effectué de cette manière est mémorisé et pourra être appliqué de nouveau si nécessaire via la commande **Personnalisé** du menu de tri (cf. ci-dessous).

- appliquer un critère de tri. Vous disposez de plusieurs critères de tris accessibles via le sous-menu **Trier** du bouton d'outils de l'éditeur :



Lorsque vous choisissez une option, le critère est appliqué à toutes les tables sélectionnées dans la fenêtre. Ce menu est inactivé lorsqu'aucune table n'est sélectionnée.

Le menu indique également via une coche le critère actuellement appliqué. Plusieurs coches sont affichées si différents critères sont utilisés dans la sélection de tables.

Les critères de tri proposés sont les suivants :

- **Alphabétique** : les champs sont triés par ordre alphabétique croissant.
- **Type** : les champs sont triés en fonction du nom des types par ordre alphabétique croissant.
- **Champs indexés** : les champs indexés sont affichés en premier.
- **Champs liés** : les champs 1 puis les champs N sont affichés en premier.
- **Visibilité** : les champs visibles sont affichés en premier.
- **Création** : les champs sont affichés en fonction de leur ordre de création (tri par défaut).
- **Personnalisé** : aucun critère de tri automatique n'est appliqué. Cette option restitue le précédent tri effectué manuellement par glisser-déposer. Si aucun glisser-déposer n'a été effectué auparavant, cette option ne fait rien.

Notes • Lorsque vous ajoutez un champ dans une table, il est toujours placé après le dernier champ existant, quel que soit le critère de tri courant.
• Le tri des tables effectué dans l'éditeur de structure n'a pas d'influence sur l'affichage des champs dans les autres éditeurs de l'application.

Couleur

Vous pouvez définir une couleur personnalisée pour les tables, les champs et les liens. Pour définir la couleur d'un ou de plusieurs objets, effectuez votre sélection et choisissez une couleur depuis :

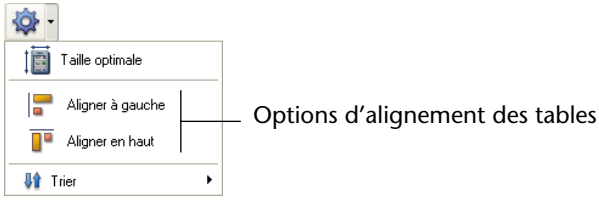
- le bouton **Couleur**  de la barre d'outils de l'éditeur,
- la commande **Couleur** du menu contextuel de l'éditeur (clic sur une table, un champ ou un lien),
- l'option **Couleur** de la palette de l'Inspecteur.

Note Les options **Libellés en couleur** et **Fond en couleur** proposées dans les Préférences des versions précédentes de 4D ne sont plus disponibles

dans 4D v11. La couleur de fond est désormais la couleur de la table et la couleur de libellé est celle du champ.

Alignement des tables

Deux options du menu d'outils permettent d'aligner verticalement et horizontalement les tables sélectionnées dans la fenêtre de l'éditeur :



Ces commandes ne sont actives que si au moins deux tables sont sélectionnées.

Qualité graphique de la structure

Cette option de la page Mode Développement/Développement des Préférences permet de faire varier le niveau de détail graphique de l'éditeur de structure.

Par défaut, la qualité est **Haute**. Vous pouvez sélectionner la qualité **Standard** afin de privilégier la rapidité d'affichage. L'effet de ce paramétrage est principalement perceptible lors de l'utilisation de la fonction de zoom.

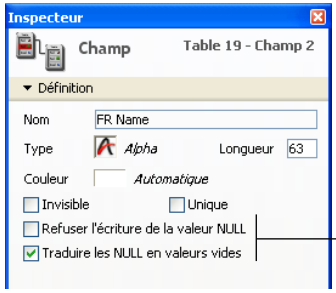
Prise en charge des attributs SQL

4D v11 inclut un puissant moteur SQL intégré (cf. chapitre "Utiliser le moteur SQL de 4D", page 205).

Le langage SQL permettant notamment la création de tables, de champs et de liens, des modifications ont été apportées à l'Inspecteur afin de prendre en charge les spécificités de ce nouveau langage.

Options de prise en charge des NULL

L'Inspecteur des champs contient deux nouvelles options destinées à la gestion de la valeur NULL dans les champs 4D :



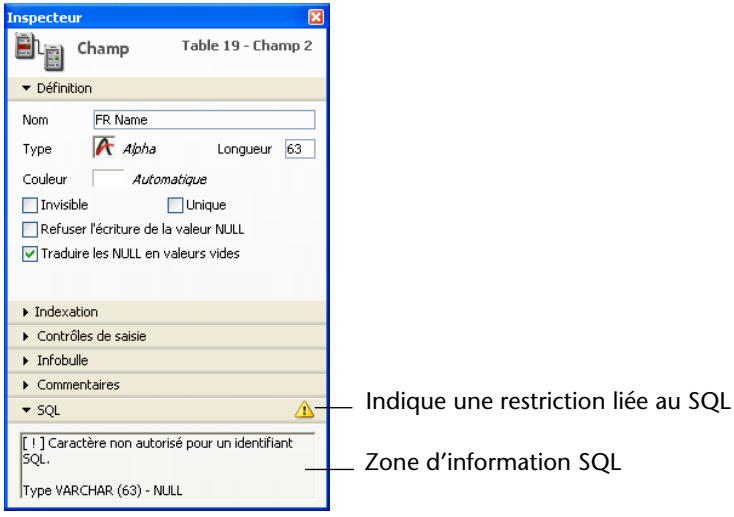
Note La valeur NULL est un concept issu du SQL. Pour plus d'informations sur ce point, reportez-vous au paragraphe "Valeurs NULL dans 4D", page 216.

- **Refuser l'écriture de la valeur NULL** : Cette option indique si le champ accepte ou non des valeurs NULL. Elle correspond à l'attribut NOT NULL du langage SQL. Si l'option est cochée, 4D n'autorisera pas de valeur NULL dans ce champ.
- **Traduire les NULL en valeurs vides** : Cette option de compatibilité permet de "forcer" 4D à considérer les éventuelles valeurs NULL présentes dans le champ comme des valeurs vides 4D standard. Ce principe permet de s'assurer que les recherches et traitements effectués par le moteur de 4D ne seront jamais faussés par la présence de valeurs NULL dans le champ.
  Ainsi, lorsque cette option est cochée, des recherches du type MonChampAlpha="" trouveront les enregistrements dont la valeur stockée dans le champ *MonChampAlpha* est NULL.

Zone SQL

La palette de l'Inspecteur comporte une nouvelle zone nommée **SQL** pour les tables, les champs et les liens. Cette zone fournit diverses

informations sur ces objets dans la perspective de leur exploitation via le langage SQL :



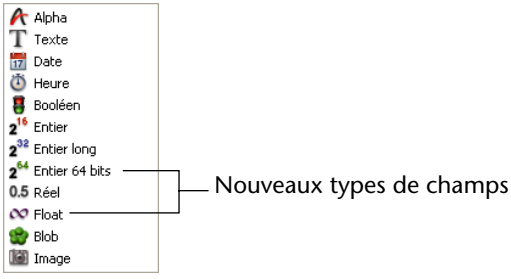
Cette zone signale notamment si le nom de la table, du champ ou du lien ne respecte pas les règles de la nomenclature SQL (par exemple, le SQL n'autorise pas que le nom d'un champ contienne des espaces, à la différence de 4D).

Pour les champs, la zone d'information SQL indique en outre les attributs SQL (type et propriétés).

Pour les liens, la zone indique les propriétés FOREIGN KEY et les REFERENCES.

Types Entier 64 bits et Float

La liste des types de champs comporte deux nouveaux types : **Entier 64 bits** et **Float** :

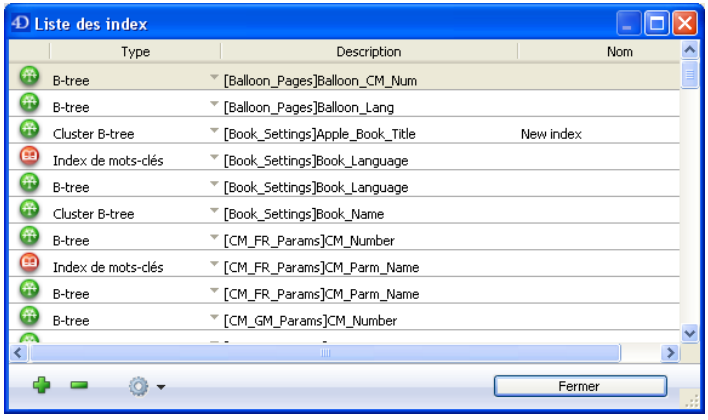


Ces types de champs sont liés à l'utilisation du moteur SQL de la base. Dans ce contexte, ils permettent de stocker des entiers 64 bits et des

nombre à virgule flottante (cf. paragraphe "Nouveaux types de champs", page 34). A noter qu'en cas de manipulation de ces champs avec le moteur de 4D, les données qu'ils contiennent sont converties en interne en réels.

Explorateur d'index

Le bouton  de la barre d'outils de l'éditeur de Structure affiche la fenêtre de l'Explorateur d'index. Cet Explorateur affiche la liste et les propriétés de tous les index de la structure :



Note Pour plus d'informations sur les types d'index, reportez-vous au paragraphe "Gestion des index", page 39.

L'Explorateur d'index permet de visualiser les principales propriétés des index :

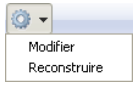
- **Type** : Type d'index. Chaque type d'index (B-tree, Cluster B-Tree, mots-clés) est représenté avec une icône différente.
  Il est possible de modifier le type d'index depuis l'Explorateur d'index en cliquant sur le triangle inversé et en sélectionnant une valeur dans le pop up menu.
- **Description** : Table et champ(s) de l'index. Dans le cas d'un index composite, cette liste contient tous les champs de l'index.
- **Nom** : Nom de l'index. Cette propriété est notamment exploitée par les nouvelles commandes du langage (cf. paragraphe "Définition structure", page 399). Les index des bases de données converties n'ont pas de nom par défaut.

Vous pouvez modifier ou ajouter un nom d'index en double-cliquant dans cette zone.

Le bouton affiche la boîte de dialogue de propriété d'index (cf. paragraphe suivant).

Le bouton permet de supprimer l'index sélectionné (une boîte de dialogue de confirmation apparaît).

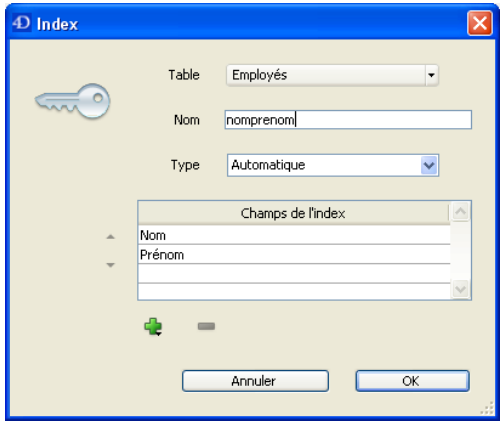
Deux commandes supplémentaires sont accessibles via le menu associé au bouton d'outils (disponible quand un index est sélectionné) :



- **Modifier** : affiche les propriétés de l'index sélectionné. Cette commande a le même effet qu'un double-clic sur une ligne de la liste (hormis dans la zone de nom).
- **Reconstruire** : permet de supprimer et de reconstruire l'index sélectionné. Une boîte de dialogue de confirmation apparaît lorsque vous sélectionnez cette commande.

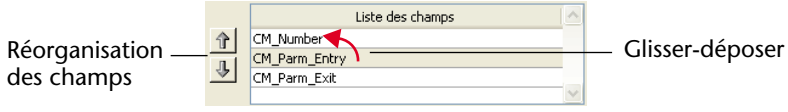
Boîte de dialogue de propriété d'index

Lorsque vous ajoutez un index via l'Explorateur d'index, le menu d'ajout d'objets ou le menu contextuel de l'éditeur de structure, la boîte de dialogue d'ajout d'index apparaît :



Cette boîte de dialogue permet de configurer la création de l'index. Elle est composée des éléments suivants :

- **Table** : liste toutes les tables de la base. Choisissez dans ce menu la table à laquelle appartiendra l'index.
- **Nom** : zone de saisie du nom de l'index.
- **Type** : menu de sélection du type d'index à créer. Si vous conservez l'option "Automatique", 4D choisira automatiquement le type d'index en fonction du contenu du champ.
- **Champ(s) de l'index** : cette zone permet de définir le ou les champ(s) associé(s) à l'index. Elle peut contenir par défaut un champ en fonction de la sélection courante dans l'éditeur.
  Pour ajouter un champ dans l'index, cliquez sur le bouton . La liste des champs de la table sélectionnée s'affiche, permettant de désigner le champ à ajouter à l'index.
  - Si vous voulez créer un index composite, ajoutez successivement chaque champ à inclure dans l'index. Une fois la liste établie, vous pouvez réordonner les champs via les boutons fléchés ou en utilisant le glisser-déposer.



  - Si vous avez choisi le type "Index de mots-clés", seuls les champs de type Alpha ou Texte peuvent être sélectionnés. Dans ce cas également, vous ne pouvez inclure qu'un seul champ dans l'index.

Pour supprimer un champ de l'index, sélectionnez-le et cliquez sur le bouton .

Une fois l'index configuré, cliquez sur le bouton **OK** pour le générer.

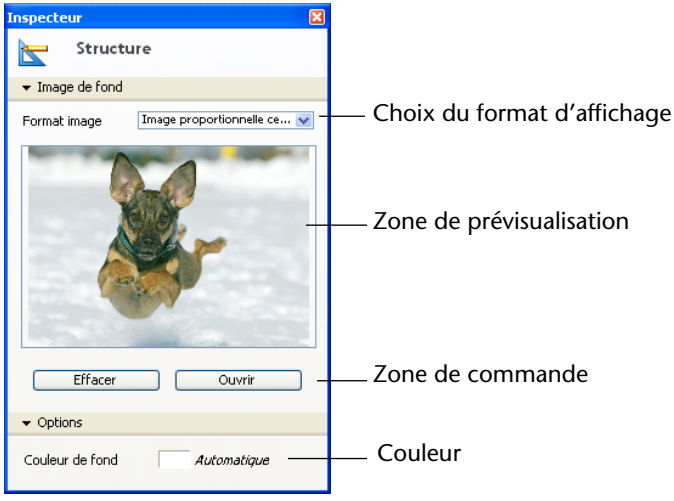
Personnaliser la fenêtre de l'éditeur

La fenêtre de l'éditeur de structure dispose de propriétés spécifiques qu'il est possible de configurer via l'Inspecteur.

Pour afficher les propriétés de la fenêtre de l'éditeur de structure, vous pouvez soit :

- double-cliquer (ou cliquer, si la palette est déjà affichée) dans un endroit vide de la fenêtre,
- cliquer avec le bouton droit de la souris dans un endroit vide de la fenêtre et choisir la commande **Propriétés de la structure...** dans le menu contextuel.

La palette de l'Inspecteur affiche alors les propriétés de la structure :



Les propriétés suivantes peuvent être définies :

- **Image de fond** : vous pouvez modifier l'image utilisée en fond de la fenêtre ainsi que son format d'affichage.
  - Pour modifier l'image, cliquez sur le bouton **Ouvrir...** ou cliquez avec le **bouton droit** de la souris dans la zone de prévisualisation et choisssez **Ouvrir...** dans le menu contextuel, puis sélectionnez le fichier contenant l'image à afficher. Vous pouvez utiliser tout format d'image.
    L'image sélectionnée s'affiche immédiatement dans la zone de prévisualisation et dans la fenêtre de l'éditeur.

- Pour modifier le format d'affichage de l'image, choisissez une valeur dans le menu **Format image**. Les formats proposés sont les formats d'affichage d'images standard de 4D.
- Pour supprimer une image personnalisée, cliquez sur le bouton **Effacer** ou cliquez avec le **bouton droit** de la souris dans la zone de prévisualisation et choisssez **Effacer** dans le menu contextuel.

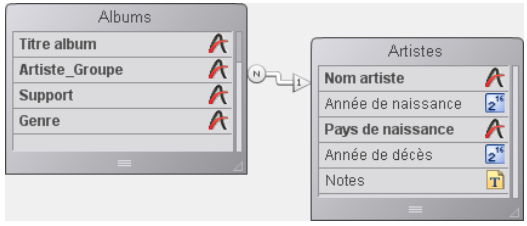
- **Couleur de fond** : vous pouvez modifier la couleur utilisée pour le fond de la fenêtre de l'éditeur. Pour cela, cliquez dans la zone de sélection de couleur et choisissez une couleur dans le menu de sélection.

Liens

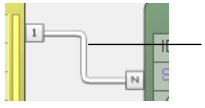
L'apparence et les propriétés des liens ont été modifiées dans 4D v11.

Apparence

L'apparence des liens a été modifiée, ils comportent notamment les lettres N et 1 :

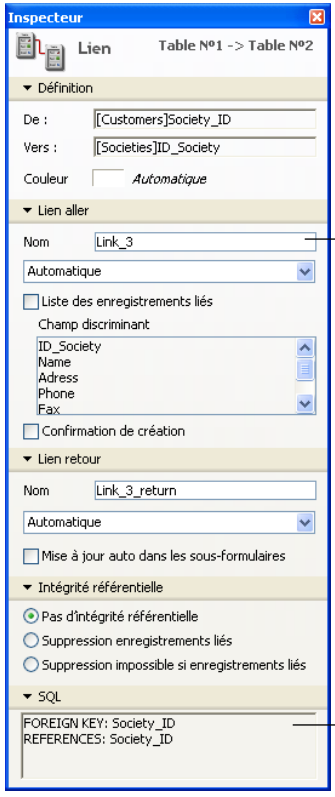


L'apparence des liens est différente lorsqu'ils ont été générés via le SQL (les connecteurs sont de forme carrée) ::



Propriétés des liens

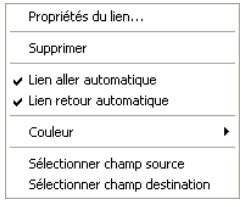
Lorsqu'un lien est sélectionné, l'Inspecteur affiche ses propriétés :



Les deux composantes d'un lien (lien aller et lien retour) peuvent désormais être nommées. Les prochaines versions de 4D v11 permettront d'exploiter cette nouvelle propriété.

Menu contextuel

Le menu contextuel associé à chaque lien permet d'accéder directement aux principales fonctions d'édition des liens et d'activer ou de désactiver la propriété "automatique" :



Ajouter un lien depuis la barre d'outils

Le bouton **[+]** de la barre d'outils de l'éditeur de structure permet d'ajouter différents types d'objets (table, champ, index ou lien) :

La commande **Lien** permet, si deux champs sont sélectionnés et si leur type le permet, de créer directement un lien entre eux. Le premier champ sélectionné est considéré comme le champ clé d'appel (champ N) et le second champ est considéré comme le champ clé primaire (champ 1).
Le lien est dessiné et l'Inspecteur affiche les propriétés du lien.
Si plus de deux champs sont sélectionnés, la commande ne fait rien.

Supprimer un lien

Dans 4D v11, il n'est plus possible de supprimer un lien en le retraçant depuis son origine pour le lâcher "dans le vide". Désormais, pour supprimer un lien, vous disposez de deux possibilités :

- sélectionner le lien et appuyer sur la touche **Suppr** ou **Ret. Arr.**,
- choisir la commande **Supprimer** dans le menu contextuel associé au lien.

Dans les deux cas, une boîte de dialoge d'alerte apparaît afin de vous permettre de confirmer ou d'annuler cette action.

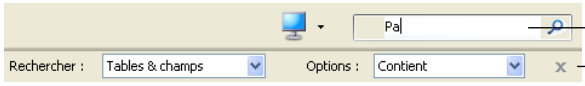
Chercher dans la structure

4D v11 vous permet d'effectuer des recherches dans la fenêtre de l'éditeur de structure. La recherche peut être effectuée parmi les éléments suivants :

- noms de tables et/ou nom de champs
- numéros de tables

Pour effectuer une recherche, saisissez la chaîne de caractères ou le numéro de table à rechercher dans la zone de de recherche de la barre d'outils, affichant par défaut "Chercher structure". La saisie d'une valeur dans cette zone provoque l'affichage d'une barre d'options sous

la zone de recherche, permettant de préciser la portée et la nature de la recherche :

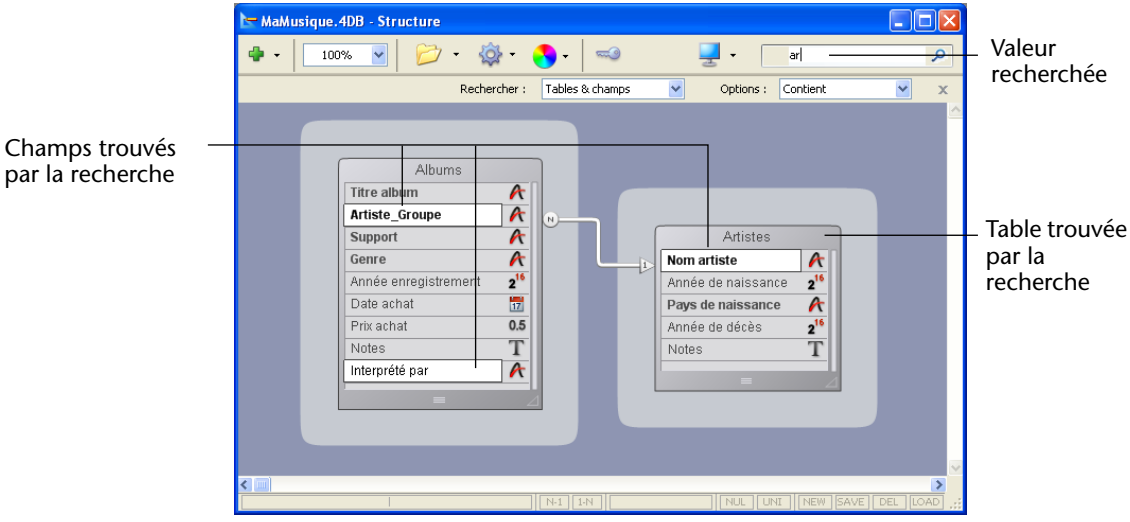


- Le menu Rechercher permet de définir la portée de la recherche (tables et champs ou tables uniquement) :

- Le menu Options permet de définir le type de recherche :

  - *Contient* (par défaut) : la recherche de “le” trouvera “Table”, “Lettre”, “Eléments”, etc.
  - *Commence* : la recherche de “pa” trouvera “Pays”, “paye” mais pas “Repas”.
  - *Numéro* : la recherche de “2” trouvera les tables numéro 2, 12, 20, 21, etc.

La recherche est effectuée en temps réel, à mesure que vous saisissez des valeurs. Les tables et/ou les champs trouvé(e)s par la recherche sont “éclairé(e)s”



Si la recherche n'aboutit pas, la zone de recherche est colorée de rouge :

Pour sortir du mode "recherche", cliquez sur le bouton x dans la barre d'options de recherche ou supprimez tous les caractères de la zone de recherche.

Exporter et importer des définitions de structures

4D v11 permet d'exporter la définition de la structure de la base de données sous forme de fichier XML ou HTML. Inversement, il est possible d'utiliser une définition de structure enregistrée au format XML pour générer à la volée une nouvelle base de données 4D.

Ces nouvelles possibilités répondent à différents besoins :

- permettre de représenter des structures sous des formes personnalisées (rapports, tableaux....) ou de les analyser dans d'autres environnements,
- permettre de générer des bases de données à partir de fichiers de description.

Format d'une définition de structure 4D

Les définitions de structure de 4D v11 sont basées sur le format XML. Vous pouvez visualiser une définition de structure à l'aide d'un simple éditeur de texte. Le format XML permet également d'envisager tout type d'exploitation, notamment via des transformations XSL. 4D utilise d'ailleurs un fichier .XSL pour exporter la définition de structure au format HTML.

Une définition de structure inclut les tables, les champs, les index, les liens, leurs attributs, ainsi que diverses caractéristiques nécessaires à la decription complète de la structure. La "grammaire" interne des définitions de structure 4D est documentée par l'intermédiaire de fichiers DTD — également utilisés pour la validation des fichiers XML. Les fichiers DTD utilisés par 4D sont regroupés dans le dossier **DTD**, situé à côté de l'application 4D. Les fichiers **base_core.dtd** et **common.dtd** sont utilisés pour la description de structure. Pour plus d'informations sur les définitions de structure 4D, n'hésitez pas à consulter ces fichiers ainsi que les commentaires qu'ils contiennent.

Exporter une définition de structure

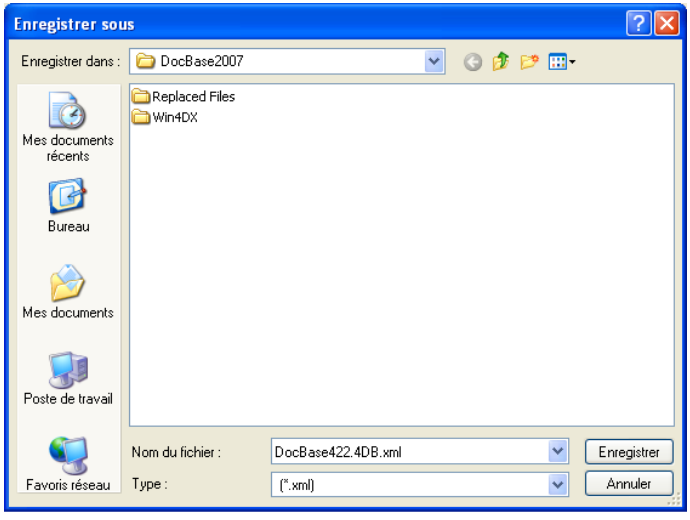
4D v11 vous permet d'exporter une structure au format XML ou HTML. Choisissez le format en fonction de vos besoins :

- **Format XML** : une structure au format XML pourra être visualisée dans un simple éditeur de texte ou exploitée de différentes manières (transformation XSL personnalisée, importation et analyse dans un autre logiciel, etc.).
  Choisissez ce format si vous souhaitez utiliser la définition de structure pour créer de nouvelles bases.
- **Format HTML** : ce format permet la représentation de la structure sous forme de rapport, visualisable et imprimable via un navigateur.

▶ Pour exporter une définition de structure en XML:

1 **Sélectionnez la commande Exporter > Définition de structure vers le fichier XML... dans le menu Fichier de 4D.**

Une boîte de dialogue standard d'enregistrement apparaît, vous permettant de désigner le nom et l'emplacement du fichier à exporter.



2 **Désignez le nom et l'emplacement de l'export et validez la boîte de dialogue.**

▶ Pour exporter une définition de structure HTML :

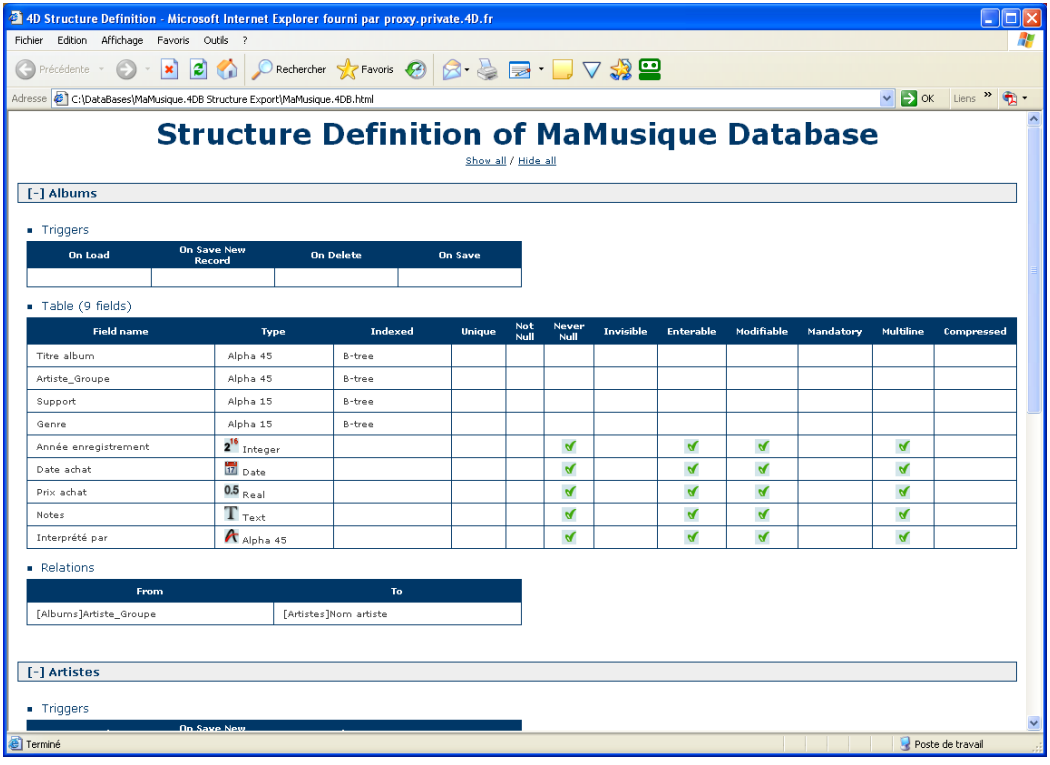
1 **Sélectionnez la commande Exporter > Définition de structure vers le fichier HTML... dans le menu Fichier de 4D.**

Une boîte de dialogue de sélection de dossier apparaît, vous permettant de désigner l'emplacement où seront stockés les fichiers HTML.

2 Cliquez sur le bouton Créer un nouveau dossier ou désignez un dossier existant.

4D v11 crée automatiquement à l'emplacement désigné un dossier nommé "*Mastructure.4db* Structure Export" (*Mastructure.4db* est le nom du fichier de structure de la base) contenant les éléments exportés.

Une boîte de dialogue vous permet de visualiser directement le résultat de l'export dans le navigateur par défaut. Les définitions de structure au format HTML ont l'apparence suivante :



Personnaliser la transformation XSL

Pour générer des pages HTML de définition de structure, 4D effectue des transformations XSL par défaut en utilisant le fichier "Structure_to_html.xsl" placé dans le sous-dossier /Resource/*langue*.lproj de l'application.

Note Si ce fichier n'est pas présent, l'exportation html n'est pas proposée dans la boîte de dialogue d'exportation.

Vous pouvez personnaliser ces transformations comme vous le souhaitez en utilisant un fichier de feuille de style XSL personnalisé.

Pour cela, il suffit de créer un fichier nommé “Structure_to_html.xsl” (vous pouvez dupliquer le fichier par défaut) et de le placer au même niveau que le fichier .4db. 4D utilisera alors ce fichier pour générer la définition de structure au format html.

Créer une base de données à partir d’une définition de structure

Les définitions de structures exportées au format XML peuvent être utilisées pour créer à la volée de nouvelles bases de données à l’identique. Dans ce cas, la définition de structure peut être considérée comme un modèle de structure, qu’il est possible de dupliquer à loisir.

Une définition de structure XML peut être utilisée telle quelle ou être modifiée au préalable via un éditeur XML. Ce principe permet d’envisager tout type de mécanisme ayant pour but de générer des structures par programmation.

Par ailleurs, le format interne des fichiers XML de description des structures 4D étant public (cf. paragraphe “Format d’une définition de structure 4D”, page 131), il est possible de construire ce type de fichier depuis d’autres environnements de bases de données ou toute application de conception afin de générer automatiquement des bases de données 4D.

► Pour créer une base de données depuis une définition de structure :

1 **Sélectionnez la commande Nouveau > Base de données à partir d’une définition de structure... dans le menu Fichier de 4D.**

Une boîte de dialogue standard d’ouverture de documents apparaît, vous permettant de désigner le fichier de description à ouvrir. Vous devez sélectionner un fichier au format XML respectant la “grammaire” des descriptions de structure 4D (le programme valide le fichier via la DTD).

2 **Sélectionnez un fichier XML de description de structure puis cliquez sur OK.**

4D affiche une boîte de dialogue vous permettant de choisir le nom et l’emplacement de la base de données à créer.

3 **Choisissez le nom et l’emplacement de la base de données à créer et cliquez sur le bouton Enregistrer.**

Si le fichier XML est valide, 4D referme la base courante (le cas échéant) et crée une nouvelle structure basée sur la définition de structure et affiche la fenêtre de l’Explorateur. Un fichier de données vide est également créé par défaut.

6 Editeur de méthodes

L'éditeur de méthodes de 4D v11 propose plusieurs nouveautés :

- possibilité de saisie du code SQL,
- affichage prédictif contextuel des constantes,
- nouvelles options de recherche et de remplacement,
- nouvelle architecture des macros.

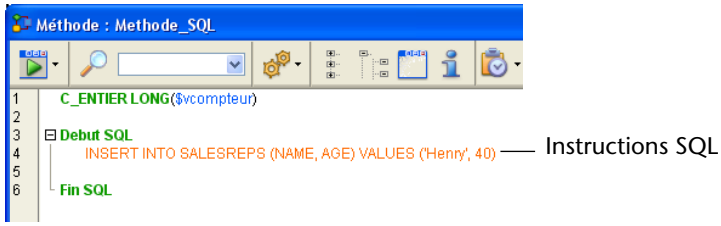
Saisir du code SQL dans l'éditeur de méthodes

4D v11 est doté d'un moteur SQL natif (cf. chapitre "Utiliser le moteur SQL de 4D", page 205).

La saisie directe de code SQL dans l'éditeur de méthodes est l'un des moyens permettant de tirer parti de ce nouveau moteur.
Les deux autres moyens sont les commandes ODBC (cf. paragraphe "Sources de données externes", page 260) et la nouvelle routine CHERCHER PAR SQL, page 266 .

Pour insérer du code SQL dans l'éditeur de méthodes de 4D, le principe est simple : il suffit d'encadrer le code SQL avec les marqueurs **Debut SQL** et **Fin SQL**.

Les instructions incluses dans le paragraphe ainsi délimité ne seront pas analysées par l'interpréteur classique de 4D :



Les principes de fonctionnement de ces balises sont les suivants :

- Vous pouvez placer un ou plusieurs blocs de balises Debut SQL/Fin SQL dans la même méthode. Vous pouvez générer des méthodes entièrement composées de code SQL ou mixer du code 4D et du code SQL dans la même méthode.
- Vous pouvez écrire plusieurs instructions SQL sur une même ligne ou sur différentes lignes en les séparant par un ";". Par exemple, vous pouvez écrire :

```
Debut SQL
   INSERT INTO SALESREPS (NAME, AGE) VALUES ('Henry',40);
   INSERT INTO SALESREPS (NAME, AGE) VALUES ('Bill',35)
Fin SQL
```

ou :

```
Debut SQL
   INSERT INTO SALESREPS (NAME, AGE) VALUES ('Henry',40);INSERT
                              INTO SALESREPS (NAME, AGE) VALUES ('Bill',35)
Fin SQL
```

A noter que le débogueur de 4D évaluera le code SQL ligne par ligne. Dans certains cas, il peut être préférable d'utiliser plusieurs lignes.

Déboguer le code SQL

Le débogueur standard de 4D permet d'analyser les instructions SQL situées dans un bloc Debut SQL/Fin SQL. Les instructions SQL écrites sur plusieurs lignes seront évaluées ligne par ligne, comme les instructions standard du langage 4D. S'il y a plusieurs instructions SQL sur une même ligne, le débogueur de 4D les évaluera en une seule fois.

Le débogueur permet d'obtenir avec le code SQL les mêmes informations qu'avec le code 4D standard. Pour plus d'informations sur le débogueur, reportez-vous au manuel *Langage* de 4D.

Les erreurs liées au SQL peuvent être interceptées par une méthode installée via un APPELER SUR ERREUR. En cas d'erreur, si aucune méthode d'appel sur erreur n'est installée, 4D interrompt l'exécution de la commande et de la méthode courantes.

Noms des éléments de la structure 4D

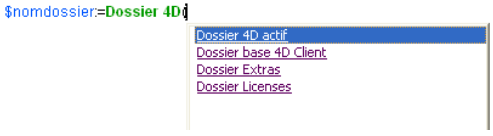
Dans le cas d'une instruction SQL contenant un champ (une colonne) 4D comme dans l'exemple suivant :

```
SELECT eName FROM PERSONS INTO <<[MATABLE]Nom>>
```

Si le nom de la table ou du champ 4D est modifié au niveau de la structure, il ne sera plus reconnu dans le code, vous devez le mettre à jour dans chaque instruction SQL.

Constantes contextuelles

Dans l'éditeur de méthodes de 4D v11, l'insertion de constantes à l'aide de la fonction de saisie prédictive s'effectue désormais de façon contextuelle. Les constantes affichées dans la fenêtre de saisie prédictive dépendent du contexte de saisie. En d'autres termes, seules les constantes pertinentes pour le paramètre de la commande en cours d'écriture sont désormais proposées dans la fenêtre :



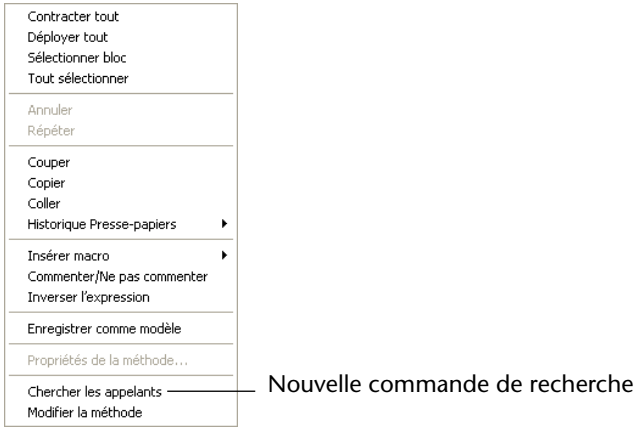
Ce mécanisme fonctionne pour chaque paramètre des commandes ainsi que pour les fonctions.

Recherches

Deux nouvelles fonctions liées à la recherche et au remplacement sont disponibles dans l'éditeur de méthodes : la recherche des appelants et le remplacement sélectif.

Chercher les appelants

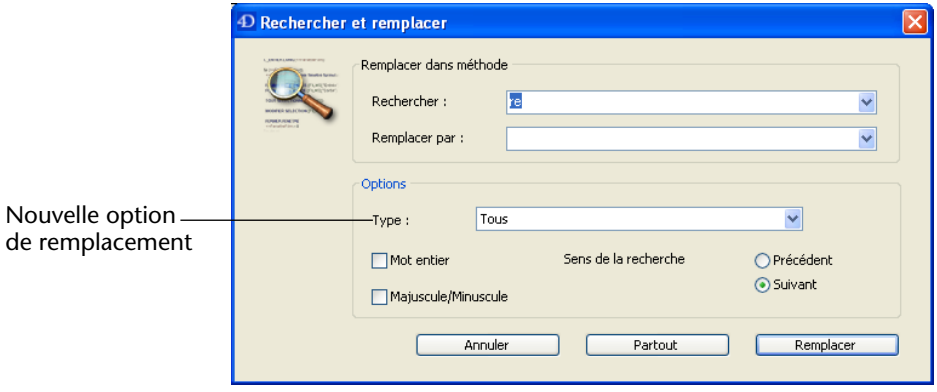
Une nouvelle commande est disponible dans le menu contextuel de l'éditeur de méthodes lorsqu'une méthode projet est sélectionnée : **Chercher les appelants**.



Cette commande affiche dans une fenêtre de résultat la liste des objets (méthodes ou menus) référençant la méthode projet sélectionnée. Elle est également disponible depuis l'Explorateur (cf. paragraphe "Rechercher les appelants", page 81).

Option de remplacement sélectif

Dans 4D v11, la boîte de dialogue standard "Rechercher et remplacer" de l'éditeur de méthodes comporte une nouvelle option permettant de restreindre les remplacements à un seul type d'objet cible :



Les types suivants sont proposés :

- **Tous** (fonctionnement standard)
- **Variable** : ne remplace la chaîne que dans les noms de variables
- **Méthode** : ne remplace la chaîne que dans les noms de méthodes
- **Commentaires** : ne remplace la chaîne que dans les commentaires.

Nouvelle architecture des macros

Les fichiers de macro-commandes de 4D v11 sont désormais entièrement conformes à la norme xml. Cela signifie notamment que les lignes de déclaration xml <?xml version="1.0" ...?> et de déclaration de document < !DOCTYPE macros SYSTEM "http://www.4d.com/dtd/2007/macros.dtd"> sont obligatoires au début d'un fichier de macro pour que celui-ci soit chargé. Les différents types d'encodages du xml sont pris en charge. Il est toutefois recommandé d'utiliser un encodage compatible Mac/PC (UTF-8).

4D fournit une DTD permettant de valider les fichiers de macros. Ce fichier est situé à l'emplacement suivant :

- Windows : 4D Developer\Resources\DTD\macros.dtd
- Mac OS : 4D Developer.app:Contents:Resources:DTD:macros.dtd

Si un fichier de macros ne contient pas les lignes de déclaration ou ne peut pas être validé, il n'est pas chargé.

Ce nouveau mécanisme est incompatible avec le format des macros des versions précédentes. Par conséquent, 4D v11 charge les macros à partir d'un nouveau dossier, nommé "**Macros v2**". Les macros doivent être présentes sous forme d'un ou plusieurs fichiers xml placés dans ce dossier.

Le dossier "Macros v2" peut être situé :

- dans le dossier **4D** actif de la machine (comme dans les versions précédentes de 4D). Les macros sont alors partagées pour toutes les bases.
- à côté du fichier de structure de la base. Les macros ne sont chargées que pour cette structure.
- pour les composants : dans le dossier Components de la base. Les macros ne sont alors chargées que si le composant est installé.

Ces trois emplacements peuvent être utilisés simultanément : il est possible d'installer un dossier "Macros v2" à chaque emplacement. Les macros seront chargées dans l'ordre suivant : dossier 4D, fichier de structure, composant 1... composant N.

Migration des macros précédentes

Un mécanisme de conversion automatique est mis en place pour assurer la compatibilité de 4D v11 avec les macros définies dans les versions précédentes. Au moment du lancement de 4D, le programme vérifie la présence du dossier "Macros v2" dans le dossier **4D** actif.
S'il existe, les fichiers macros qu'il contient sont chargés.
S'il n'existe pas, il est créé et 4D vérifie la présence du fichier "Macros.xml" (4D v2003/2004) et/ou du dossier "Macros" (4D 2004). Le contenu de ce fichier et/ou de ce dossier est converti et recopié dans le nouveau dossier "Macros v2". Lors de la conversion des fichiers macros existants, des lignes de déclaration xml et de document sont ajoutées.

Si le dossier 4D actif ne contient aucun fichier ou dossier macros antérieur, un dossier "Macros v2" contenant les macros par défaut est créé.

Incompatibilités liées à la norme xml

Le respect de la norme xml pour les fichiers de macros nécessite l'observation de règles de syntaxe strictes.
Ce principe peut entraîner des incompatibilités avec le code de macros existantes et empêcher le chargement des fichiers xml. Les principales sources de dysfonctionnements sont les suivantes :

- Les commentaires du type "// mon commentaire", autorisés à l'intérieur des éléments <macro> dans le format de macros précédent, sont incompatibles avec la syntaxe xml.
Les lignes de commentaires doivent respecter la forme standard "<!-- mon commentaire -->".
- Les symboles <> employés notamment pour les noms des objets inter-process doivent être encodés. Par exemple, la variable <>*params* devra être écrite *<>params*.
- La balise de déclaration initiale <macros> pouvait être omise dans les versions précédentes de 4D. Elle est désormais obligatoire, sinon le fichier n'est pas chargé.

Nouveaux attributs pour l'élément <macro>

Deux nouveaux attributs sont pris en charge pour l'élément **<macro>** : **method_event** et **version.**

- **method_event** : cet attribut permet de provoquer l'appel automatique de la macro en fonction de la phase courante de manipulation de chaque méthode (création, fermeture...).
  Il peut prendre l'une des valeurs suivantes :
  - *on_load* : la macro est déclenchée à l'ouverture de chaque méthode,
  - *on_save* : la macro est déclenchée au moment de la sauvegarde de chaque méthode (fermeture d'une méthode modifiée ou sauvegarde via la commande **Fichier>Sauvegarder**),
  - *on_create* : la macro est déclenchée à la création de chaque méthode,
  - *on_close* : la macro est déclenchée à la fermeture de chaque méthode.

  on_save et *on_close* peuvent être utilisés conjointement — en d'autres termes, ces deux événements sont générés dans le cas de la fermeture d'une méthode modifiée.
  En revanche, *on_create* et *on_load* ne sont jamais générés de façon consécutive.

  Ce nouvel attribut permet par exemple de préformater les méthodes au moment de leur création (commentaires dans la zone d'en-tête) ou d'enregistrer des informations du type date et heure au moment de leur fermeture.

- **Version** : cet attribut permet d'activer le nouveau mode de prise en charge des sélections de texte pour la macro (cf. paragraphe suivant). Pour activer le nouveau mode, passez l'attribut **version="2"** dans l'élément Macro.
  Si vous omettez cet attribut ou passez version="1" dans l'élément Macro, l'ancien mode est conservé.

Variables de sélection de texte des méthodes

Dans les versions précédentes de 4D, le programme maintenait automatiquement un ensemble de variables process pour la manipulation de texte dans les méthodes lors de l'emploi de la balise <method> : des variables d'entrée (*_textSel*, *_blobSel*, etc.) pour récupérer du texte et des variables de sortie (*_textReplace*, *_blobReplace*, etc.) pour insérer du texte.

Par soucis de compatibilité, ce mécanisme est toujours pris en charge dans 4D v11, mais il est désormais obsolète pour les raisons suivantes :

- l'emploi de variables BLOB pour la gestion de textes de taille supérieure à 32000 caractères n'est plus nécessaire (cf. paragraphe "Extension des capacités", page 33),
- la gestion de variables est incompatible avec la nouvelle architecture des composants, dans laquelle les espaces d'exécution des variables sont cloisonnés (cf. paragraphe "Nouvelle architecture des composants", page 50). Un composant version 11 ne peut pas accéder aux textes des méthodes de la base hôte et inversement via les variables prédéfinies.

  Par conséquent, il est désormais conseillé d'utiliser le nouveau mode de prise en charge des sélections de texte via les commandes LIRE PARAMETRE MACRO et FIXER PARAMETRE MACRO. Ces commandes permettent de s'affranchir du cloisonnement des espaces d'exécution base hôte/composants et autorisent donc la création des composants dédiés à la gestion de macros.

  Pour activer ce nouveau mode pour une macro, vous devez déclarer le nouvel attribut **Version** avec la valeur **2** dans l'élément Macro. Dans ce cas, 4D ne gère plus les variables prédéfinies *_textSel*, *_textReplace* etc. et les commandes LIRE PARAMETRE MACRO et FIXER PARAMETRE MACRO sont utilisables. Cet attribut doit être déclaré ainsi :

  ```
  <macro name="MaMacro" version="2">
  --- Texte de la macro ---
  </macro>
  ```

  Si vous ne passez pas cet attribut, le mode précédent est conservé.

7 Formulaires et objets

De nombreuses modifications dans 4D v11 sont relatives à la création et la gestion des formulaires et de leurs objets :

- les nouveaux formulaires projet permettent de générer des interfaces indépendantes des tables,
- de nombreuses nouveautés sont disponibles pour les List box,
- la gestion des champs et variables image a été optimisée,
- diverses nouvelles fonctions sont disponibles dans l'éditeur de formulaires : nouveaux badges, affichage des sous-formulaire en mode Page, options de glisser-déposer, apparence "métal" (Mac OS), thermomètre "barber shop",
- plusieurs nouveautés concernent les formats d'affichage des dates et des heures ainsi que les caractères d'emplacement des numériques.

Formulaires projet

4D v11 permet de créer et d'utiliser des formulaires "autonomes", non rattachés à des tables. Ces formulaires sont appelés **formulaires projet**.

Par souci de clarification, les formulaires 4D classiques, c'est-à-dire les formulaires associés à des tables, sont appelés **formulaires table**.

Note La mise en place des formulaires projet a entraîné des modifications au niveau du langage de 4D. Pour plus d'informations, reportez-vous au paragraphe "Formulaires projet", page 277.

Pourquoi utiliser des formulaires projet ?

La création de formulaires projet a été rendue nécessaire dans 4D v11 par la mise en place de la nouvelle architecture des composants (cf. paragraphe "Nouvelle architecture des composants", page 50). Avec

cette architecture, les composants peuvent contenir des formulaires mais ils ne peuvent pas contenir de tables. Il était donc indispensable de permettre la création de formulaires indépendants des tables.

En outre, les formulaires projet répondent au besoin de formulaires dédiés à l'interface (généralement appelés via la commande DIALOGUE) et non spécifiquement liés à des tables. Dans les versions précédentes de 4D, il était d'usage de créer une table [Interface] servant uniquement à générer les formulaires des dialogues.

Enfin, les formulaires projet permettent de créer plus facilement des interfaces conformes aux normes des systèmes d'exploitation. En particulier, l'appel via la commande DIALOGUE de formulaires projet affichant des sélections d'enregistrements dans des sous-formulaires est désormais recommandé par 4D pour l'affichage d'enregistrements en liste. Moyennant un peu de programmation supplémentaire, cette combinaison remplace avantageusement les commandes MODIFIER SELECTION et VISUALISER SELECTION.

Caractéristiques

Les formulaires projet se distinguent des formulaires table par les caractéristiques suivantes :

- Les formulaires projet peuvent être uniquement de type détaillé (page). Les mécanismes des formulaires sortie (liste) ne sont pas compatibles avec les formulaires projet.
- Les formulaires projet n'apparaissent pas dans la Liste des tables et ne peuvent pas être désignés comme formulaire entrée ou sortie courant. Ils ne peuvent pas être utilisés dans l'éditeur d'étiquettes ni l'éditeur d'import/export de 4D.
  Les formulaires projet peuvent être affichés uniquement via la commande DIALOGUE ou en tant que formulaires hérités.
- Les formulaires projet peuvent contenir les mêmes types d'objets que les formulaires table, y compris des champs.
  Lorsque des champs sont utilisés, le formulaire projet stocke le numéro de la table et du champ. En cas de copie du formulaire d'une base à une autre ou dans le cadre d'un composant, les références sont copiées. La table et le champ utilisés sont ceux de la base d'arrivée. En cas d'incompatibilité (table inexistante, type de champ incorrect...), le formulaire ne fonctionnera pas correctement.
  Comme les formulaires projet sont principalement destinés à être utilisés dans le contexte de la commande DIALOGUE, les actions standard

relatives à la gestion des enregistrements (Enregistrement suivant, Supprimer enregistrement, etc.) ne sont pas proposées par défaut dans l'éditeur ni dans l'assistant de création de formulaire. Vous devez gérer via les commandes du langage l'affichage et les éventuelles modifications des données des enregistrements.
En revanche, lorsque les formulaires projet sont utilisés comme formulaires hérités par des formulaires table, l'usage des mécanismes automatiques de gestion des enregistrements est possible.

- Les formulaires projet peuvent comporter une méthode formulaire, comme les formulaires table. La page "Méthodes" de l'Explorateur a été modifiée afin de refléter ce nouveau type de méthode formulaire (cf. paragraphe "Page Méthodes", page 83).

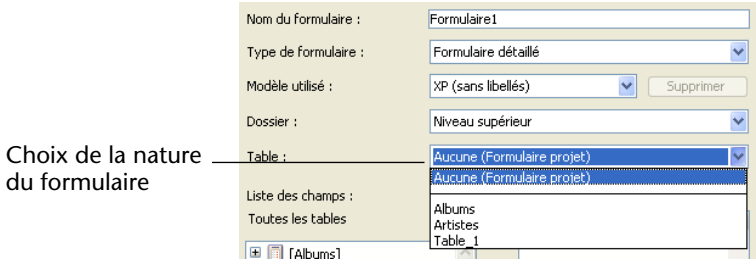
Création

Les formulaires projet sont créés comme les formulaires table : vous pouvez créer des formulaires projet "vides" à partir de l'Explorateur puis les remplir à l'aide de l'éditeur de formulaires, ou utiliser l'assistant de création de formulaires.

La nature du formulaire est définie au moment de sa création, mais vous pouvez à tout moment transformer un formulaire projet en formulaire table et inversement.

Utiliser l'assistant de création de formulaire

Pour créer un formulaire projet à l'aide de l'assistant de création de formulaire, sélectionnez l'option **Aucune (Formulaire projet)** dans le champ "Table :" de la première page de l'assistant :



Dans ce cas, seule la liste de toutes les tables est disponible.

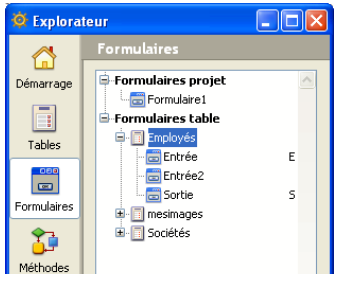
Lorsque vous sélectionnez cette option, le pop up menu "Type de formulaire" propose uniquement les types **Formulaire détaillé** et **Formulaire impression détaillé**. En effet, il n'est pas possible de créer des formulaires projet de type liste (cf. paragraphe "Caractéristiques", page 144).

Dans les pages avancées de l'assistant (disponibles lorsque vous cliquez sur le bouton **Avancé...**), certaines options non compatibles avec le concept de formulaires projet ne sont pas accessibles :

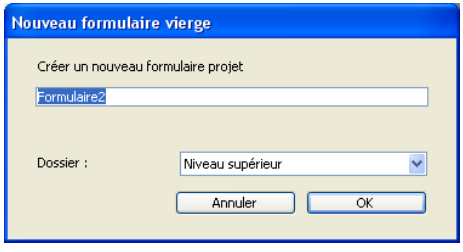
- Comme la première page de l'assistant, la page "Champs" n'affiche pas de table principale et propose uniquement la liste de toutes les tables. L'option "Champs liés saisissable" n'est pas disponible.
- Dans la page "Options", l'option "Nombre d'enregistrements" n'est pas disponible.
- Dans la page "Boutons", les actions prédéfinies relatives aux enregistrements ne sont pas disponibles (Enregistrement suivant, Dernier enregistrement, Supprimer enregistrement, etc.).
- La page "Sous-formulaire" est identique à celle des formulaires table. A noter que la page "Sous-formulaire" est désormais disponible dans tous les cas, même si la table ne comporte pas de lien.

Utiliser l'Explorateur

Vous pouvez créer des formulaires projet vierges depuis l'Explorateur. Dans 4D v11, la page "Formulaires" propose deux nouveaux éléments par défaut, placés au premier niveau de la liste : **Formulaires projet** et **Formulaires table** :



Pour créer un formulaire projet vierge, sélectionnez le libellé **Formulaires projet** et cliquez sur le bouton d'ajout . La boîte de dialogue suivante s'affiche :



Un formulaire projet peut avoir le même nom qu'un formulaire table existant dans la base. La nouvelle commande PAS DE TABLE PAR DEFAUT permet de s'assurer de l'utilisation du formulaire projet lorsqu'un formulaire table du même nom existe (cf. paragraphe "Formulaires projet", page 277). En revanche, deux formulaires projet ne peuvent pas avoir le même nom.

Note Les propriétés **Formulaire d'entrée** et **Formulaire de sortie** sont liées à la notion de table. Par conséquent, ces propriétés ne peuvent pas être sélectionnées pour les formulaires projet. Les lettres **E** et **S** n'apparaissent pas dans la liste des formulaires projet.

Transformer un formulaire table en formulaire projet (et inversement)

Il est possible à tout moment de transformer un formulaire table en formulaire projet ou d'effectuer l'opération inverse.
Attention, dans le cas de la transformation de formulaires table en formulaires projet, les éventuels automatismes relatifs à la gestion des données présents dans le formulaire table ne fonctionneront plus, une fois le formulaire transformé. De même, un formulaire de type "liste écran" ou "impression liste" sera transformé en formulaire projet de type "page".

Le changement de nature d'un formulaire est effectué par glisser-déposer ou copier/coller dans la page **Formulaires** de l'Explorateur. Vous pouvez effectuer cette opération dans la même base ou entre deux bases différentes.

► Pour transformer un formulaire projet en formulaire table ou inversement :

1 **Dans la page Formulaires de l'Explorateur, cliquez sur le formulaire à transformer et déposez-le sur l'élément de destination.**

Dans le cadre de la transformation d'un formulaire projet en formulaire table, vous devez déposer le formulaire sur le nom de la table à laquelle le rattacher.

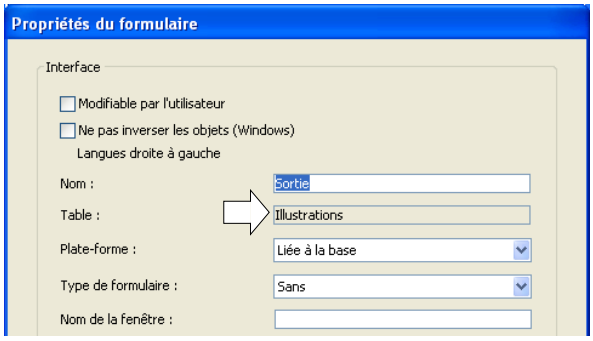
Par défaut, le formulaire est déplacé lorsque le glisser-déposer est effectué à l'intérieur de la même base de données. Si vous souhaitez copier le formulaire, maintenez la touche **Alt** (Windows) ou **Option** (Mac OS) enfoncée pendant le glisser-déposer. Lorsque le glisser-déposer a lieu entre deux bases différentes, seule la copie du formulaire est possible.

Vous pouvez également utiliser les commandes standard **Copier/ Coller** du menu contextuel de l'Explorateur.

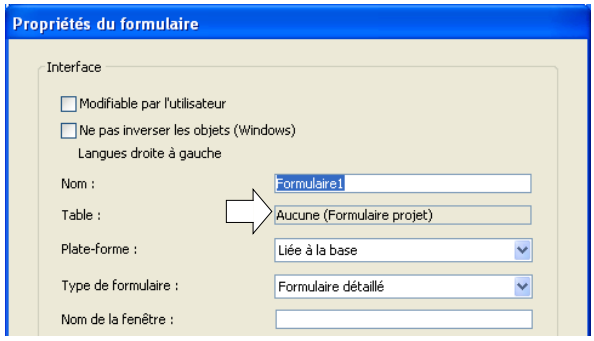
Propriétés

La boîte de dialogue des propriétés de formulaires comporte désormais le champ “Table” (non saisissable), indiquant la table à laquelle appartient le formulaire. Pour un formulaire projet, ce champ contient la valeur “Aucune (Formulaire projet)” :

Formulaire table



Formulaire projet



Cette nouveauté permet notamment de distinguer les formulaires projet des formulaires table.

List box

Plusieurs nouveautés relatives aux objets de type List box sont disponibles dans 4D v11 :

- possibilité d’associer des champs ou des expressions aux colonnes des list box,
- possibilité de remplir une list box avec le résultat d’une requête SQL.

Note Ces nouveautés entraînent des modifications au niveau des commandes du thème “List box” dans le langage de 4D v11. A ce sujet, reportez-vous au paragraphe “List box”, page 337.

Associer des champs ou des expressions aux list box

Dans 4D v11, il est possible d’associer des champs ou des expressions à des colonnes de list box. Dans les versions précédentes de 4D, seuls des tableaux pouvaient être associés aux list box.

Lorsqu’une list box est associée à un champ, elle peut fonctionner avec la sélection courante de la base ou une sélection temporaire.

Dans le cas de la sélection courante, toute modification effectuée côté base de données est automatiquement reportée dans la list box et

inversement. La sélection courante est donc toujours identique aux deux emplacements.

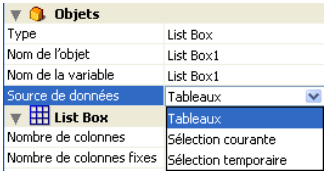
Vous pouvez également associer une colonne de list box à une expression. L'expression pourra être basée sur un ou plusieurs champs (par exemple [Employés]Nom+" "+[Employés]Prénom) ou être simplement une formule (par exemple Chaine(Millisecondes)). L'expression peut également être une méthode projet, une variable ou un élément de tableau.

Note Il n'est pas possible de combiner dans une même list box des colonnes basées sur des champs (ou des expressions) et des colonnes basées sur des tableaux.

Nouvelles propriétés des list box

De nouvelles propriétés de list box permettent de prendre en charge l'association de champs et d'expressions. A noter que les colonnes de list box disposent également de nouvelles propriétés (ce point est traité dans le paragraphe suivant).

- **Source de données**
  La nouvelle propriété **Source de données** a été ajoutée dans le thème "Objets". Elle permet de définir le type de données avec lequel la list box va fonctionner : sélection courante, sélection temporaire ou tableaux.



  - Sélectionnez l'option **Sélection courante** si vous souhaitez utiliser des expressions, des champs ou des méthodes dont les valeurs seront évaluées pour chaque enregistrement de la sélection courante d'une table.
  - Sélectionnez l'option **Sélection temporaire** si vous souhaitez utiliser des expressions, des champs ou des méthodes dont les valeurs seront évaluées pour chaque enregistrement d'une sélection temporaire.
  - Si l'option **Tableaux** est sélectionnée, le fonctionnement de la list box est identique à celui des versions précédentes de 4D. L'option **Tableaux** est nécessaire si vous souhaitez pouvoir récupérer

le résultat d'une requête SQL dans la list box (cf. paragraphe "Affichage du résultat d'une requête SQL", page 159).

- **Table principale**
  Cette nouvelle propriété est ajoutée au thème "Source de données" lorsque la source de données **Sélection courante** est choisie.
  Cette propriété permet de désigner la table dont la sélection courante sera utilisée. Cette table et sa sélection courante constitueront la référence pour les champs associés aux colonnes de la list box (références de champs ou expressions contenant des champs). Même si certaines colonnes contiennent des champs d'autres tables, le nombre de lignes affichées sera défini par la table principale.

  Le menu associé à cette propriété affiche toutes les tables de la base, que le formulaire soit relié à une table (formulaire table) ou non (formulaire projet). Par défaut, la propriété affiche la première table de la base.

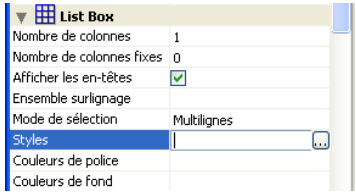
  Pour plus d'informations sur le fonctionnement de cette propriété, reportez-vous au paragraphe "Saisie de données", page 158.

- **Sélection temporaire**
  Cette nouvelle propriété est ajoutée au thème "Source de données" lorsque la source de données **Sélection temporaire** est choisie.
  Vous devez saisir le nom d'une sélection temporaire valide. La sélection temporaire peut être process ou interprocess.
  Le contenu de la list box sera basé sur cette sélection. La sélection temporaire devra exister et être valide au moment de l'affichage de la list box, sinon la list box s'affichera vide. Si vous laissez vide la zone de nom, la list box s'affichera vide.

Note Les sélections temporaires sont des listes ordonnées d'enregistrements. Elles permettent de conserver en mémoire l'ordre et l'enregistrement courant d'une sélection. Pour plus d'informations, reportez-vous au manuel *Langage* de 4D.

- **Propriétés de style (styles, couleurs de police, couleurs de fond)**
  Les noms de ces propriétés de style du thème "List Box" sont modifiés lorsque la source de données **Sélection courante** ou **Sélection temporaire** est sélectionnée. Dans ce cas, les propriétés **Tableau de styles**, **Tableau couleurs de police** et **Tableau couleurs de fond** sont remplacées par **Styles**, **Couleurs de police** et **Couleurs de fond**.

En effet, avec une source de données de type “sélection”, il n’est pas possible d’utiliser des tableaux pour les styles. Vous devez utiliser des expressions ou des variables (hormis des tableaux) pour les styles, les couleurs de police et les couleurs de fond. L’expression ou la variable sera évaluée pour chaque ligne affichée.
Vous pouvez utiliser l’éditeur de formules pour chaque propriété. Pour définir une expression de propriété via l’éditeur de formules, cliquez sur le bouton [...] qui apparaît lorsque vous sélectionnez la zone :



Comme dans les versions précédentes, vous pouvez utiliser les constantes du thème “Styles de caractères” pour définir la propriété **Styles**. De même, vous devez utiliser des valeurs de couleurs RVB pour définir les propriétés **Couleurs de police** et **Couleurs de fond** — vous pouvez utiliser les constantes du thème “FIXER COULEURS RVB” (reportez-vous à la description de la commande FIXER COULEURS RVB).
Vous pouvez aussi utiliser la nouvelle fonction Choisir (cf. page 409).

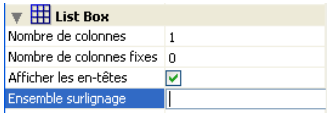
L’exemple suivant utilise des noms de variables : saisissez *StyleSociété* dans la zone **Styles** et *CouleurSociété* dans la zone **Couleurs de police**. Dans la méthode du formulaire, vous pouvez écrire le code suivant :

```
StyleSociété:=Choisir([Sociétés]ID;Gras;Normal;Italique;Souligné)
CouleurSociété:=Choisir([Sociétés]ID;Coul arrière plan;Coul claire;
                                    Coul premier plan;Coul sombre)
```

- **Ensemble surlignage**
  Cette nouvelle propriété est ajoutée dans le thème “List Box” lorsque la source de données **Sélection courante** ou **Sélection temporaire** est choisie. Elle permet de désigner l’ensemble à utiliser pour gérer le surlignage des enregistrements dans la list box (lorsque la source **Tableaux** est sélectionnée, cette gestion est effectuée via un tableau booléen du même nom que la list box, comme dans les versions précédentes de 4D).

  4D crée un ensemble par défaut, que vous pouvez modifier si nécessaire (l’ensemble peut être process ou interprocess). L’ensemble est automatiquement maintenu par 4D : si l’utilisateur sélectionne une ou plusieurs lignes dans la list box, l’ensemble est immédiatement mis

à jour. Si vous voulez sélectionner une ou plusieurs lignes par programmation, il vous suffit d'appliquer les commandes du thème "Ensembles" à cet ensemble.



Notes

• Le surlignage des lignes de la list box et celui des enregistrements de la table sont totalement indépendants.

• Si la propriété "Ensemble surlignage" ne contient aucun nom, il ne sera pas possible d'effectuer de sélection dans la listbox.

- **Mode de sélection**

  Cette nouvelle propriété apparaît dans le thème "List Box" quelle que soit la source de données choisie. Elle remplace la propriété **Sélection multiple** présente dans les versions précédentes de 4D. Elle propose le nouveau mode de sélection *Aucun*. Trois modes sont désormais disponibles :

  - *Aucun* : Aucune ligne ne peut être sélectionnée et aucune valeur ne peut être saisie. La sélection et la gestion des données peut être uniquement effectuée par programmation. Le clic ou le double-clic dans la liste n'a aucun effet (même si l'option **Saisissable** est cochée pour les colonnes), toutefois les événements formulaire Sur clic et Sur double clic sont générés. Dans ce mode, le développeur a le contrôle total des sélections (via l'ensemble de gestion du surlignage) et de la saisie de données (via la commande EDITER ELEMENT).
    Les événements formulaire Sur nouvelle sélection et Sur avant saisie ne sont pas générés. En revanche, l'événement Sur après modification est généré lorsque des données sont saisies par l'utilisateur à l'aide de la commande EDITER ELEMENT.
  - *Ligne unique* : Comme dans les versions précédentes de 4D, une seule ligne peut être sélectionnée à la fois.
  - *Multilignes* : Comme dans les versions précédentes de 4D, plusieurs lignes (adjacentes ou non) peuvent être sélectionnées.

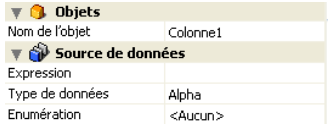
Nouvelles propriétés des colonnes de list box

Lorsque la source de données **Sélection courante** ou **Sélection temporaire** est sélectionnée pour une list box, vous pouvez associer un champ ou une expression à chaque colonne.
Dans ce cas, les propriétés **Nom de la variable** et **Type de variable**

n'apparaissent plus dans le thème "Objets" (elles restent disponibles pour les list box ayant une source de données **Tableaux**).

Note Les objets de type List box disposent également de nouvelles propriétés (ce point est traité dans le paragraphe précédent).

- **Expression**
  La nouvelle propriété **Expression** apparaît dans le thème "Source de données" lorsque la source de données de la list box est **Sélection courante** ou **Sélection temporaire** :



  Cette propriété vous permet de définir l'expression à associer à la colonne. Vous pouvez saisir :

  - une expression 4D (expression simple, formule ou méthode 4D). Le résultat de l'expression sera automatiquement affiché lorsque vous passerez en mode Application.
  - une variable simple (dans ce cas, la variable doit être explicitement déclarée en vue de la compilation). Vous pouvez utiliser tout type de variable hormis les BLOBs et les tableaux. La valeur de la variable sera généralement calculée dans l'événement Sur affichage corps.
  - un champ utilisant la syntaxe standard [Table]Champ (exemple : *[Employés]Nom*). Les types de champs suivants sont utilisables :
    - Alpha
    - Texte
    - numérique
    - Date
    - Heure
    - Image
    - Booléen

    Vous pouvez utiliser des champs de la table principale ou d'autres tables.

  Dans tous les cas, vous pouvez définir l'expression à l'aide de l'éditeur de formules de 4D en cliquant sur le bouton [...] dans la Liste des propriétés :

Si vous passez une expression, la colonne ne sera pas saisissable, même si l'option **Saisissable** est cochée.
Dans le cas d'un champ ou d'une variable, la colonne sera saisissable en fonction de l'état de l'option **Saisissable**.

En mode Développement, le type de source de données est affiché dans la première ligne de la colonne. Par exemple, *Champ=[Table1]MonChp*.

Si l'expression définie est incorrecte, la colonne de list box affichera un message d'erreur en mode Application.

- **Type de données**
La nouvelle propriété **Type de données** est disponible dans le thème "Source de données" des colonnes lorsque la source de données de la list box est **Sélection courante** ou **Sélection temporaire**.

  Ce menu permet de définir le type de l'expression ou de la variable associée à la colonne. Elle est utilisée pour déterminer le format d'affichage à appliquer et permet de mettre à jour le menu **Type d'affichage** dans le thème "Affichage".
Si un champ est saisi dans la zone Expression, la propriété **Type de données** n'est pas affichée, le format d'affichage correspond dans ce cas au type du champ.

Affichage des champs

Vous pouvez associer aux colonnes d'une list box des champs provenant de la table principale et/ou des champs provenant de tables différentes. Pour plus d'informations sur la table principale, reportez-vous au paragraphe "Nouvelles propriétés des list box", page 149.

Cependant, dans tous les cas, le contenu de la list box sera basé sur la sélection courante (ou une sélection temporaire) de la table principale de la list box :

- Si vous utilisez uniquement des champs appartenant à la table principale, le contenu des lignes de la list box sera simplement calqué sur celui de la sélection de la table principale.
- Si vous utilisez des champs n'appartenant pas à la table principale, les tables "étrangères" doivent être reliées à la table principale par des liens aller, sinon les champs "étrangers" seront affichés vides. Les liens automatiques seront activés pour chaque enregistrement de la sélection de la table principale et la list box affichera les données correspondantes dans les champs liés.
Si vous utilisez des liens manuels, vous devrez programmer l'activation des liens afin d'afficher les données dans la list box.

Si une incohérence dans la définition de la list box entraîne l'affichage de colonnes vides, un message d'erreur apparaît en mode Application dans chaque colonne incorrecte.

▼ Nous allons illustrer les différents cas à l'aide d'exemples.
Soit une base comportant deux tables, [Sociétés] et [Employés].

La sélection courante de la table [Sociétés] est la suivante :

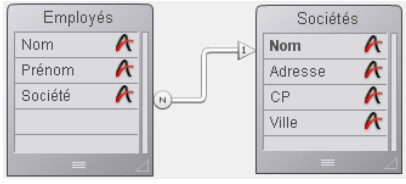
Nom
Encyclopédie internationale
Aventures et voyages
Fournier S.A.

La sélection courante de la table [Employés] est la suivante :

Prénom	**Nom**	**Société**
Corinne	Quipard	Encyclopédie internationale
André	Banjo	Aventures et voyages
Vincent	Froidure	Encyclopédie internationale
Olivier	Canal	Encyclopédie internationale
Sylvie	Ferrero	Aventures et voyages
Roland	Lenusse	Fournier S.A.
Arnaud	Forge	Encyclopédie internationale
Elizabeth	Wesson	Fournier S.A.
Rolande	Longué	Aventures et voyages
Pascale	Pradel	Aventures et voyages

Le champ [Sociétés]Nom est associé à la première colonne d'une list box. Les champs [Employés]Prénom et [Employés]Nom sont associés aux deux colonnes suivantes. La source de données de la list box est la sélection courante.

■ **Cas 1** : Les deux tables sont reliées par un lien automatique.



1) Nous choisissons la table [Employés] comme table principale. La list box affiche la sélection courante de la table [Employés] et active le lien automatique afin d'afficher le nom de la société pour chaque

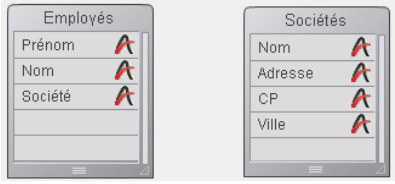
employé :

Sociétés	Prénoms	Noms
Encyclopédie internationale	Corinne	Quipard
Aventures et voyages	André	Banjo
Encyclopédie internationale	Vincent	Froidure
Encyclopédie internationale	Olivier	Canal
Aventures et voyages	Sylvie	Ferrero
Fournier S.A.	Roland	Lenusse
Encyclopédie internationale	Arnaud	Forge
Fournier S.A.	Elizabeth	Wesson
Aventures et voyages	Rolande	Longué
Aventures et voyages	Pascale	Pradel

2) Nous choisissons la table [Sociétés] comme table principale. La list box affiche la sélection courante de la table [Sociétés]. Cette sélection comportant trois enregistrements, trois lignes seulement sont affichées dans la list box. Les colonnes des champs [Employés]Prénom et [Employés]Nom sont vides :

Sociétés	Prénoms	Noms
Encyclopédie internationale		
Aventures et voyages		
Fournier S.A.		

- **Cas 2** : Les deux tables ne sont pas liées (ou sont liées par un lien manuel).

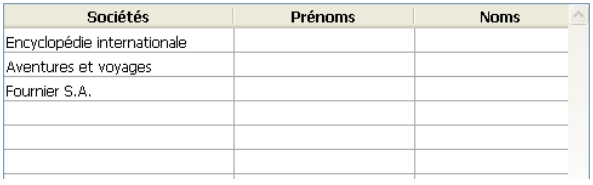


1) Nous choisissons la table [Employés] comme table principale. La list box affiche la sélection courante de la table [Employés]. La

colonne du champ [Sociétés]Nom est vide :

Sociétés	Prénoms	Noms
	Corinne	Quipard
	André	Banjo
	Vincent	Froidure
	Olivier	Canal
	Sylvie	Ferrero
	Roland	Lenusse
	Arnaud	Forge
	Elizabeth	Wesson
	Rolande	Longué
	Pascale	Pradel

2) Nous choisissons la table [Sociétés] comme table principale. La list box affiche la sélection courante de la table [Sociétés]. Cette sélection comportant trois enregistrements, trois lignes seulement sont affichées dans la list box. Les colonnes des champs [Employés]Prénom et [Employés]Nom sont vides :



Sociétés	Prénoms	Noms
Encyclopédie internationale		
Aventures et voyages		
Fournier S.A.		

Note Bien entendu, vous pouvez gérer par programmation les sélections des différentes tables et donc l'affichage des colonnes associées aux champs n'appartenant pas à la table principale.

Affichage des expressions

Pour utiliser une colonne calculée, saisissez une expression valide dans la zone **Expression** du thème "Source de données" pour la colonne. Vous pouvez utiliser l'éditeur de formules : cliquez sur le bouton [...] dans la zone Expression.

L'expression doit retourner une valeur. Vous devez indiquer le type de la valeur à l'aide de la propriété **Type de données** dans le même thème.

L'expression est évaluée pour chaque enregistrement de la sélection (courante ou temporaire) de la table principale. Si celle-ci est vide, la colonne n'affichera aucun résultat.

Tri standard

Dans les list box basées sur des sélections d'enregistrements, la fonction de **tri standard** (tri via un clic sur l'en-tête d'une colonne) est disponible uniquement :

- lorsque la source de données est la **Sélection courante**,

- avec les colonnes associées à des champs (de type alpha, numérique, date, heure ou booléen).

Dans les autres cas (list box basées sur des sélections temporaires, colonnes associées à des expressions), la fonction de tri standard n'est pas disponible.

Le tri standard de la list box modifie celui de la sélection courante dans la base de données. En revanche, les enregistrements surlignés et l'enregistrement courant ne sont pas modifiés.
Le tri standard entraîne la synchronisation de toutes les colonnes de la list box, colonnes calculées comprises.

Déplacement des lignes Dans les list box de type sélection, il n'est pas possible de réordonner les lignes en les déplaçant à l'aide de la souris (l'attribut "Lignes déplaçables" est inactif).

Saisie de données Dans les list box basées sur des sélections d'enregistrements, il est possible de saisir et de modifier des données dans les colonnes associées à des champs. Les colonnes associées à des expressions ne sont jamais saisissables.

Pour qu'une colonne associée à un champ soit saisissable, il faut que le champ ne soit pas déclaré "Non modifiable" dans l'éditeur de structure et que la propriété "Saisissable" soit appliquée à la colonne.

Lorsque l'utilisateur modifie la valeur d'un champ via une list box, la modification est immédiatement répercutée au niveau de la base de données.

Note Lorsqu'un champ est affiché dans une list box et à un autre emplacement dans un formulaire, les modifications effectuée sur le champ via la list box ne sont pas immédiatement répercutées aux autres emplacements.
En revanche, si un même champ est référencé à différents endroit d'une même list box, toute modification effectuée sur une instance du champ sera immédiatement répercutée à toutes les instances au sein de la list box.

De manière générale, tous les contrôles de saisie définis pour les champs au niveau de la structure (Obligatoire, Non modifiables...) sont pris en charge dans la list box.

Note Les propriétés des champs définies en structure sont prioritaires par rapport à celles de colonnes. Par exemple, un champ déclaré “Non modifiable” en structure ne pourra pas être modifié, même si la propriété “Saisissable” est appliquée à la colonne.

Si un enregistrement est verrouillé au niveau de la base (verrouillage lié au multiprocess ou au client/serveur), il n’est pas possible de le modifier au niveau de la list box : le message d’erreur standard indiquant que l’enregistrement est verrouillé apparaît.

Si un enregistrement est supprimé de la sélection via une commande du type SUPPRIMER ENREGISTREMENT, la ligne correspondante est automatiquement supprimée de la list box. Dans le cas de list box utilisées dans plusieurs process, une ligne vide est affichée pour l’enregistrement supprimé dans les process autres que le process courant.

Note Lorsqu’une list box est associée à un champ ou une expression, il n’est pas possible d’ajouter ou de supprimer des lignes à l’aide des commandes du thème “List Box” (reportez-vous au paragraphe “List box”, page 337).

Gestion des sélections utilisateur

Dans le cas des list box basées sur des enregistrements, la gestion des sélections utilisateur dans la list box (clics ou saisie clavier) s’effectue par la combinaison de la commande LIRE POSITION CELLULE LISTBOX et des commandes de manipulation de la sélection courante, par exemple ALLER DANS SELECTION.

Affichage du résultat d’une requête SQL

4D v11 comprend un moteur SQL performant permettant d’effectuer des requêtes sur les données (cf. chapitre “Utiliser le moteur SQL de 4D”, page 205).

Il est possible de placer directement le résultat d’une requête SQL dans une list box. Seules les requêtes de type SELECT peuvent être utilisées.

Les principes de mise en oeuvre sont les suivants :

- Vous créez la list box devant recevoir le résultat de la requête. Il est recommandé de placer dans la list box autant de colonnes qu’il y aura de colonnes SQL dans le résultat de la requête SQL (cf. ci-dessous).
- La source de données de la list box doit être **Tableaux**.

- Vous exécutez la requête SQL de type SELECT et assignez le résultat à la variable associée la list box. Vous pouvez utiliser les mots-clés Debut SQL/Fin SQL (cf.paragraphe “Echanger des données entre 4D et le moteur SQL”, page 206).
- Lorsque le contenu d’une list box provient d’une requête SQL, les colonnes ne sont ni triables ni modifiables par l’utilisateur.
- Chaque nouvelle exécution d’une requête SELECT avec la list box provoque la réinitialisation des colonnes (il n’est pas possible de remplir progressivement une même list box à l’aide de plusieurs requêtes SELECT).

Note Ce mécanisme n’est pas utilisable avec une base SQL externe.

▼ Nous voulons récupérer tous les champs de la table PERSONS et placer leur contenu dans la list box dont le nom de variable est *vlistbox*. Dans la méthode objet d’un bouton (par exemple), il suffit d’écrire :

```
Debut SQL
   SELECT * FROM PERSONS INTO <<vlistbox>>
Fin SQL
```

Nombre de colonnes

Si le nombre de colonnes SQL obtenues dépasse le nombre de colonnes définies dans la list box, 4D ajoutera automatiquement les colonnes nécessaires dans la list box et les typera en fonction du contenu des champs SQL. Ces colonnes supplémentaires seront accessibles en lecture uniquement et il ne sera pas possible de les gérer par programmation. Si vous souhaitez pouvoir accéder à toutes les colonnes de la list box par programmation, vous devez vous assurer que les nombres de colonnes correspondent exactement.

Si le nombre de colonnes définies dans la list box dépasse le nombre de colonnes obtenues par le SQL, les colonnes inutilisées sont masquées dans la list box.

Optimisation des variables et champs image

La gestion et l’affichage des champs et variables image ont été modifiées.

En premier lieu, le fonctionnement de ces deux types d’objets a été harmonisé : leur interface et les possibilités qu’ils offrent sont désormais identiques.

En outre, de nouvelles fonctions relatives à ces objets sont disponibles. Enfin, plusieurs optimisations internes ont été effectuées concernant le traitement des images dans 4D.

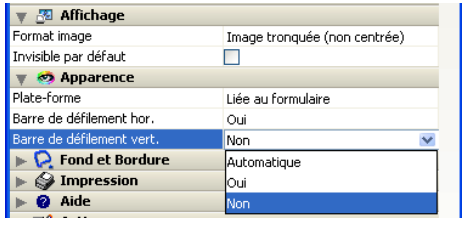
ATTENTION : *Les variables image insérées dans les formulaires doivent désormais être typées dans la Liste des propriétés. Reportez-vous au* *paragraphe "Typage des variables image dans les formulaires", page 164*.

Barres de défilement

Les objets de type image peuvent désormais comporter des barres de défilement.

Important : Pour activer les barres de défilement pour les images, le type d'affichage de l'image doit être "**Image tronquée (non centrée)**".

Dans ce cas, les propriétés **Barres de défilement hor.** et **Barres de défilement vert.** sont actives dans la Liste des propriétés pour l'objet image. Chaque propriété est définie via un menu à trois options :



- **Oui** : la barre de défilement est toujours visible, même lorsque cela n'est pas nécessaire (c'est-à-dire lorsque la dimension correspondante de l'image est plus petite que le cadre).
- **Non** : la barre de défilement n'est jamais visible.
- **Automatique** : la barre de défilement apparaît automatiquement lorsque cela est nécessaire (c'est-à-dire lorsque la dimension correspondante de l'image est supérieure au cadre).

Il est également possible de gérer les barres de défilement des objets image par programmation à l'aide de la commande CHOIX VISIBLE BARRE DEFILEMENT.

Prise en charge de formats natifs

Dans 4D v11, les formats d'image les plus courants sont désormais pris en charge en natif, en particulier les formats JPEG, PNG, BMP, GIF et TIFF (liste non exhaustive).
Cela signifie que les images sont affichées et stockées dans leur format d'origine, sans interprétation dans 4D. Les spécificités des différents formats (ombrages, zones transparentes...) sont conservées et affichées

sans altération. Dans les versions précédentes, 4D ne travaillait qu'avec des images converties au format PICT.

Cette prise en charge native est valide pour toutes les images stockées dans 4D : images de la bibliothèque, images collées dans les formulaires en mode Développement, images collées dans les champs ou variables en mode Application, etc. La nouvelle commande LISTE CODECS IMAGES permet de connaître les types natifs présents sur le poste.

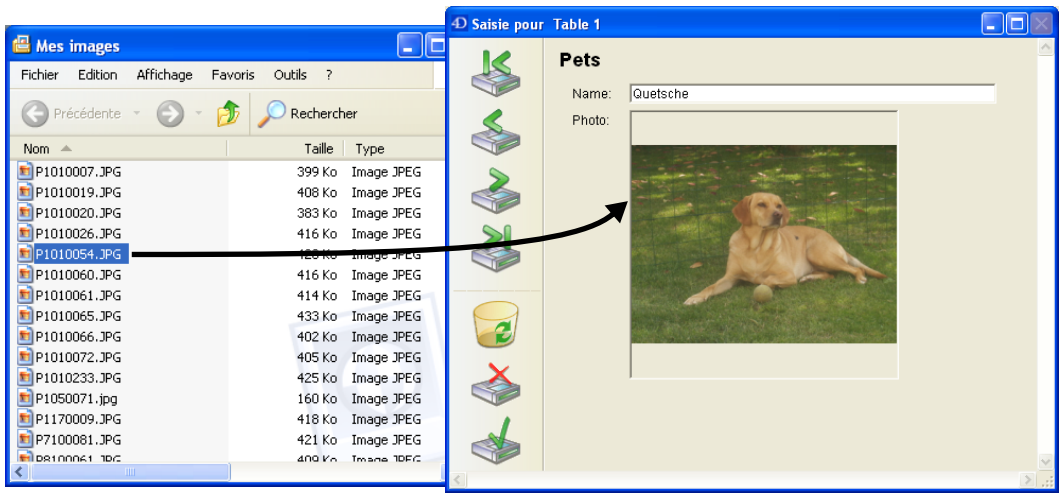
Notes

• 4D permet de "jouer" des images GIFs animées dans les formulaires lorsqu'elles sont affichées en tant qu'images statiques ou images de la bibliothèque (en mode exécution). En revanche, il n'est pas possible de "jouer" les images GIF animées stockées dans des champs ou des variables.

• Si 4D ne peut pas interpréter le format d'une image, le programme fait appel aux routines de Quicktime. Ce principe permet de conserver la compatibilité des applications utilisant des formats d'images spécifiques.

Glisser-déposer automatique

Il est désormais possible d'effectuer un glisser-déposer automatique d'une image entre une application tierce (système d'exploitation, navigateur Web, etc.) et 4D v11. Pour cela, il suffit simplement de cliquer sur le fichier image à récupérer et le déposer dans 4D (bibliothèque, formulaire en mode Développement, champ ou variable en mode Application). L'image qu'il contient est alors affichée directement dans 4D.

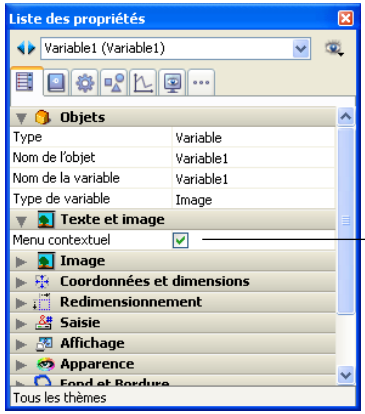


L'image d'origine doit être stockée dans l'un des formats gérés en natif par 4D (cf. paragraphe précédent). Elle est automatiquement collée en format natif dans sa zone de destination.

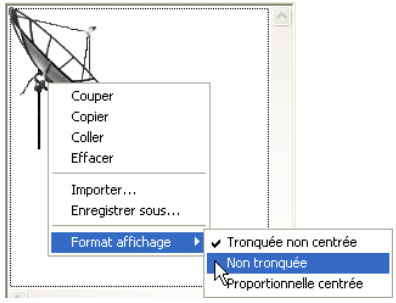
Si l'image est déposée dans un formulaire en exécution (mode Application), l'événement formulaire Sur après modification est généré.

Menu contextuel

Il est désormais possible d'associer un menu contextuel automatique aux variables et aux champs de type Image. Pour cela, il suffit de cocher l'option **Menu contextuel** dans le thème "Texte et image" de la Liste des propriétés :



Lorsque le menu contextuel est activé, l'utilisateur dispose de commandes d'édition et d'affichage lorsqu'il clique avec le **bouton droit** sur l'image en mode Application :



En plus des commandes d'édition standard (**Couper**, **Copier**, **Coller** et **Effacer**), le menu propose les commandes **Importer...**, permettant d'importer une image stockée dans un fichier et **Enregistrer sous...**, permettant de sauvegarder l'image sur disque. Ces deux commandes tirent parti de la gestion native des images : elles permettent

respectivement d'ouvrir et de sauvegarder les images dans tout format natif pris en charge par 4D.
Le menu permet également de modifier le format d'affichage de l'image : les options **Tronquée non centrée**, **Non tronquée** et **Proportionnelle centrée** sont proposées. La modification du format d'affichage via ce menu est temporaire, elle n'est pas conservée dans l'enregistrement.

Note Si la variable ou le champ image n'est pas saisissable, seules les commandes **Copier**, **Enregistrer sous...** et les commandes de formatages sont disponibles.

Optimisations

En plus des nouveautés décrites dans les paragraphes précédents, les optimisations suivantes ont été apportées aux objets de type image dans 4D v11 :

- **Interface** : les images sélectionnées dans les formulaires sont désormais entourées d'un rectangle de sélection (focus), comme les autres objets de formulaire. Dans les versions précédentes de 4D, les images sélectionnées étaient affichées en vidéo inversée.
- **Affichages multiples de la même image** : lorsqu'une même image (champ ou variable) est utilisée à plusieurs endroits dans un formulaire (par exemple une puce affichée dans un tableau), une seule instance est désormais créée en mémoire. Ce principe permet d'accélérer l'affichage des images tout et réduisant la quantité de mémoire consommée.

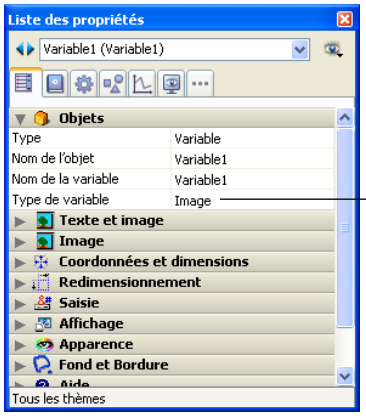
Typage des variables image dans les formulaires

Dans 4D v11, de nouveaux mécanismes natifs régissent l'affichage des variables image dans les formulaires. Ces nouveaux mécanismes requièrent davantage de rigueur lors de la configuration des variables : désormais, elles doivent obligatoirement avoir été déclarées *avant* le chargement du formulaire — c'est-à-dire avant même l'événement formulaire Sur chargement.

Pour cela, il suffit :

- soit que l'instruction C_IMAGE(nomVar) ait été exécutée avant le chargement du formulaire (typiquement, dans la méthode appelant la commande DIALOGUE),

- soit que la variable ait été typée au niveau du formulaire via le le pop up menu **Type de variable** de la Liste des propriétés :



Typage des variables image dans les formulaires

Sinon, la variable image ne sera pas affichée correctement (en mode interprété uniquement) dans 4D v11.

Note de compatibilité Les versions précédentes de 4D étant plus “tolérantes” quant à l’initialisation des variables, il est possible que ce nouveau principe entraîne des dysfonctionnements d’affichage dans les bases de données converties. Si cela se produit, assurez-vous en premier lieu que les variables image sont correctement déclarées au moyen d’une des deux solutions décrites dans ce paragraphe.

Edition des formulaires

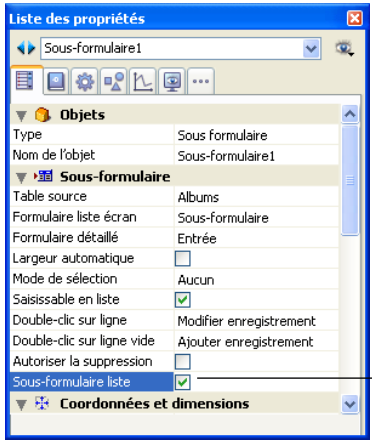
4D v11 propose diverses nouveautés relatives à l’édition et au fonctionnement des formulaires.

Sous-formulaires en page

Dans les versions précédentes de 4D, les sous-formulaires permettaient uniquement de présenter une sélection d’enregistrements sous forme de liste.

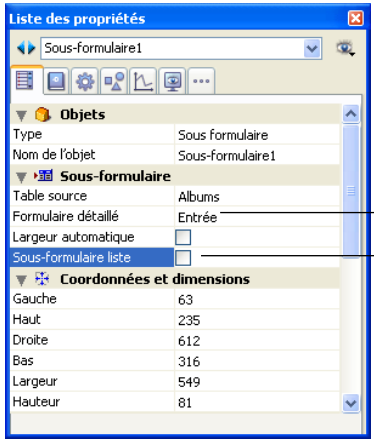
Dans 4D v11, il est possible d’utiliser des sous-formulaires en mode page, à la manière de la commande DIALOGUE.
Dans ce cas, le sous-formulaire peut afficher les données du sous-enregistrement courant ou tout type de valeur pertinente en fonction du contexte (variables, images, etc.). Il est également possible d’utiliser ce sous-formulaire pour la saisie des données.

La nouvelle propriété de sous-formulaire **Sous-formulaire liste** permet d'activer ce fonctionnement :



Lorsque cette option est cochée, le sous-formulaire est utilisé en mode liste (comme dans les versions précédentes de 4D). Par défaut, cette option est cochée.

Pour activer le mode page, il suffit de désélectionner cette option. Dans ce cas, les propriétés relatives à la configuration du sous-formulaire en liste (Mode de sélection, Double-clic sur ligne, etc.) ne sont plus affichées :



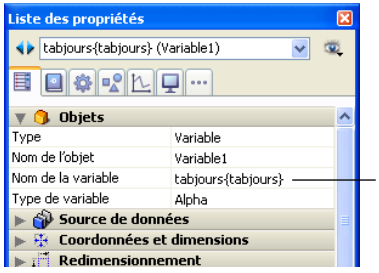
Le sous-formulaire en page utilise le formulaire entrée désigné par la propriété "Formulaire détaillé". A la différence d'un sous-formulaire en mode liste, le formulaire utilisé peut provenir de la même table que le

formulaire parent. Il est également possible d'utiliser un formulaire projet (cf. paragraphe "Formulaires projet", page 143).

En exécution, un sous-formulaire en mode page dispose des caractéristiques d'affichage standard d'un formulaire entrée. Les mécanismes des formulaires sortie (liés notamment à la gestion des taquets) ne sont pas activés.

Utilisation d'expressions comme noms de variables

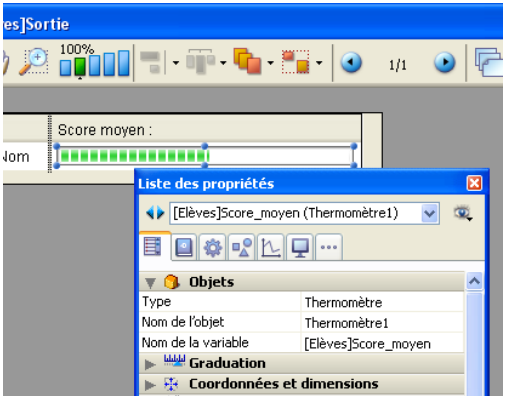
Les variables associées aux objets de formulaire peuvent désormais contenir tout type d'expression retournant une valeur — et non plus seulement un nom de variable. Il suffit pour cela de saisir directement l'expression dans la zone "Nom de variable" des propriétés de l'objet :

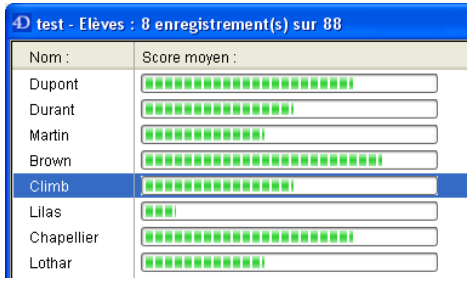


Expression utilisée comme nom de variable

Toute expression 4D valide est acceptée : expression simple, formule, fonction 4D, nom de méthode projet ou champ utilisant la syntaxe standard *[Table]Champ*. L'expression est évaluée lors de l'exécution du formulaire.

Les possibilités offertes par ce principe sont multiples. Il permet par exemple d'associer un champ numérique à un thermomètre afin d'afficher une valeur sous forme graphique en liste :

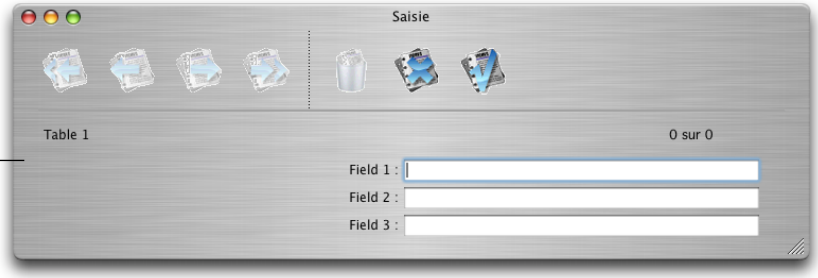




Si l'objet est saisissable et si l'expression le permet, il est également possible de saisir des valeurs via ce type d'objet.

Aspect "métal"

4D v11 vous permet de générer des fenêtres ayant l'apparence "métal" sous MacOS. Cette apparence est largement répandue dans l'interface Mac OS X :



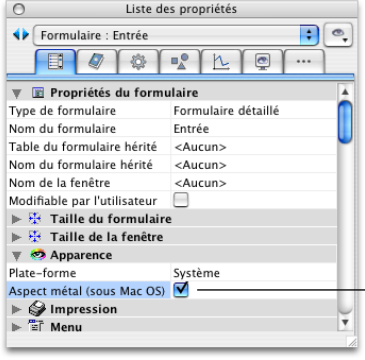
Fenêtre "métal" sous Mac OS

Pour créer des fenêtres "métal", vous disposez de deux solutions :

- utiliser la commande Creer fenetre formulaire avec la nouvelle propriété de formulaire **Aspect métal** (détaillée ci-dessous).
- utiliser la commande Creer fenetre avec la nouvelle constante Aspect métal. Cette possibilité est détaillée dans le paragraphe "Creer fenetre", page 343).

Propriété "Aspect métal sous Mac OS"

La nouvelle propriété de formulaire **Aspect métal (sous Mac OS)** du thème "Apparence" permet d'activer le mode d'apparence métal pour le formulaire lorsqu'il est affiché via la commande Creer fenetre formulaire sous Mac OS :



Propriété aspect métal

Note Sous Windows, cette propriété est sans effet.

Si le formulaire est affiché dans une fenêtre qui n'a pas été créée par la commande Creer fenetre formulaire (fenêtre du mode Développement par exemple), la propriété ne sera pas prise en compte.

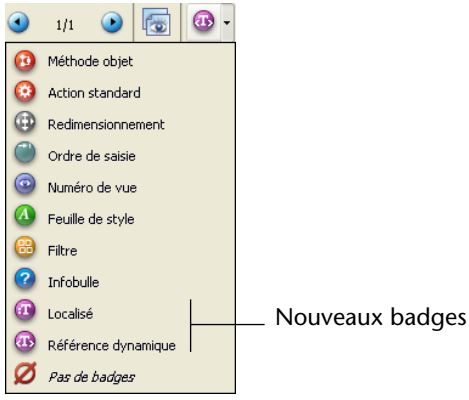
Sous Mac OS, l'effet métal est prévisualisé dans l'éditeur de formulaires lorsque l'option **Aspect métal** est cochée et lorsque les limites du formulaire sont affichées.

Nouvelles options de glisser-déposer

Le glisser-déposer a été modifié dans 4D v11. En particulier, les objets texte en contrôle natif (champs, variables, list box et combo box) disposent d'options supplémentaires pour la gestion du glisser et du déposer : **Glisser automatique** et **Déposer automatique** (thème "Action").
Ces nouvelles options et les nouveaux mécanismes mis en oeuvre sont détaillés dans le paragraphe "Glisser-Déposer", page 279 du chapitre "Langage".

Nouveaux badges

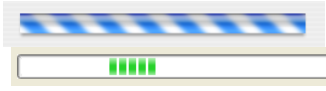
Deux nouveaux types de badges sont disponibles dans l'éditeur de formulaires :



- **Localisé** : ce badge permet de désigner les objets dont le libellé provient d'une référence (libellé débutant par ":"). La référence peut être de type ressource (STR#) ou XLIFF.
Pour plus d'informations sur les références XLIFF, reportez-vous au paragraphe "Prise en charge du standard XLIFF", page 101.
- **Référence dynamique** : ce badge permet de désigner les objets contenant une référence dynamique à un champ, une table ou une variable (syntaxe du type "<libellé>").

Thermomètre “Barber shop”

Vous pouvez utiliser des thermomètres de type “Barber shop” dans les formulaires 4D v11. Ce type de thermomètre affiche une animation continue. Les thermomètres “Barber shop” sont généralement utilisés pour indiquer à l’utilisateur que le programme est en train d’effectuer une opération longue.



Barber shop sous Mac OS

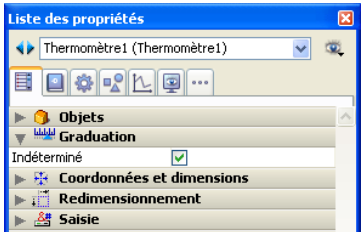
Barber shop sous Windows XP

► Pour ajouter un “Barber shop” dans un formulaire :

1 **Créez un objet “Thermomètre”.**

2 **Cochez la propriété de graduation “Indéterminé” dans la Liste des propriétés.**

L’activation de cette propriété masque les autres options de graduation des thermomètres.



A l’exécution du formulaire, le thermomètre n’est pas animé. Vous devez gérer l’animation en passant une valeur à la variable associée au thermomètre :

- 1 = démarrer l’animation,
- 0 = stopper l’animation.

Note Les thermomètres “Barber shop” fonctionnent uniquement avec l’apparence Système ou Impression.

La commande CHOIX FORMATAGE permet de passer dynamiquement d’un thermomètre “Barber shop” à une barre de progression standard.

Formats d'affichage des dates et des heures

La prise en charge des paramètres régionaux a été renforcée dans 4D v11 afin de faciliter le déploiement d'applications internationales.

Dans ce cadre, les formats d'affichage des dates et des heures ont été modifiés. Ces formats s'appuient désormais largement sur les paramètres du système et de nouveaux formats sont disponibles.

Note Pour compléter cette prise en charge des paramètres régionaux, un nouveau mécanisme gère les séparateurs décimaux et de milliers (cf. paragraphe "Préférences système pour les nombres", page 172) et la fonction Num a été modifiée dans le même esprit. Enfin, la nouvelle commande LIRE FORMATAGE SYSTEME permet d'exploiter de multiples paramètres régionaux définis dans le système.

Formats de date

Pour plus de clarté, les formats d'affichage des dates ont été renommés :

- Le nom des formats basés sur les formats définis dans système débute par "Système". Le résultat affiché par ces formats dépend de la configuration régionale du système.
- Le nom des formats basés sur des paramètres internes à l'application 4D débute par "Interne".

Le tableau suivant indique la correspondance entre les formats d'affichage des dates de 4D v11 et ceux des versions précédentes :

Nom format 4D v11	Exemple (système français)	Nom versions précédentes
Système date court	25/03/99	*Court*
Système date abrégé	jeu 25 mar 1999	*Abrégé*
Système date long	jeudi 25 mars 1999	*Long*
Interne date court spécial	25/03/99 mais 29/09/2039	*Spécial*
Interne date long	25 mars 1999	*Jour Mois Année*
Interne date abrégé	25 mar 1999	*Abrégé Jour Mois Année*
Interne date court	25/03/1999	*Spécial forcé*
ISO Date	1999-03-25T00:00:00	*ISO Date Heure*

Note Les constantes de formatage de 4D v11 reflètent ces modifications (cf. paragraphe "Constantes Format d'affichage", page 424).

- **Dates nulles** : par défaut, une date nulle est généralement affichée “00/00/00”. La nouvelle option **Vide si nulle** de la Liste des propriétés (thème “Affichage”) permet d’afficher une zone vide (“”) si la date est nulle ou contient l’attribut SQL NULL.

Formats d’heure

De nouveaux formats d’affichage des heures sont proposés dans 4D v11 (en plus des formats existants dans les versions précédentes). Ces formats sont détaillés dans le tableau suivant :

Nouveau format 4D v11	Description	Exemple pour 04:30:25
mn:s	Heure exprimée sous forme de durée depuis 00:00:00	270:25
Minutes:Seconde	Heure exprimée sous forme de durée depuis 00:00:00	270 Minutes 25 Secondes
ISO Heure	Partie “heure” du format composite ISO	0000-00-00T04:30:25
Système heure court	Format d’heure standard défini dans le système	04:30:25
Système heure long abrégé	*Mac OS uniquement* : Format d’heure abrégé défini dans le système Sous Windows, ce format équivaut au format Système heure court	4•30•25 AM
Système heure long	*Mac OS uniquement* : Format d’heure long défini dans le système Sous Windows, ce format équivaut au format Système heure court	4:30:25 AM HNEC

Note Les constantes de formatage de 4D v11 reflètent ces modifications (cf. paragraphe “Constantes Format d’affichage”, page 424).

- **Heures nulles** : une heure nulle est affichée par exemple “00:00:00” (l’affichage dépend du format appliqué à l’objet). La nouvelle option **Vide si nulle** de la Liste des propriétés (thème “Affichage”) permet d’afficher une zone vide (“”) si l’heure est nulle ou contient l’attribut SQL NULL.

Préférences système pour les nombres

Désormais, dans les bases de données 4D v11, les formats d’affichage numériques sont automatiquement basés sur les paramètres système régionaux. 4D substitue les caractères “.” et “,” dans les formats d’affichage des numériques par, respectivement, le séparateur des milliers et le séparateur décimal définis dans le système d’exploitation.

Le point et la virgule sont alors considérés comme des *caractères d'emplacement*, à l'instar du 0 ou du #.

Dans les versions précédentes de 4D, les formats d'affichage des numériques ne tenaient pas compte des paramètres régionaux du système. Par exemple, le format "###,##0.00" est un format valide pour un système américain. Toutefois, lorsqu'il est appliqué à une valeur numérique affichée sur un système français ou suisse, le résultat est incorrect.

Dans les bases de données converties, par souci de compatibilité, le nouveau mécanisme n'est pas actif. Pour en bénéficier, vous devez cocher l'option "**Formater les nombres selon les préférences système**", accessible dans la page Application/Compatibilité des Préférences. Le tableau suivant compare le résultat en fonction de l'état de l'option de compatibilité :

Valeur à afficher	15353,33	
Format d'affichage	###,##0.00	
Option "Formater les nombres selon les préférences système" dans les bases converties	**Non cochée (par défaut)**	**Cochée**
Affichage sur un système US *séparateur milliers = ,* *séparateur décimal = .*	15,353.33	15,353.33
Affichage sur un système français *séparateur milliers = <espace>* *séparateur décimal = ,*	<<<<<<<< (erreur)	15 353,33
Affichage sur un système suisse *séparateur milliers = '* *séparateur décimal = ,*	<<<<<<<< (erreur)	15'353,33

Les bases créées avec 4D v11 ont un fonctionnement équivalent à celui de l'option **Cochée**.

Note La commande Num accepte désormais un nouveau paramètre permettant de désigner un séparateur décimal spécifique (cf. routine Num, page 387). En outre, la nouvelle commande LIRE FORMATAGE SYSTEME permet de connaître la valeur courante des paramètres système régionaux.

8 Editeur de menus

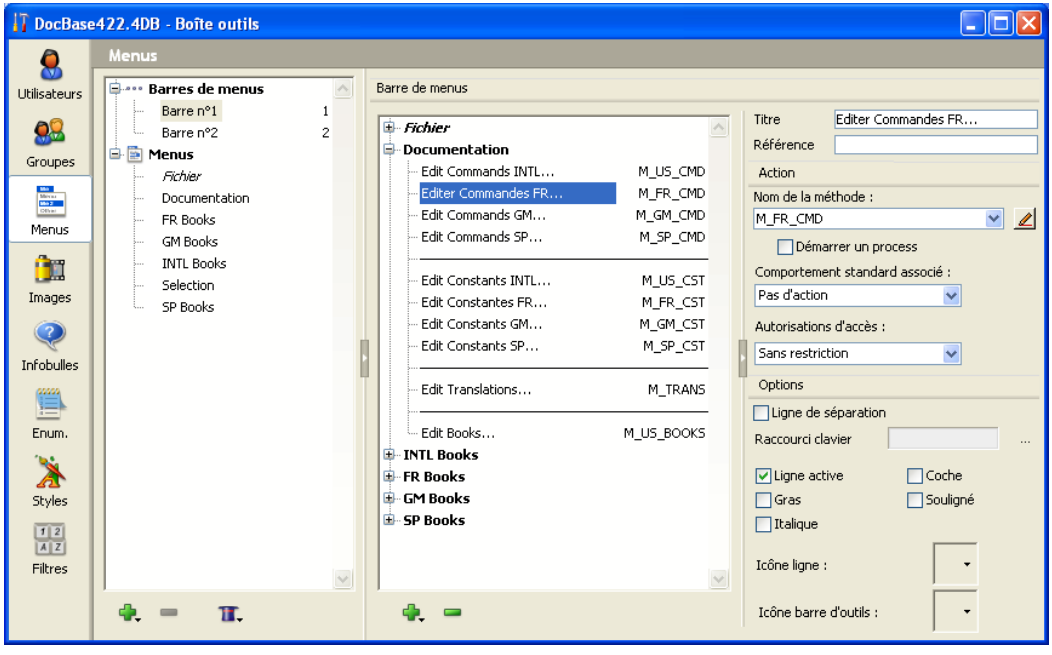
L'éditeur de menus a été modifié dans 4D v11 de manière à procurer davantage de souplesse pour la gestion des menus et des barres de menus. En outre, de nouvelles fonctions sont disponibles.

Note La gestion programmée des menus a également été modifiée. Pour plus d'informations, reportez-vous au paragraphe "Menus", page 294.

Interface

L'éditeur de menus est accessible via le bouton **Menus** de la Boîte à outils :

Nouvel éditeur de menus

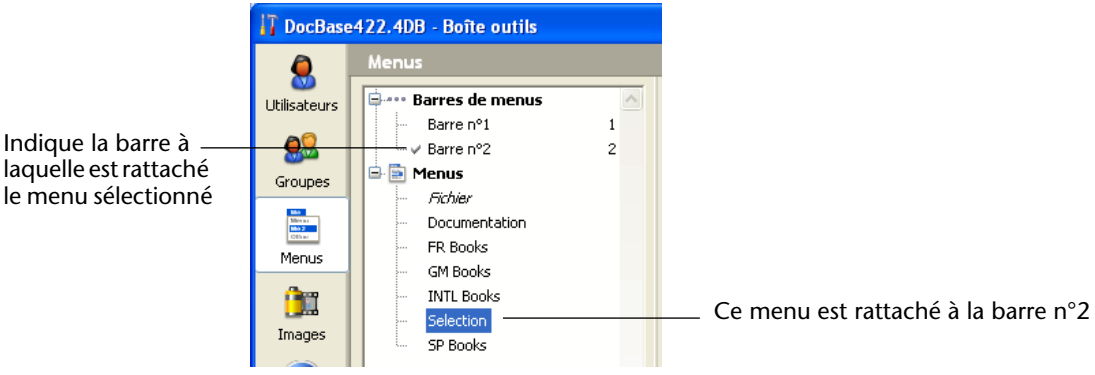


Les menus et les barres de menus sont désormais affichés sous forme de deux éléments d'une même liste hiérarchique, dans la partie gauche de la boîte de dialogue. Ce principe permet de visualiser d'un coup d'oeil tous les menus définis dans la base, sans qu'il soit nécessaire de sélectionner au préalable une barre de menus. Les menus sont classés par ordre alphabétique.
En outre, la gestion des menus attachés (menus utilisés à plusieurs emplacements) est facilitée car chaque menu source peut être facilement identifié.

Note Dans 4D v11, les menus "connectés" sont appelés menus "attachés".

Chaque menu peut être attaché à une barre de menus ou à un autre menu. Dans le deuxième cas, le menu devient un sous-menu (cf. paragraphe "Sous-menus hiérarchiques", page 180).

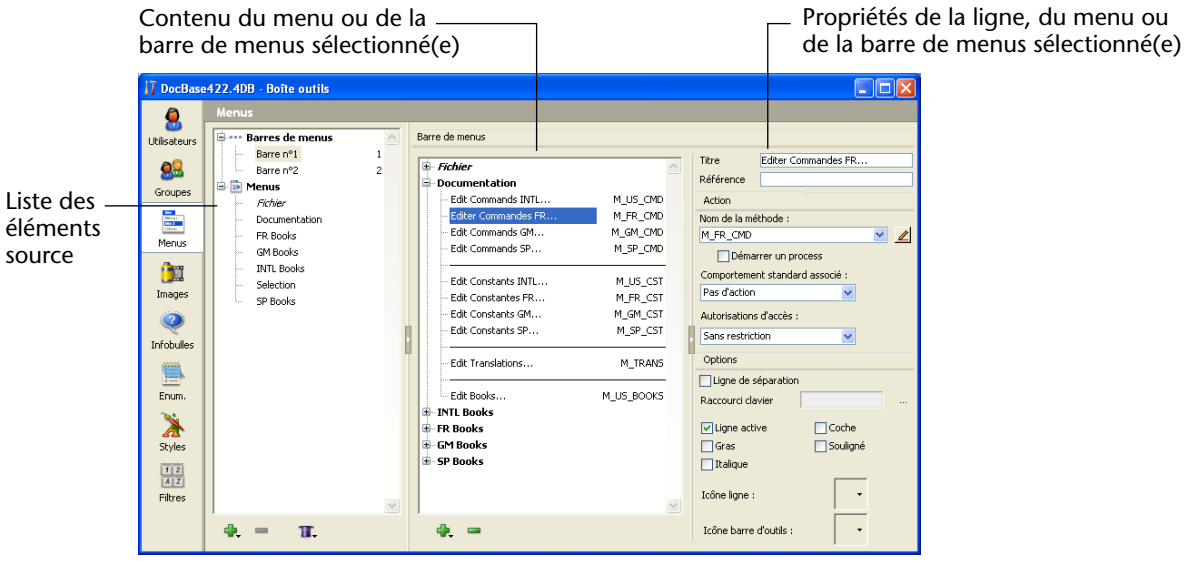
Une coche (√) indique l'élément (barre de menus ou menu) auquel appartient le menu sélectionné. Lorsqu'un menu est associé à plusieurs éléments, plusieurs coches sont affichées.



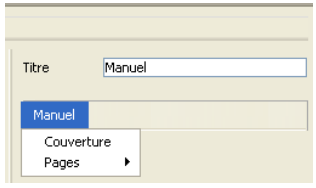
Si le menu n'est pas utilisé, aucune coche n'apparaît (cf. paragraphe "Menus indépendants", page 179).

Pour visualiser le contenu d'une barre de menus ou d'un menu, cliquez sur son libellé dans la liste de gauche de l'éditeur. La liste des éléments appartenant à la barre ou au menu s'affiche dans la zone centrale. Les propriétés de la barre ou du menu apparaissent également dans la zone

de droite de la fenêtre. Pour afficher les propriétés d'une ligne de menu, sélectionnez la ligne dans la partie centrale de la fenêtre.

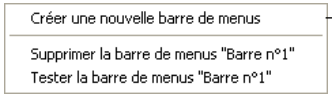


Lorsqu'un menu est sélectionné, la zone de droite permet également de prévisualiser le menu :



Menu contextuel et menu "assistant"

L'éditeur de menus comporte un menu contextuel permettant d'accéder directement aux actions possibles en fonction du type d'élément sur lequel le clic a eu lieu (barre, menu, ligne).

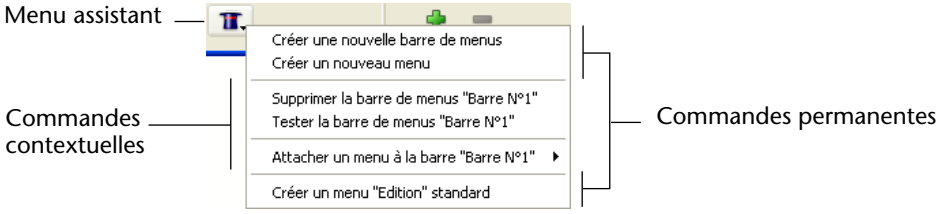


Menu contextuel en cas de clic droit sur une barre de menus

Le menu contextuel permet d'ajouter ou de supprimer un élément, de déployer ou contracter la liste, et propose des actions plus spécifiques.

L'éditeur de menus comporte également un menu "assistant" accessible lorsque vous cliquez sur le bouton en forme de chapeau,

situé au-dessous de la liste de gauche. Ce menu comporte des commandes permanentes et des commandes contextuelles :



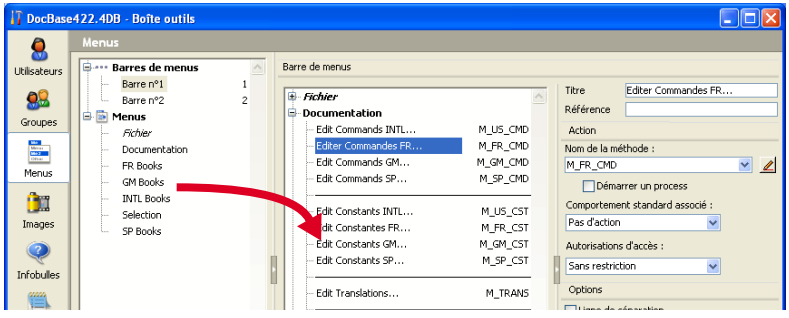
Les commandes permanentes vous permettent de créer une nouvelle barre ou un nouveau menu, ainsi qu'un menu **Edition** standard (cf. paragraphe "Menu Edition (compatibilité)", page 184). Les commandes contextuelles sont relatives à l'élément sélectionnée (barre ou menu) et proposent des actions de gestion appropriées.

Gestion des menus

Rattacher un menu à une barre de menus

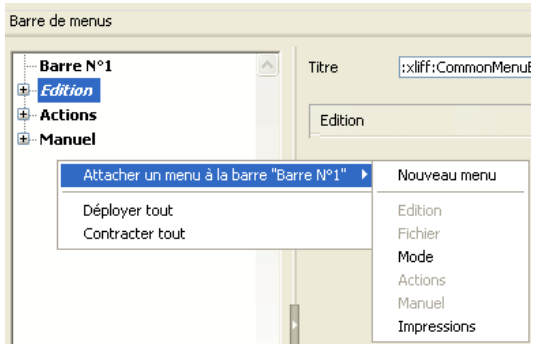
La connexion d'un menu à une barre de menus s'effectue par glisser-déposer, par le menu du bouton "assistant" ou par le menu contextuel de la zone centrale.

- **Utiliser le glisser-déposer** : cliquez sur une barre de menus afin d'afficher son contenu dans la liste centrale ; sélectionnez un menu dans la liste de gauche et faites-le glisser à l'emplacement désiré dans la liste centrale :

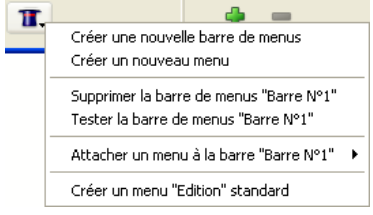


- **Utilliser le menu contextuel** : cliquez sur une barre de menus afin d'afficher son contenu dans la liste centrale ; cliquez avec le bouton droit dans cette zone et sélectionnez la commande **Attacher un menu à**

la barre “nom barre” > puis choisissez le menu à utiliser comme sous-menu :

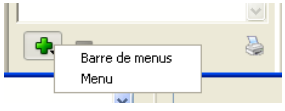


- **Utilliser le menu “assistant”** : sélectionnez une barre de menus dans la liste de gauche puis cliquez surle bouton “Assistant” situé au-dessous de la liste ; sélectionnez la commande **Attacher un menu à la barre “nom barre”** > puis choisissez le menu à utiliser comme sous-menu :



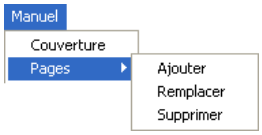
Menus indépendants

Il est désormais possible de définir des menus “indépendants”, c’est-à-dire non rattachés à une barre ou à un autre menu spécifique. Ces menus peuvent être configurés dans l’éditeur de menus mais devront être gérés via les commandes du langage.
Pour créer un menu indépendant, sélectionnez la commande **Menu** dans le menu associé au bouton de création sous la liste des barres de menus / menus :



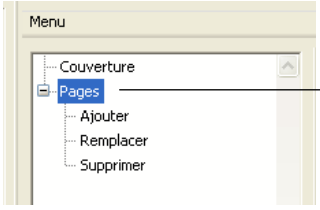
Sous-menus hiérarchiques

Dans 4D v11, il est possible de définir des sous-menus hiérarchiques. Dans une barre de menus, les sous-menus permettent de regrouper des fonctions thématiques à l'intérieur d'un même menu :



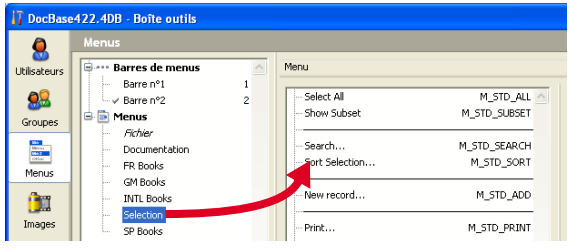
Les sous-menus et leurs lignes peuvent disposer des mêmes attributs que les menus (actions, méthodes, raccourcis, icônes, etc.)

Dans l'éditeur de menus, les sous-menus apparaissent sous forme d'éléments de liste hiérarchique :



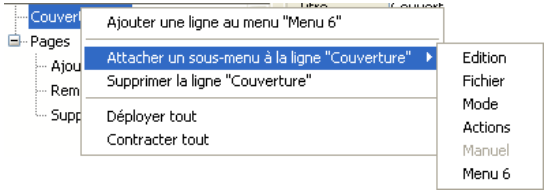
► Pour créer un sous-menu, il suffit d'associer (rattacher) un menu existant à une ligne d'un autre menu. Vous disposez de deux possibilités :

- **Utiliser le glisser-déposer** : sélectionnez un menu dans la liste de gauche et faites-le glisser dans la liste centrale sur la ligne à laquelle rattacher le sous-menu :



- **Utilliser le menu contextuel** : dans la liste centrale, cliquez avec le bouton droit sur la ligne à laquelle rattacher le menu. Dans le menu con-

textuel, sélectionnez la commande **Attacher un sous-menu à la ligne “nom ligne”** > puis choisissez le menu à utiliser comme sous-menu :



Le menu rattaché devient alors sous-menu. Le libellé de la ligne est conservé (le nom d’origine du sous-menu est ignoré), mais ce libellé peut être modifié.

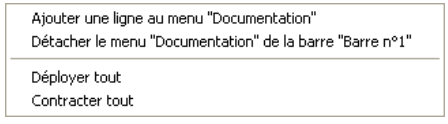
Les lignes du sous-menu conservent leurs caractéristiques et leurs propriétés, le fonctionnement du sous-menu est identique à celui d’un menu standard.

Vous pouvez créer des sous-menus de sous-menus. Il suffit pour cela de déployer le sous-menu dans la liste centrale et de rattacher un menu à une sous-ligne. Vous pouvez ajouter des sous-menus sur une profondeur virtuellement illimitée. A noter toutefois que pour des raisons d’ergonomie d’interface, il n’est généralement pas conseillé de dépasser deux niveaux de sous-menus.

Détacher un menu ou un sous-menu

Vous pouvez à tout moment détacher un menu d’une barre ou un sous-menu d’un menu. Le menu détaché n’est alors plus disponible dans la barre ou le sous-menu, mais reste présent dans la liste des menus.

Pour détacher un menu, cliquez avec le bouton droit dans la liste centrale sur le menu ou le sous-menu à détacher puis choisissez la commande **Détacher le menu “nom menu” de la barre “nom barre”** ou **Détacher le sous-menu de la ligne “nom ligne”** :



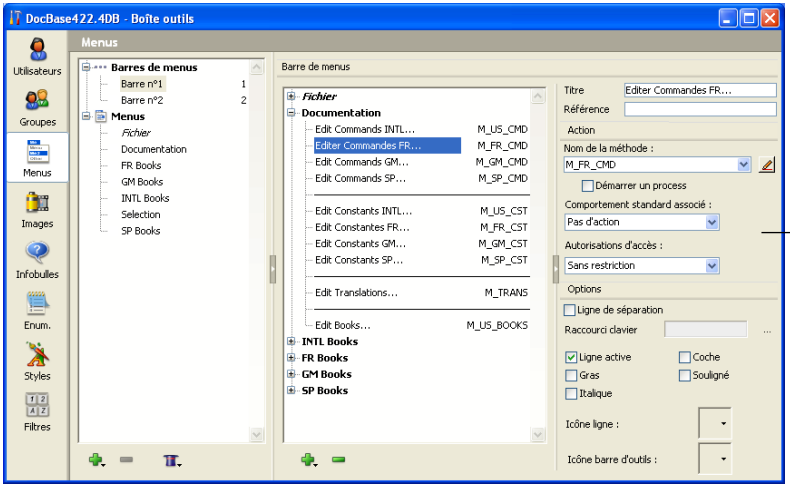
Libellés en référence

Comme dans les versions précédentes de 4D, il est possible de définir des commandes de menus par référence (libellés affichés en italique dans l’éditeur). 4D v11 reste compatible avec les libellés en ressources mais la notation XLIFF est désormais recommandée (cf. paragraphe “Prise en charge du standard XLIFF”, page 101).

Seule la référence du menu est stockée, le libellé est affiché en fonction de la langue courante de l'application. Ce principe facilite la diffusion d'applications avec des interfaces multi-langues.

Nouvelles propriétés

De nouvelles options et propriétés sont disponibles pour les menus et les lignes de menus. Ces propriétés sont affichées dans la partie droite de l'éditeur lorsqu'un élément est sélectionné dans la zone centrale :



Zone de saisie et d'affichage des propriété

Référence de ligne de menu

Il est possible d'associer une référence personnalisée à chaque ligne de menu. Une référence de ligne de menus est une chaîne de caractères dont le contenu est libre.

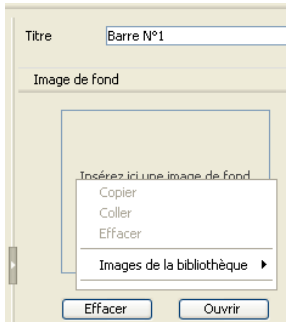
Les références de lignes de menus sont principalement utiles pour la gestion programmée des menus, en particulier lors de l'utilisation de la nouvelle commande Pop up menu dynamique.

Image de fond native

Il est désormais possible de désigner une image de fond dans son format natif. Pour cela, sélectionnez une barre de menus dans la liste de gauche puis cliquez sur le bouton **Ouvrir**, la boîte de dialogue standard d'ouverture de fichiers propose de nombreux types natifs pour l'image de fond : JPEG, PNG, GIF, etc.

Note La prise en charge de formats d'image natifs dans 4D v11 est détaillée dans le paragraphe "Optimisation des variables et champs image", page 160.

Il est également possible de sélectionner une image de fond provenant de la bibliothèque d'images de 4D via le menu contextuel :



Actions standard

Les actions proposées dans le pop up menu "Action standard associée" ont été modifiées, principalement en regard du nouveau fonctionnement du mode Développement (cf. paragraphe "Interface du mode Développement", page 75).

- **Retour au mode Développement** : fait passer au premier plan les fenêtres et la barre de menus du mode Développement de 4D.
  Cette action remplace les deux anciennes actions Utilisation et Structure. Lorsque la base est exécutée en mode interprété, cette action provoque l'affichage de la fenêtre courante du mode Développement. Lorsque la base est exécutée en mode compilé, cette action provoque l'affichage de la fenêtre des enregistrements de la table courante (en mode compilé, seul l'accès aux enregistrements est possible).
- **Application** : fait passer au premier plan les fenêtres et la barre de menus du mode Test application de 4D. Cette action remplace strictement l'ancienne action Menus créés.
- **CSM** : affiche la fenêtre du Centre de Sécurité et de Maintenance.

Actions standard dans les bases converties

Dans les bases de données converties, les actions standard Utilisation et Structure sont conservées dans les barres de menus existantes, mais elles ne sont pas sélectionnables dans le pop up menu "Comportement standard associé". L'action Menus créés est renommée Tester l'application.
Le fonctionnement des actions Utilisation et Structure est identique à

celui des versions précédentes, à la différence près que l'action Utilisation provoque désormais l'affichage de la fenêtre de liste des enregistrements du mode Développement.

Coche

Cette nouvelle option permet d'associer par défaut une coche système (√) à la ligne de menu. Vous pourrez ensuite gérer masquer ou afficher la coche au moyen des commandes du langage (FIXER MARQUE LIGNE MENU et Lire marque ligne menu).

Icône ligne

La nouvelle option **Icône ligne** vous permet d'associer une icône de menu à la ligne sélectionnée :

Une fois définie, cette icône est affichée directement dans le menu, en regard de la ligne :

Menu Edition (compatibilité)

L'option **v 6.8** n'est plus disponible pour les menus dans 4D v11. Cette option permettait, dans les bases de données converties, de maintenir un ancien fonctionnement, basé sur l'ajout automatique d'un menu **Edition** géré par le système.

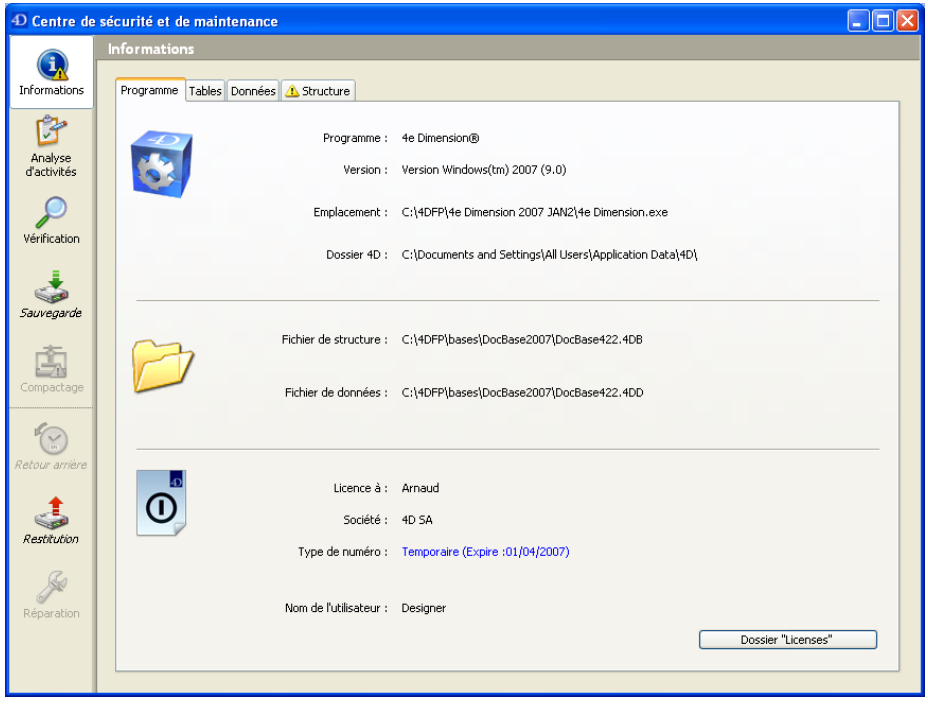
Désormais, le menu **Edition** doit être entièrement géré comme les autres menus. L'assistant de conversion ajoute automatiquement un menu **Edition** standard aux barres de menus :

- des bases de données converties depuis une version 6.8,
- des bases de données converties depuis une version 200x et comportant l'option **v 6.8**.

Une boîte de dialogue vous indique les barres de menus ayant été modifiées. Notez bien les barres concernées car vous devez incrémenter la valeur du paramètre *menu* dans les instructions de gestion et d'exécution de ces barres de menus. En effet, avec l'ancien fonctionnement, le menu **Edition** n'était pas comptabilisé.

9 Centre de Sécurité et de Maintenance

4D v11 inclut une nouvelle fenêtre qui rassemble tous les outils nécessaires au contrôle, à la maintenance, à la sauvegarde et au compactage des fichiers de données et de structure. Cette nouvelle fenêtre est appelée Centre de sécurité et de maintenance (CSM) :



Le panneau de navigation situé sur la gauche de la fenêtre comporte des boutons vous permettant de sélectionner le type de fonction ou d'information auquel vous souhaitez accéder.

La fenêtre du CSM centralise des fonctions disséminées dans différents environnements dans les versions précédentes de 4D : boîte de dialogue d'A propos, boîte de dialogue d'ouverture, gestionnaire de sauvegarde et application 4D Tools.

Note sur 4D Tools Le CSM contient des fonctions auparavant présentes dans l'application 4D Tools. Du fait de cette intégration, 4D Tools n'est plus nécessaire en tant qu'application indépendante et n'existe plus en version 11.

Le CSM est disponible dans toutes les applications 4D : 4D monoposte, 4D Server, 4D Runtime interpreted, 4D Runtime Single User et 4D Runtime Volume Licence.

Affichage du CSM

Vous pouvez accéder à la fenêtre du CSM de plusieurs manières. Le mode d'accès détermine également le mode d'ouverture de la base : mode "maintenance" ou mode "standard". En mode maintenance, la base n'est pas ouverte par 4D, seule sa référence est fournie au CSM. En mode standard, la base est ouverte par 4D.

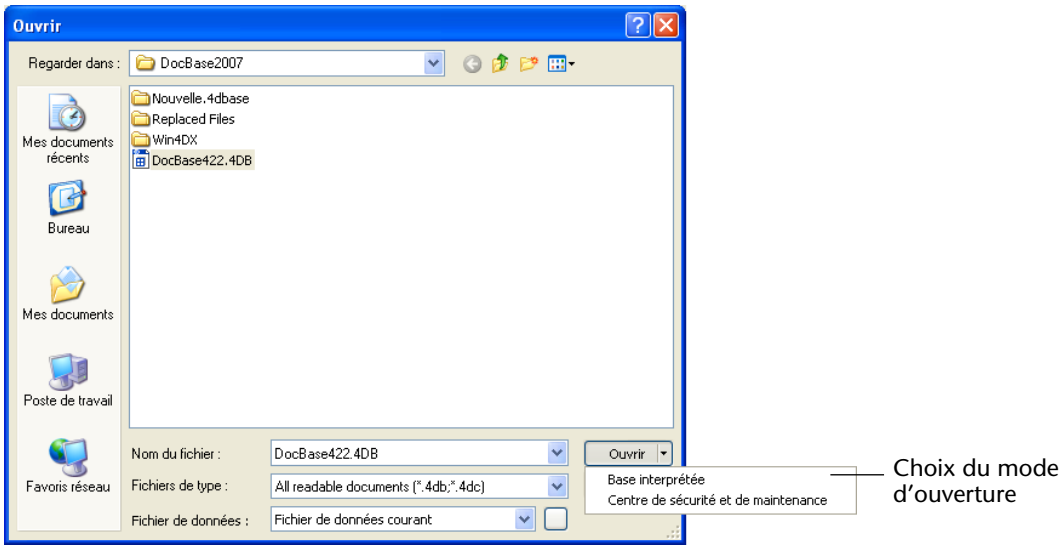
Affichage en mode maintenance

En mode maintenance, seule la fenêtre du CSM est affichée (la base n'est pas ouverte par l'application 4D). Ce principe permet notamment d'accéder à des bases trop endommagées pour pouvoir être ouvertes en mode standard par 4D. En outre, certaines opérations (compactage, réparation...) nécessitent que la base soit ouverte en mode maintenance (cf. paragraphe "Restrictions d'accès", page 188).

Vous pouvez ouvrir le CSM en mode maintenance depuis deux emplacements :

- **Boîte de dialogue standard d'ouverture**
  La boîte de dialogue standard d'ouverture de base de données

comporte l'option **Centre de sécurité de maintenance** sous forme de menu associé au bouton **Ouvrir** :



Il vous suffit alors de désigner la base à examiner puis de cliquer sur le bouton **Ouvrir**.

- **Menu Aide ou bouton CSM de la barre d'outils (base non ouverte)** Lorsque vous appelez cette fonction, une boîte de dialogue standard d'ouverture de fichiers apparaît, vous permettant de désigner la base à examiner.

Note Dans 4D v11, il est possible de lancer l'application sans qu'aucune base de données ne soit ouverte (cf. paragraphe "Ouverture et création des bases de données", page 72).

Affichage en mode standard

En mode standard, une base de données est ouverte. Dans ce mode, certaines fonctions de maintenance ne sont pas disponibles. Vous disposez de plusieurs possibilités :

- **Menu Aide ou le bouton CSM de la barre d'outils** en mode Développement.
  Cette commande provoque l'affichage de la fenêtre du CSM. Cette fonction n'est pas accessible en mode Application.
- **Via l'action standard "CSM"** qu'il est possible d'associer à une commande de menu créé.
- **A l'aide de la nouvelle commande** OUVRIR CENTRE DE SECURITE.

Restrictions d'accès

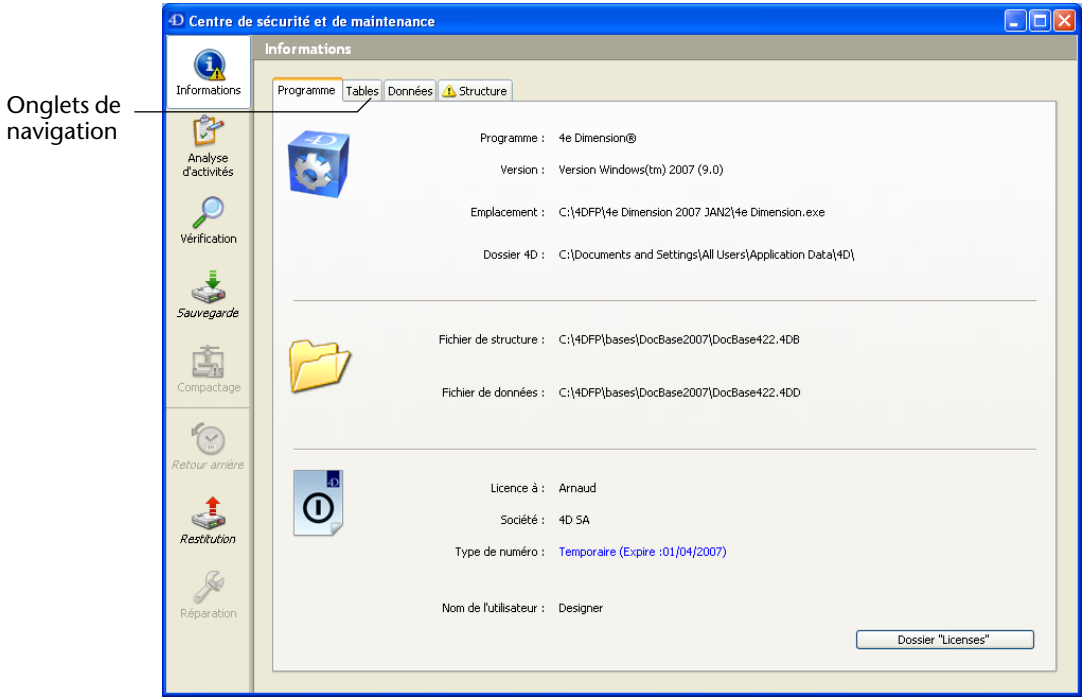
Certaines fonctions du CSM ne sont pas disponibles suivant le type d'application, le mode d'ouverture du CSM ou (lorsque les mots de passe sont activés) le profil d'utilisateur :

- les fonctions qui agissent sur la structure de l'application (vérification, réparation et compactage) sont accessibles uniquement depuis les applications 4D et 4D Server. Dans les applications 4D Client et 4D Desktop, les boutons et onglets correspondants sont masqués.
- les informations relatives au contenu des fichiers de données et de structure sont disponibles lorsque la bases de données est ouverte uniquement (le CSM doit avoir été ouvert en mode standard).
- les fonctions de compactage, retour arrière, restitution et réparation des données ne sont utilisables qu'avec des bases de données non ouvertes (le CSM doit avoir été ouvert en mode maintenance). Si ces fonctions sont sollicitées alors que la base est ouverte en mode standard, une boîte de dialogue s'affiche, vous permettant de relancer l'application en mode maintenance.
- si les mots de passe sont activés, les fonctions de compactage, retour arrière, restitution et réparation des données ne sont accessibles que pour l'Administrateur et le Super_Utilisateur.

Page Informations

La page "Informations" fournit diverse informations relatives à l'environnement 4D et l'environnement système, la base de données et

les fichiers de l'application. Chaque page d'information est accessible via des onglets situés dans la partie supérieure de la fenêtre :

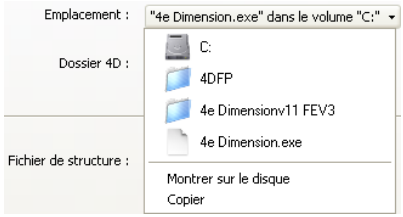


Programme et Tables

Les deux premières pages fournissent des informations d'environnement qui étaient auparavant présentes dans la boîte de dialogue d'**A propos** de 4D :

- **Programme** : indique le nom, la version et l'emplacement de l'application ainsi que du Dossier 4D ; la partie centrale de la fenêtre indique le nom et l'emplacement des fichiers de structure de données de la base. La partie inférieure de la fenêtre indique le nom du détenteur de la licence 4D, le type de licence, ainsi que le nom de l'utilisateur de la base lorsque les mots de passe sont activés.
  Le bouton **Dossier "Licenses"** affiche le contenu du dossier Licenses actif dans une nouvelle fenêtre système.
  - **Affichage et sélection des chemins d'accès** : dans la page **Programme**, les chemins d'accès sont affichés sous forme de pop up

menus contenant l'enchaînement des dossiers à partir du disque :



Si vous sélectionnez un élément du menu (disque ou dossier), il s'afficher dans une nouvelle fenêtre système.
La commande **Montrer sur le disque** affiche le fichier ou dossier cible dans une nouvelle fenêtre système.
La commande **Copier** copie le chemin d'accès complet sous forme de texte dans le Presse-papiers.

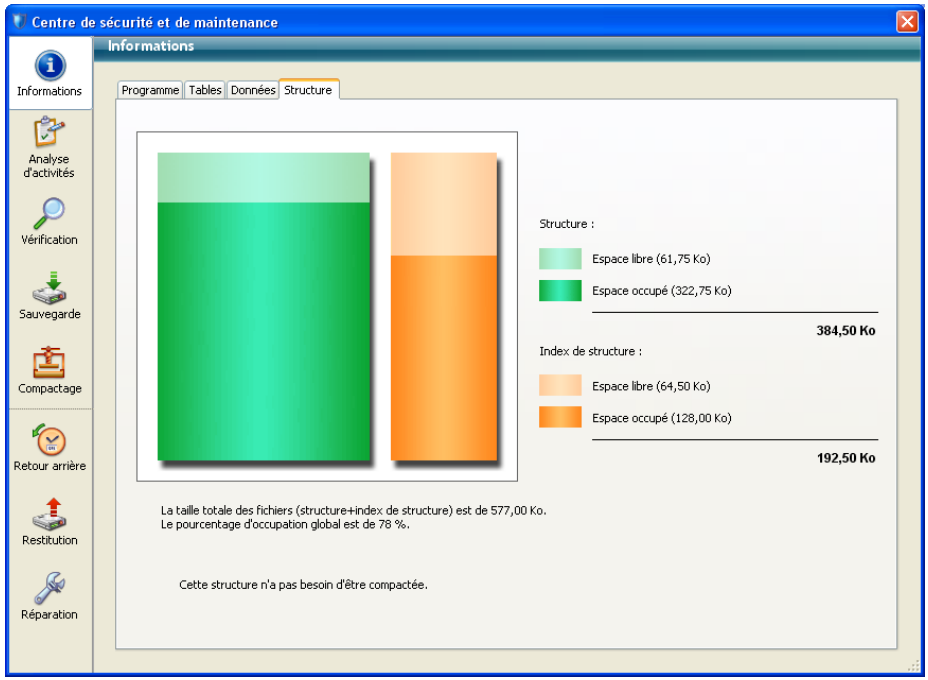
- **Tables** : liste toutes les tables de la base ainsi que leurs caractéristiques (numéro de chaque table, nombre total d'enregistrements, de champs et d'index pour la table).

Données et Structure

Les pages **Données** et **Structure** fournissent des informations sur le remplissage et l'occupation des fichiers de données et de structure de la base.

Notes
- Ces pages ne sont pas accessibles en mode Maintenance.
- La page **Structure** est disponible uniquement dans les applications 4D et 4D Server.

Ces informations sont fournies sous forme de valeurs en octets, Ko, Mo ou Go (le menu située en bas à droite permet de définir l'unité d'affichage) et sont également représentées sous forme graphique :

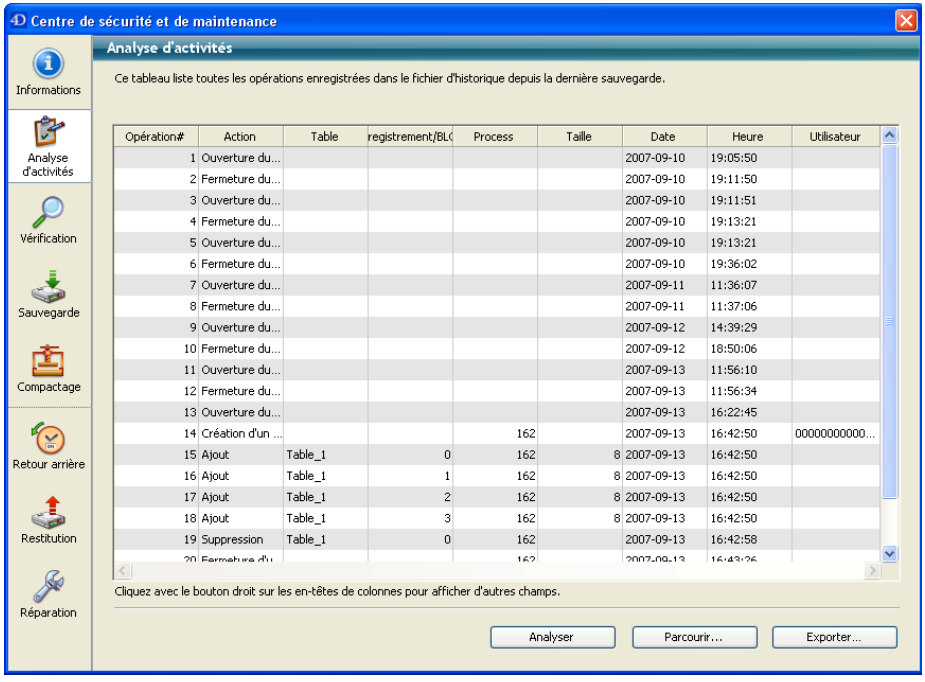


Des fichiers trop fragmentés réduisent les performances du disque dur et donc de la base. Si le taux d'occupation est trop faible, 4D vous le signale par une icône d'avertissement (qui apparaît dans le bouton **Informations** et dans l'onglet du type de fichier correspondant) et indique qu'un compactage est requis :

Une icône d'avertissement est également affichée sur le bouton de la page **Compactage** (cf. paragraphe "Compactage", page 197) :

Page Analyse d'activité

Cette page permet de visualiser les opérations présentes dans le fichier d'historique.



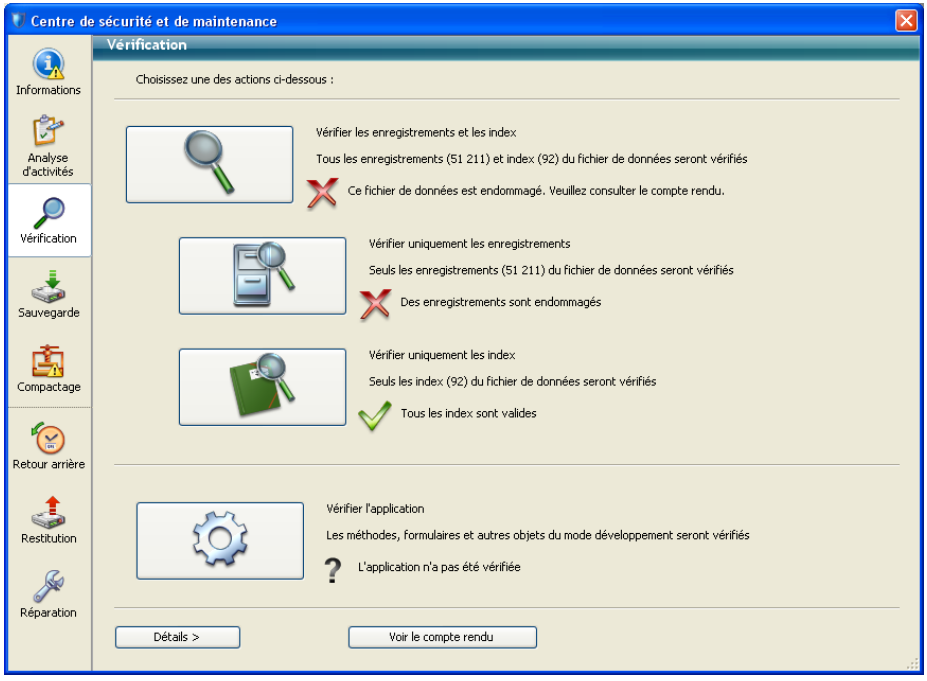
Cliquez sur le bouton **Analyser** pour afficher le contenu du fichier d'historique courant de la base sélectionnée (nommé par défaut *nomdonnées.journal* dans 4D v11).
Le bouton **Parcourir...** vous permet de sélectionner et d'ouvrir un autre fichier d'historique de la base.
Le bouton **Exporter...** vous permet d'exporter le contenu du fichier sous forme de texte.

Cette page reprend les informations présentes dans la boîte de dialogue "Vérification de l'historique" de 4D : liste des opérations de base de données enregistrées dans le fichier d'historique (ajout, modification ou suppression d'enregistrements, ainsi que les transactions).

En outre, la date et l'heure d'ouverture et de fermeture du fichier de données sont également affichées.

Page Vérification

Cette page permet de vérifier l'intégrité des données et de la structure. La vérification peut porter sur les enregistrements et/ou les index ainsi que sur les objets du développement (méthodes, formulaires...).



Note Cette fonction effectue uniquement une vérification des objets. Si des erreurs sont trouvées et des réparations requises, il vous sera nécessaire d'utiliser la page **Réparation** (voir ci-dessous).

Actions

La page comporte quatre boutons d'action permettant un accès direct aux fonctions de vérification :

- **Vérifier les enregistrements et les index** : lance la procédure de vérification globale des données.
- **Vérifier uniquement les enregistrements** : lance la procédure de vérification des enregistrements uniquement (les index ne sont pas vérifiés).
- **Vérifier uniquement les index** : lance la procédure de vérification des index uniquement (les enregistrements ne sont pas vérifiés).

Note La vérification des enregistrements et des index peut également être effectuée en mode détaillé, table par table (cf. paragraphe “Détails”, page 194).

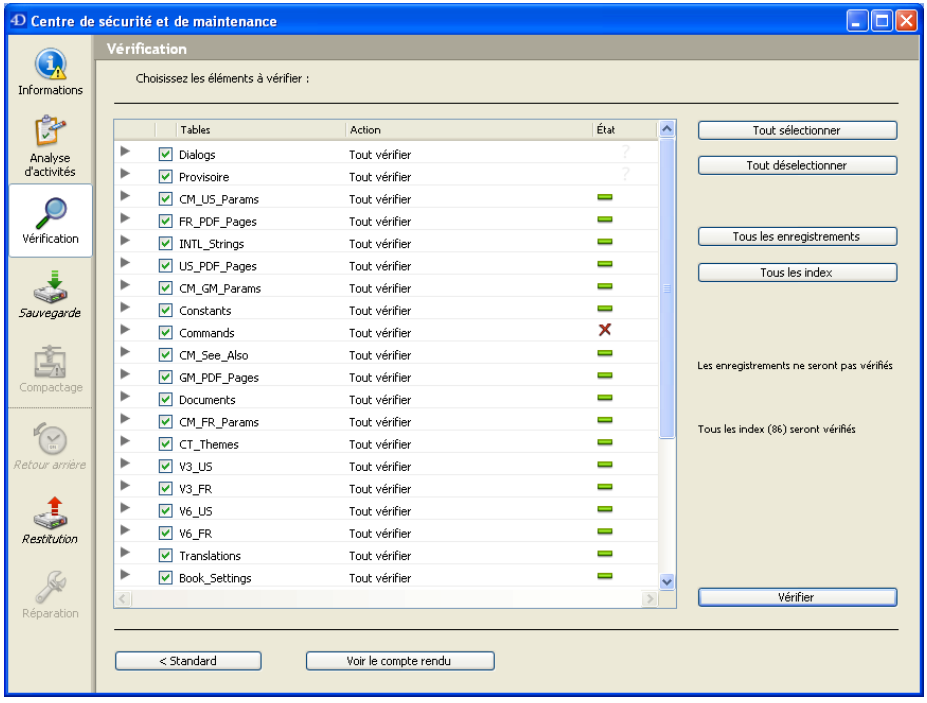
- **Vérifier l’application** : lance la procédure de vérification de tous les objets définis dans le mode Développement (tables, méthodes, formulaires...).

Voir le compte rendu

Quelle que soit la vérification demandée, 4D génère un fichier compte-rendu dans le dossier de la base. Ce fichier est créé au format html et permet de visualiser l’ensemble des vérifications effectuées.
Lorsque vous cliquez sur le bouton **Voir le compte rendu**, 4D affiche le contenu du dossier de la base dans une nouvelle fenêtre et sélectionne le fichier de compte-rendu.

Détails

Le bouton **Détails** provoque l’affichage d’une page détaillée permettant de visualiser et de sélectionner les enregistrements et les index à vérifier :



La désignation des éléments à vérifier permet notamment de gagner du temps lors de la vérification.

La liste principale affiche toutes les tables de la base. Pour chaque table, vous pouvez limiter la vérification aux enregistrements et/ou à chaque index. Cliquez sur l'icône en forme de triangle pour déployer le contenu d'une table ou les index d'un champ et sélectionnez/désélectionnez les cases à cocher en fonction de vos souhaits. Par défaut, tout est sélectionné.
Vous pouvez également utiliser les boutons raccourcis **Tout sélectionner**, **Tout désélectionner**, **Tous les enregistrements** et **Tous les index**.

Pour chaque ligne de table, la colonne "Action" résume les opérations à effectuer. Lorsque la table est déployée, les lignes "Enregistrements" et "Champs indexés" indiquent le nombre d'éléments concernés.

La colonne Etat affiche le statut de la vérification de chaque élément à l'aide de symboles :

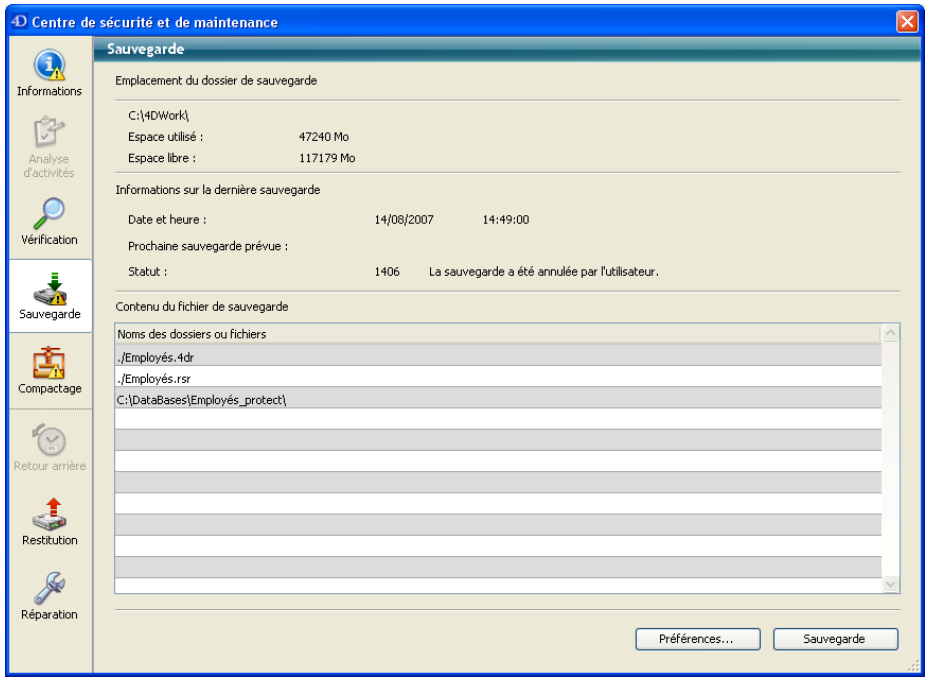
— Vérification effectuée, pas de problème
— Vérification effectuée, problèmes rencontrés
— Vérification partielle effectuée
— Vérification non effectuée

Cliquez sur le bouton **Vérifier** pour lancer la vérification ou sur le bouton **<Standard** pour retourner à la page standard.

Note La page standard ne tient pas compte des modifications effectuées dans la page détaillée : lorsque vous cliquez sur un bouton de vérification dans la page standard, tous les éléments sont vérifiés. En revanche, les paramétrages effectués dans la page détaillée sont conservés d'une session à l'autre.

Sauvegarde

La page **Sauvegarde** permet de visualiser les paramètres de sauvegarde de la base et de lancer une sauvegarde manuelle :



Cette page ne permet pas de modifier les paramètres de sauvegarde. Pour cela, vous devez cliquer sur le bouton **Préférences...**

- **Emplacement du fichier de sauvegarde**
  Cette zone affiche les informations relatives à l'emplacement du fichier de sauvegarde de la base. Elle indique également l'espace occupé et l'espace disponible sur le volume de sauvegarde.
- **Informations sur la dernière sauvegarde**
  Cette zone fournit la date et l'heure de la dernière sauvegarde (automatique ou manuelle) réalisée sur la base.
- **Contenu du fichier de sauvegarde**
  Cette zone liste les fichiers et dossiers inclus dans le fichier de sauvegarde.

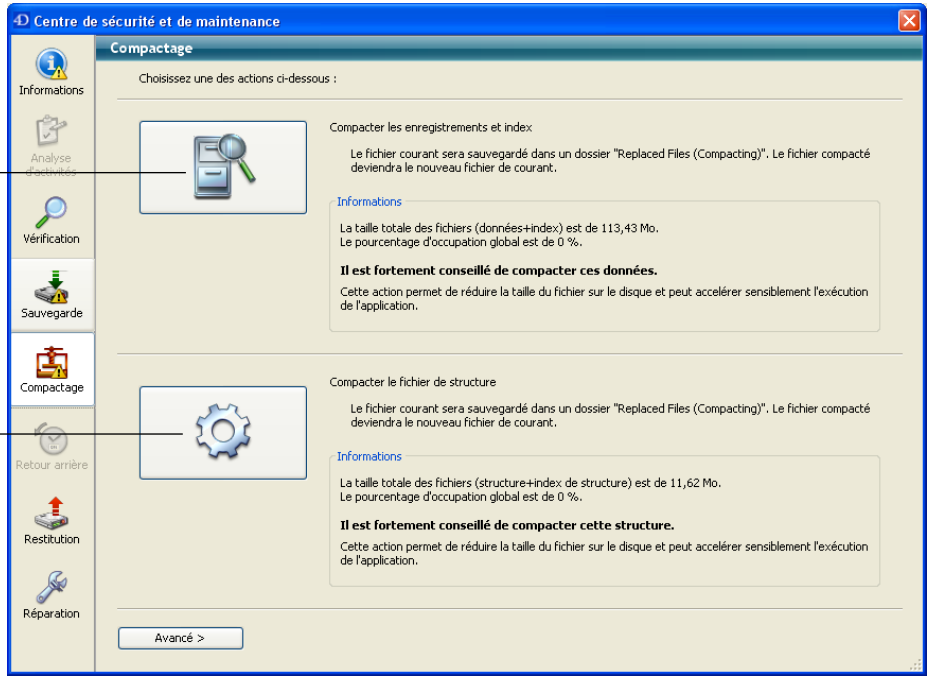
Le bouton **Sauvegarde** permet de lancer une sauvegarde manuelle.

Compactage

Cette page permet d'accéder aux fonctions de compactage du fichier de données et de structure.

Compactage du fichier de données

Compactage du fichier de structure



Pourquoi compacter ?

Les fichiers peuvent comporter des emplacements inutilisés (des "trous"). En effet, lorsque vous supprimez des enregistrements, des formulaires, etc., l'emplacement qu'ils occupaient précédemment dans le fichier devient vacant. 4D réutilise ces emplacements vides lorsque c'est possible, mais la taille des données étant variable, les suppressions ou modifications successives génèrent inévitablement des espaces inutilisables pour le programme. Il en va de même lorsqu'une grande quantité de données vient d'être supprimée : les emplacements vides restent inaffectés dans le fichier.

Le rapport entre la taille du fichier de données et l'espace réellement utilisé pour les données est le taux *d'occupation* des données. Un taux trop faible peut entraîner, outre un gaspillage de place, une dégradation des performances de la base. La fonction de compactage permet de réorganiser et d'optimiser le stockage des données afin de faire disparaître les "trous".

La zone “Informations” résume les données relatives à la fragmentation des fichiers et indique les opérations nécessaires. La page “Informations” du CSM détaille la fragmentation courante des fichiers de la base (cf. paragraphe “Page Informations”, page 188).

Note Le compactage n’est disponible qu’en mode maintenance. Si vous tentez d’effectuer cette opération en mode standard, une boîte de dialogue d’alerte vous prévient que la base va être fermée puis relancée en mode maintenance.
Il est cependant possible de compacter un fichier de données non ouvert par la base de données (cf. paragraphe “Compacter des fichiers de données en mode avancé”, page 200).

Compacter le fichier de données ou de structure

La procédure de compactage standard du fichier de données et de structure est identique.

Pour démarrer directement le compactage du fichier de données ou de structure, cliquez sur le bouton correspondant dans la fenêtre du CSM :

Données Structure

Note Le compactage incluant la duplication du fichier d’origine, le bouton est désactivé si la place sur le disque contenant le fichier est insuffisante.

Cette opération défragmente le fichier principal ainsi que les éventuels fichiers d’index.
4D effectue une copie des fichiers d’origine et les place dans un dossier nommé **Replaced Files (Compacting)**, créé à côté du fichier d’origine. A l’issue de l’opération, les fichiers défragmentés remplacent automatiquement les fichiers d’origine. La base de données est immédiatement opérationnelle.

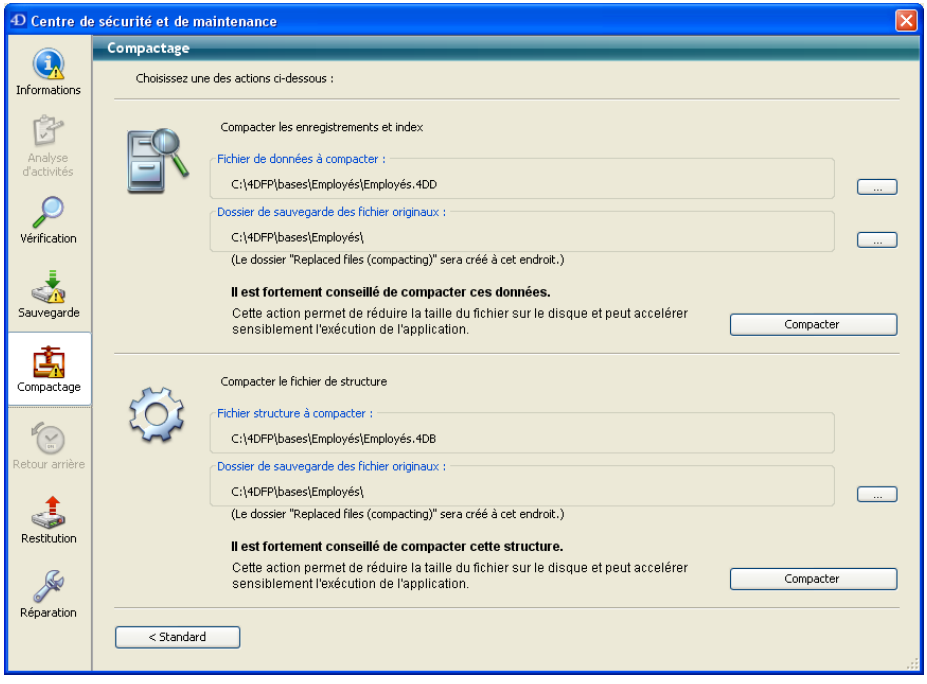
Notes
- Vous pouvez modifier ce fonctionnement par défaut via le mode avancé (cf. page suivante).
- Si vous effectuez plusieurs compactages, un nouveau dossier est créé à chaque fois. Il est nommé “Replaced Files (Compacting)_1”, “Replaced Files (Compacting)_2”, etc.

Chaque compactage entraîne la duplication du fichier d'origine et donc l'augmentation de la taille du dossier de l'application. Il est important de tenir compte de ce mécanisme (notamment sous Mac OS où les applications 4D apparaissent sous forme de* packages*) pour que l'application ne grossisse pas de façon excessive. Une intervention manuelle à l'intérieur du package permet peut être utile afin de supprimer les copies des fichiers d'origine.

A l'issue du compactage, 4D génère un fichier compte-rendu dans le dossier de la base. Ce fichier, nommé *nombase_compact_log*, est créé au format xml et html. Il permet de visualiser l'ensemble des opérations effectuées.
Lorsque vous cliquez sur le bouton **Voir le compte rendu**, 4D affiche le contenu du dossier de la base dans une nouvelle fenêtre ainsi que le fichier de compte-rendu html.

Mode avancé

La page Compactage comporte un bouton **Avancé>**, permettant d'accéder à une page d'options pour le compactage des fichiers de données et de structure :



La partie supérieure de la page affiche le chemin d'accès du fichier de données courant ainsi qu'un bouton permettant de désigner un autre fichier de données. La partie inférieure affiche le chemin d'accès du fichier de structure de la base ainsi qu'une option de compactage. Lorsque vous cliquez sur l'un des boutons **Compacter**, l'opération débute immédiatement.

Compacter des fichiers de données en mode avancé

Le mode avancé permet de compacter des fichiers de données autres que le fichier de données courant et/ou de désactiver les automatismes mis en oeuvre en mode standard (duplication et remplacement du fichier d'origine).

Pour sélectionner un fichier de données autres que le fichier courant, cliquez sur le bouton [...]. Une boîte de dialogue standard d'ouverture de documents s'affiche, vous permettant de désigner le fichier de données à compacter. Vous devez sélectionner un fichier de données compatible avec le fichier de structure ouvert.
Une fois la boîte de dialogue validée, le chemin d'accès du fichier à compacter est indiqué dans la fenêtre.

Cliquez sur le bouton **Compacter** de la zone pour démarrer le compactage. Une boîte de dialogue standard d'enregistrement de documents s'affiche, vous permettant de désigner l'emplacement du fichier défragmenté. Indiquez le nom et l'emplacement du nouveau fichier et cliquez sur **OK**.

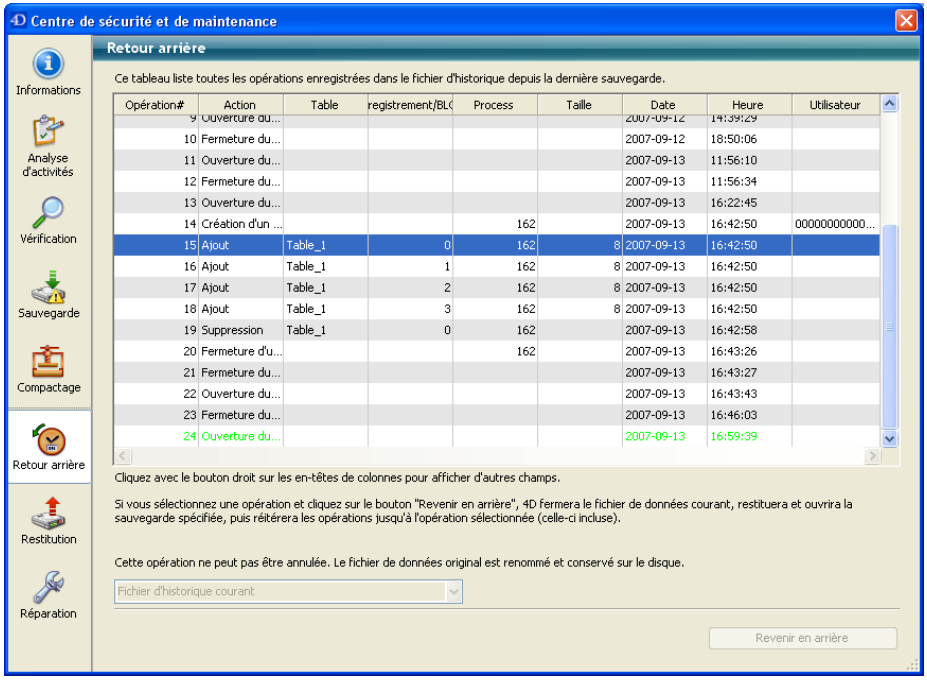
Il est important de noter que dans ce mode aucun automatisme spécifique n'est mis en oeuvre : le fichier de données d'origine n'est ni déplacé ni modifié.

Note Les fichiers d'index éventuellement associés au fichier de données désigné seront automatiquement compactés.

Retour arrière

Cette page permet d'accéder à la fonction de retour en arrière parmi les opérations effectuées sur le fichier d'historique. Cette fonction s'apparente à une fonction d'annulation sur plusieurs niveaux. Elle est

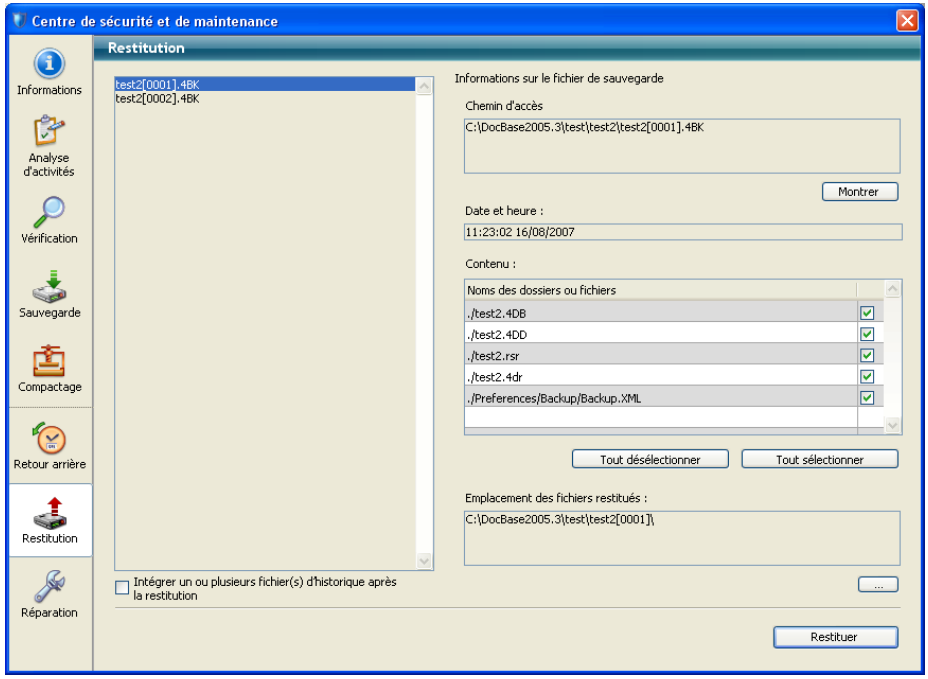
utile notamment lorsqu'un enregistrement a été supprimé par erreur dans une base de données.



Note Dans la version 11 de 4D, le fichier d'historique d'une base de données comporte par défaut l'extension ".journal" (au lieu de ".4DL" dans les versions précédentes du programme).

Restitution

Cette page permet de visualiser et de restituer manuellement les sauvegardes de la base de données :



La liste située dans la partie gauche de la fenêtre affiche les sauvegardes existantes de la base. Lorsque vous sélectionnez une sauvegarde dans cette liste, la partie droite de la fenêtre affiche les informations relatives à cette sauvegarde :

- **Chemin d'accès** : chemin d'accès complet du fichier de sauvegarde sélectionné. Le bouton **Montrer** ouvre le dossier de sauvegarde dans une fenêtre système.
- **Date et heure** : date et heure de la sauvegarde
- **Contenu** : contenu du fichier de sauvegarde. Chaque élément de la liste est associé à une case à cocher, permettant de spécifier si vous souhaitez ou non le restituer. Vous pouvez utiliser les boutons **Tout sélectionner** ou **Tout désélectionner** pour paramétrer la liste des éléments à restituer.
- **Emplacement des fichiers restitués** : dossier dans lequel seront placés les fichiers restitués. Pour modifier cet emplacement, cliquez sur le

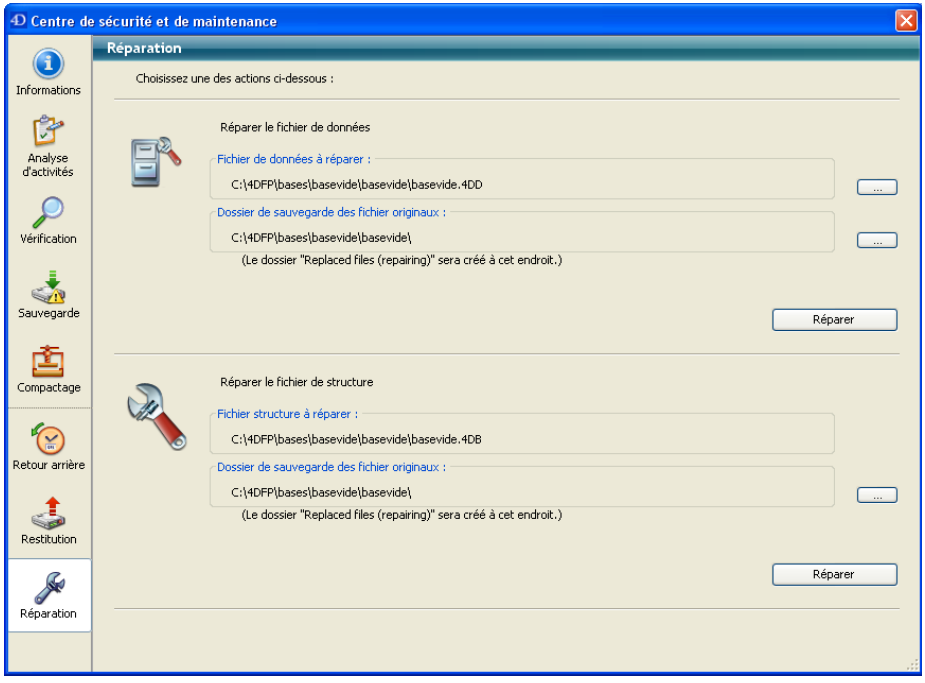
bouton [...] et désignez le dossier dans lequel vous souhaitez effectuer la restitution.

L'option **Intégrer un ou plusieurs fichier(s) d'historique après la restitution** permet d'intégrer successivement plusieurs sauvegardes d'historiques dans une base de données. Si, par exemple, vous disposez de 4 sauvegardes d'historique consécutives (correspondant à 4 sauvegardes de la base), vous pouvez restituer la première sauvegarde de la base puis intégrer une à une les sauvegardes d'historique. Ce principe permet par exemple de récupérer un fichier de données alors que les derniers fichiers de sauvegarde de la base sont manquants. Lorsque cette option est cochée, 4D affiche une boîte de dialogue standard d'ouverture de fichiers à l'issue de la restitution, vous permettant de sélectionner la sauvegarde d'historique à intégrer. La boîte de dialogue d'ouverture est affichée de nouveau après chaque intégration, jusqu'à ce qu'elle soit annulée.

Réparation

Cette page permet de réparer le fichier de données ou le fichier de structure lorsqu'il a été endommagé. Généralement, vous n'utiliserez ces fonctions qu'à la demande de 4D, lorsque des anomalies auront été détectées à l'ouverture de la base ou à la suite d'une vérification.

Note Pour plus d'informations sur la fonction de vérification, reportez-vous au paragraphe "Page Vérification", page 193.



Chaque réparation entraîne la duplication du fichier d'origine et donc l'augmentation de la taille du dossier de l'application. Il est important de tenir compte de ce mécanisme (notamment sous Mac OS où les applications 4D apparaissent sous forme de packages***) pour que l'application ne grossisse pas de façon excessive. Une intervention manuelle à l'intérieur du package permet peut être utile afin de supprimer les copies des fichiers d'origine.***

10 Utiliser le moteur SQL de 4D

L'une des plus importantes nouveautés proposées par 4D v11 est la prise en charge du standard SQL au coeur du moteur de la base de données.

Le SQL (*Structured Query Language*) est un langage standard utilisé pour créer, organiser, gérer et rechercher des informations stockées dans une base de données. Le SQL n'est pas en soi un système de gestion de données, c'est à la fois une composante intégrée de ce système, un langage et une interface de communication.

Le nouveau moteur SQL intégré de 4D est globalement conforme à la norme SQL92 (ou SQL-2), avec toutefois certaines différences liées à des implémentations avancées ou spécifiques et des limitations.

Ce chapitre traite des sujets suivants :

- les principes de l'implémentation du SQL dans 4D v11 et les modes d'accès à ce langage,
- les différences entre la norme SQL92 et le langage SQL de 4D.

Pour une liste détaillée des commandes du langage SQL prises en charge par 4D, veuillez vous reporter au *Guide de référence 4D SQL*.

Accéder au moteur SQL de 4D

Envoi de requêtes au moteur SQL de 4D

Le moteur SQL intégré de 4D peut être interrogé de trois manières :

- via la **nouvelle commande** CHERCHER PAR SQL. Le principe consiste à passer la clause WHERE d'une commande SQL SELECT en tant que paramètre à la commande. Par exemple :
  CHERCHER PAR SQL([OFFICES];"SALES > 100")
  Pour plus d'informations sur l'emploi de la commande CHERCHER PAR SQL, reportez-vous à la description de la routine CHERCHER PAR SQL, page 266.
- via les **commandes ODBC intégrées** de 4D, placées dans le thème "Sources de données externes" (ODBC SET PARAMETER, ODBC EXECUTE, etc.). Ces commandes ont été modifiées dans 4D v11 de manière à pouvoir fonctionner avec le moteur SQL 4D de la base de données courante.
  Pour plus d'informations sur l'emploi des commandes ODBC de haut niveau avec le moteur SQL de 4D, reportez-vous au paragraphe "Sources de données externes", page 260.
- via **l'éditeur de méthodes standard** de 4D. Les instructions SQL peuvent être saisies directement dans l'éditeur de méthodes standard de 4D. Il suffit d'insérer les requêtes SQL entre deux nouveaux mots-clés : Debut SQL et Fin SQL. Le code compris entre ces balises ne sera pas analysé par l'interpréteur de 4D et sera exécuté par le moteur SQL.
  Pour plus d'informations sur l'écriture de requêtes SQL au sein de méthodes 4D, reportez-vous au paragraphe "Saisir du code SQL dans l'éditeur de méthodes", page 135.

Echanger des données entre 4D et le moteur SQL

Référencer les expressions 4D

Il est possible de faire référence à tout type d'expression 4D valide (variable, champ, tableau, expression...) au sein des clauses WHERE et INTO des expressions SQL. Pour désigner une référence 4D, vous pouvez utiliser indifféremment l'une des deux notations suivantes :

- placer la référence entre chevrons "<<" et ">>"
- faire précéder la référence de deux-points ":"

Exemples :

```
C_ALPHA(80;vNom)
vNom:=Demander("Nom :")
ODBC EXECUTER("SELECT age FROM PEOPLE WHERE name=<<vNom>>")
```

ou bien :

```
C_ALPHA(80;vNom)
vNom:=Demander("Nom :")
Debut SQL
  SELECT age FROM PEOPLE WHERE name= :vNom
Fin SQL
```

Note Lorsque vous manipulez des variables interprocess (nommées <>*mavar*), vous devez encadrer le nom des variables avec des crochets [] (par exemple <<[<>*mavar*]>> ou :[<>*mavar*]).

Récupérer les données des requêtes SQL dans 4D

Les données issues d'une commande SELECT doivent être manipulées soit à l'intérieur d'un bloc Debut SQL/Fin SQL via la clause INTO de la commande SELECT, soit via les commandes 4D du thème "Source de données externes" (ODBC) :

- Dans le cas d'une structure Debut SQL/Fin SQL, vous pouvez utiliser la clause INTO de la requête SQL pour désigner tout objet 4D valide (champ, variable ou tableau) comme destinataire des valeurs. Vous

```
Debut SQL
  SELECT ename FROM emp INTO <<[Employés]Nom>>
Fin SQL
```

- Avec la commande ODBC EXECUTER, vous pouvez également utiliser les paramètres supplémentaires :

```
ODBC EXECUTER("SELECT ename FROM emp";[Employés]Nom)
```

Pour plus d'informations, reportez-vous à la description de la routine ODBC EXECUTER, page 261.

La principale différence entre les deux façons de récupérer les données issues d'une requête SQL (balises Début SQL/Fin SQL et commandes ODBC) est que dans le premier cas toutes les informations sont retournées à 4D en une seule fois, tandis que dans le second cas chaque enregistrement doit être chargé explicitement à l'aide de la commande ODBC CHARGER ENREGISTREMENT.

Par exemple, en supposant que la table PERSONS comporte 100 enregistrements.

- Récupération dans un tableau en utilisant les commandes ODBC :

```
TABLEAU ENTIER(tAnneeNaiss;0)
C_ALPHA(40;vNom)
vNom:="Smith"
$SQLStm:="SELECT Birth_Year FROM PERSONS WHERE
                                                ename= <<vNom>>"
ODBC EXECUTER($SQLStm;tAnneeNaiss)
Tant que(Non (ODBC Fin de selection))
  ODBC CHARGER ENREGISTREMENT(10)
Fin tant que
```

Ici, il est donc nécessaire d'effectuer 10 boucles afin de récupérer les 100 enregistrements. Pour charger tous les enregistrements en une seule fois, il suffit d'écrire :
ODBC CHARGER ENREGISTREMENT(ODBC Tous les enregistrements)

- Récupération dans un tableau en utilisant les balises Debut SQL/Fin SQL :

```
TABLEAU ENTIER (tAnneeNaiss;0)
C_ALPHA(40;vNom)
vNom:="Smith"
Debut SQL
  SELECT Birth_Year FROM PERSONS
    WHERE ename= <<vNom>> INTO <<tAnneeNaiss>>
Fin SQL
```

Dans ce cas, après l'exécution de l'instruction SELECT, le tableau *tAnneeNaiss* contient 100 éléments, chaque élément stockant une année de naissance provenant des 100 enregistrements.

Si, au lieu d'un tableau nous souhaitons récupérer les données dans un champ 4D, 4D créera automatiquement autant d'enregistrements que nécessaire pour contenir toutes les valeurs. Reprenons l'exemple d'une table PERSONS contenant 100 enregistrements.

- Récupération dans un champ en utilisant les commandes ODBC:

```
C_ALPHA(40;vNom)
vNom:="Smith"
$SQLStm:="SELECT Birth_Year FROM PERSONS WHERE
                                                ename= <<vNom>>"
ODBC EXECUTER($SQLStm;[Matable]Annee_Naiss)
```

```
Tant que(Non (ODBC Fin de selection))
  ODBC CHARGER ENREGISTREMENT(10)
Fin tant que
```

Ici, il est nécessaire d'effectuer 10 boucles afin de récupérer les 100 enregistrements. Chaque passage dans la boucle entraînera la création de 10 enregistrements dans la table [Matable] et chaque valeur *Birth_Year* récupérée de la table PERSONS sera stockée dans le champ Annee_Naiss.

- Récupération dans un champ en utilisant les balises Debut SQL/Fin SQL :

```
C_ALPHA(40;vNom)
vNom:="Smith"
Debut SQL
  SELECT Birth_Year FROM PERSONS
    WHERE ename= <<vNom>> INTO <<[Matable]Annee_Naiss>>
Fin SQL
```

Dans ce cas, pendant l'exécution de l'instruction SELECT, 100 enregistrements seront créés dans la table [Matable], les valeurs de la colonne *Birth_Year* étant stockées dans le champ 4D Annee_Naiss.

Utilisation d'une listbox

4D inclut un automatisme spécifique permettant de placer les données issues d'une requête SELECT dans une listbox. Pour plus d'informations, reportez-vous au paragraphe "Affichage du résultat d'une requête SQL", page 159.

Optimisation des requêtes

Pour des raisons d'optimisation, il est préférable d'utiliser des expressions 4D plutôt que des fonctions SQL dans les requêtes. En effet, les expressions 4D seront calculées une fois avant l'exécution de la requête tandis que les fonctions SQL sont évaluées pour chaque enregistrement trouvé.

Par exemple, avec l'instruction suivante :

```
ODBC EXECUTER("SELECT nomcomplet FROM PEOPLE WHERE
                                    nomcomplet=<<vNom+vPrenom>>")
```

... l'expression vNom+vPrenom est calculée une fois, avant l'exécution de la requête. Avec l'instruction suivante :

```
ODBC EXECUTER("SELECT nomcomplet FROM PEOPLE WHERE
                          nomcomplet=CONCAT(<<vNom>>,<<vPrenom>>)")
```

... la fonction CONCAT(<<vNom>>,<<vPrenom>>) est appelée pour chaque enregistrement de la table, autrement dit l'expression est évaluée pour chaque enregistrement.

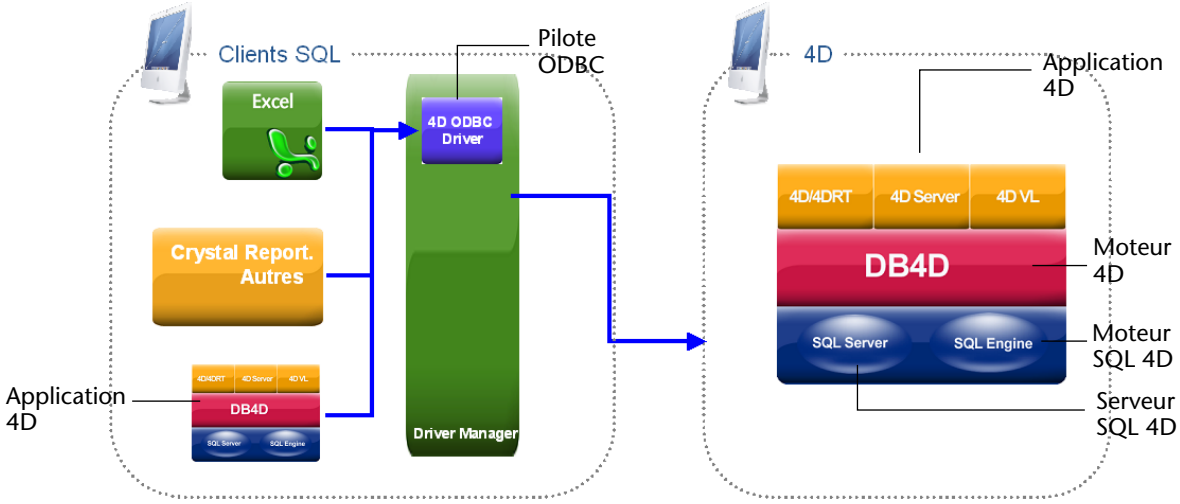
Configuration du serveur SQL de 4D

4D v11 inclut un serveur SQL performant permettant des accès externes aux données stockées dans la base 4D. Ces accès s'effectuent via un pilote ODBC 4D.

Le serveur SQL d'une application 4D peut être stoppé ou démarré à tout moment. En outre, pour des raisons de performances et de sécurité, vous pouvez définir le port TCP et l'adresse IP d'écoute et restreindre les possibilités d'accès à la base de données 4D.

Accès externes au serveur SQL

L'accès externe au serveur SQL de 4D est effectué via ODBC. 4D fournit un nouveau pilote ODBC (ou *driver* ODBC) permettant à toute application tierce (tableur de type Excel®, autre SGBD...) ou à une autre application 4D de se connecter au serveur SQL de 4D. Ce principe est résumé dans le schéma suivant :

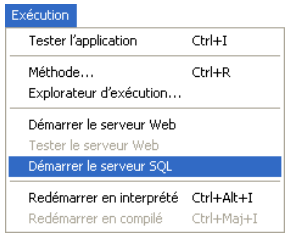


Le pilote ODBC 4D doit être installé sur le poste de la partie SQL cliente. L'installation et la configuration du pilote ODBC 4D est détaillée dans un manuel séparé.

Démarrer et stopper le serveur SQL

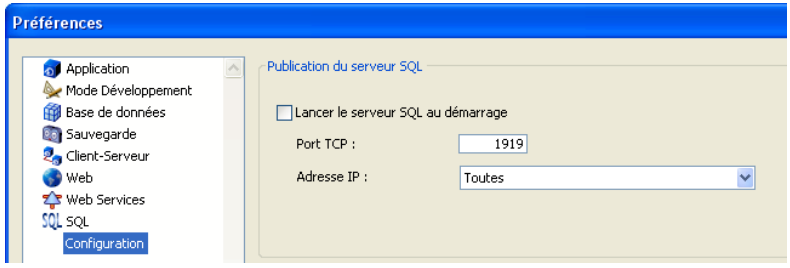
Dans 4D v11, le serveur SQL peut être démarré et stoppé de trois manières :

- Manuellement, via les commandes **Démarrer le serveur SQL** dans le menu **Exécution** de 4D :



  Lorsque le serveur est lancé, le libellé de cette commande devient **Arrêter le serveur SQL.**

- Automatiquement au démarrage de l'application, via les Préférences. Pour cela, affichez la page **SQL/Configuration** et cochez l'option **Lancer le serveur SQL au démarrage** :



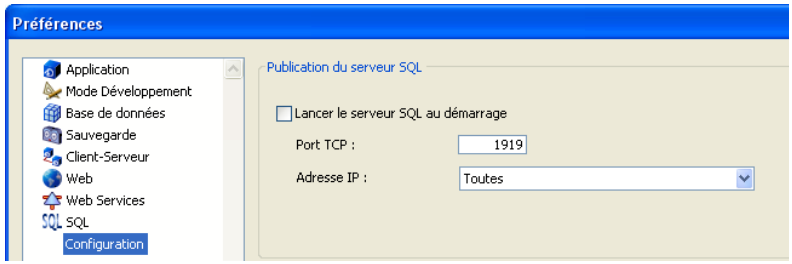
- Par programmation, via les nouvelles commandes LANCER SERVEUR SQL et ARRETER SERVEUR SQL (thème "SQL").

  Lorsque le serveur SQL est arrêté (ou tant qu'il n'a pas été lancé), 4D ne répond à aucune requête SQL externe.

Note L'arrêt du serveur SQL n'a pas d'influence sur le fonctionnement du moteur SQL de 4D. Le moteur SQL est accessible en permanence pour les requêtes internes.

Préférences de publication du serveur SQL

Il est possible de configurer les paramètres de publication par défaut du serveur SQL intégré de 4D. Ces paramètres sont accessibles dans la page **SQL/Configuration** des Préférences de la base :



- L'option **Lancer le serveur SQL au démarrage** permet de démarrer le serveur SQL au démarrage de l'application (cf. paragraphe "Démarrer et stopper le serveur SQL", page 211).
- **Port TCP** : Par défaut, le serveur SQL de 4D répond aux requêtes sur le port TCP 1919. Si ce port est déjà utilisé par un autre service ou si vos paramètres de connexion requièrent une autre configuration, vous pouvez changer le port TCP utilisé par le serveur SQL de 4D.

Note Si vous passez 0, 4D utilisera le numéro de port TCP par défaut, c'est-à-dire 1919.

- **Adresse IP** : Vous pouvez définir l'adresse IP de la machine sur laquelle le serveur SQL doit traiter les requêtes SQL. Par défaut, le serveur répond sur toutes les adresses IP (option **Toutes**).
  La liste déroulante "Adresse IP" liste automatiquement toutes les adresses IP présentes sur la machine. Lorsque vous sélectionnez une adresse particulière, le serveur ne répond qu'aux requêtes dirigées sur cette adresse. Cette fonctionnalité est destinée aux applications 4D hébergées sur des machines ayant plusieurs adresses TCP/IP.

Note Côté client, l'adresse IP et le port TCP du serveur SQL auquel l'application se connecte doivent être correctement configurées dans la définition de la source de données ODBC.

- **Activer SSL** : Cette option indique si le serveur SQL doit activer le protocole SSL pour traiter les connexions SQL.

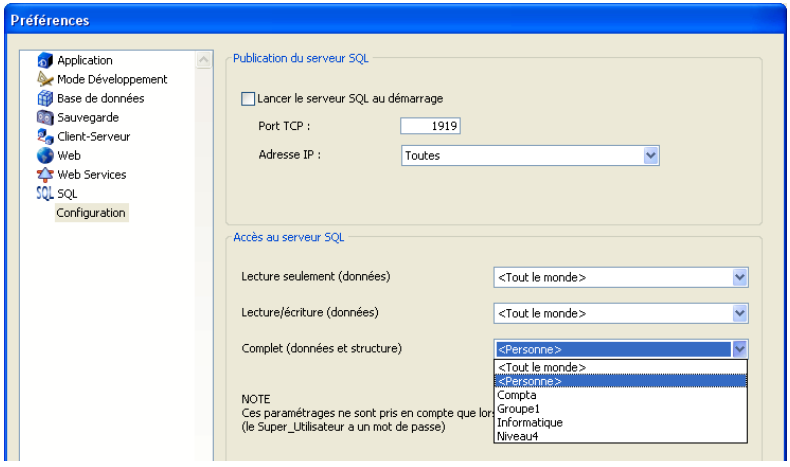
Contrôle des accès à la base de données 4D

Pour des raisons de sécurité, il est possible de contrôler les actions que les requêtes externes adressées au serveur SQL peuvent effectuer dans la base de données 4D.

Ce contrôle est effectué à deux niveaux :

- au niveau du type d'action autorisé,
- au niveau de l'utilisateur ayant effectué la requête.

Ces paramétrages sont effectués via la page **SQL/Configuration** des Préférences de la base :



Vous pouvez configurer séparément trois types d'accès à la base 4D via le serveur SQL :

- "Lecture seulement (données)" : libre accès en lecture à toutes les données des tables de la base mais ne permet pas l'ajout, la modification ou la suppression d'enregistrements ni la modification de la structure de la base.
- "Lecture/écriture (données)" : accès en lecture et en écriture (ajout, modification et suppression) à toutes les données des tables de la base mais ne permet pas la modification de la structure de la base.
- "Complet (données et structure)" : accès en lecture et en écriture (ajout, modification et suppression) à toutes les données des tables de la base et à la structure de la base (tables, champs, liens, etc).

Vous pouvez désigner un ensemble d'utilisateurs pour chaque type d'accès. Pour cela, vous disposez de trois types d'options :

- **<Personne>** : si vous sélectionnez cette option, le type d'accès sera refusé pour toutes les requêtes, quelle que soit leur provenance. Ce paramètre peut être utilisé même si le système de gestion des accès via des mots de passe de 4D n'est pas activé.
- **<Tout le monde>** : si vous sélectionnez cette option, le type d'accès sera accepté pour toutes les requêtes (aucun contrôle n'est effectué).
- *groupe d'utilisateurs* : cette option permet de désigner un groupe d'utilisateurs comme seul autorisé à effectuer le type d'accès associé. Cette option requiert que la gestion des mots de passe de 4D ait été activée. L'utilisateur à l'origine des requêtes fournit son nom et son mot de passe lors de la connexion au serveur SQL via ODBC.

ATTENTION : *Ce mécanisme s'appuie sur les mots de passe de 4D. Pour que le contrôle d'accès au serveur SQL soit effectif, le système de mots de passe de 4D doit être actif (un mot de passe doit avoir été attribué au Super_Utilisateur).*

Note Une option de sécurité supplémentaire peut être définie au niveau de chaque méthode projet 4D. Pour plus d'informations sur ce point, reportez-vous au paragraphe "Option "Disponible via SQL"", page 219.

Implémentations et limitations du moteur SQL de 4D

Fondamentalement, le moteur SQL de 4D est conforme à la norme SQL92. Cela signifie que, pour une description détaillée des commandes, fonctions, opérateurs et syntaxes à utiliser, vous pouvez vous référer à n'importe quelle documentation sur le SQL92. De multiples ressources sur ce thème sont disponibles, par exemple, sur Internet.

Cependant, le moteur SQL de 4D ne prend pas en charge 100 % des fonctions du SQL92 et propose en outre des fonctions supplémentaires, non présentes dans cette norme.

Ce paragraphe décrit les principales implémentations et limitations du moteur SQL de 4D. Vous trouverez une liste détaillée des fonctions SQL92 et de leur prise en charge par 4D dans le paragraphe "Prise en charge des fonctions SQL", page 228.

Limitations générales

Le moteur SQL de 4D étant intégré au coeur de la base de données de 4D, les limitations relatives au nombre maximum de tables, de colonnes (champs) et d'enregistrements par base ainsi que les règles de nommage des tables et des colonnes sont identiques à celles du moteur standard de 4D. Elles sont listées ci-dessous :

- Nombre maximum de tables par base : deux milliards en théorie, mais limitation à 32767 pour des raisons de compatibilité avec 4D v11.
- Nombre maximum de colonnes (champs) par table : deux milliards en théorie, mais limitation à 32767 pour des raisons de compatibilité avec 4D v11.
  Dans 4D v11 "Standard edition", le nombre maximum de colonnes par table est limité à 511.
- Nombre maximum de lignes (enregistrements) par table : un milliard. Pour les sous-champs, la limite est d'un milliard de sous-enregistrements pour la totalité des enregistrements.
- Nombre maximum de clés d'index : un milliard x 64.
- Une clé primaire ne peut pas être une valeur NULL et doit être unique. Il n'est pas obligatoire d'indexer les colonnes (champs) clés primaires.
- Nombre maximum de caractères pour les noms de tables et de champs : 31 caractères (limitation de 4D).
  Il n'est pas possible de créer plusieurs tables avec le même nom. Les mécanismes de contrôle standard de 4D sont appliqués.

Types de données

Le tableau suivant indique les types de données pris en charge dans le SQL de 4D ainsi que leur type correspondant dans 4D :

Type SQL 4D	Description	Type 4D v11
Varchar	Texte aphanumérique	Texte
Real	Nombre à virgule flottante compris dans l'intervalle +/-3,4E38	Réel
Numeric	Nombre compris dans l'intervalle +/-2E64	Entier 64 bits
Float	Nombre à virgule flottante (virtuellement infini)	Réel
Smallint	Nombre compris entre -32 768 et 32 767	Entier
Int	Nombre compris entre -2 147 483 648 et 2 147 483 647	Entier long
Bit	Champ qui n'accepte que la valeur TRUE ou FALSE	Booléen
Boolean	Champ qui n'accepte que la valeur TRUE ou FALSE	Booléen

Blob	Jusqu'à 2 Go ; tout objet binaire tel qu'une image, un document, une application...	BLOB
Bit varying	Jusqu'à 2 Go ; tout objet binaire tel qu'une image, un document, une application...	BLOB
Clob	Jusqu'à 2 Go de texte. Ce type de colonne (champ) ne peut pas être indexé. Il n'est pas stocké dans l'enregistrement lui-même.	Texte
Text	Jusqu'à 2 Go de texte. Ce type de colonne (champ) ne peut pas être indexé. Il n'est pas stocké dans l'enregistrement lui-même.	Texte
Timestamp	Date et Heure sous la forme Jour Mois Année Heures:Minutes:Secondes:Millisecondes	Parties Date et Heure gérées séparément (conversion automatique)
Duration	Durée sous la forme Jour:Heure:Minutes:Secondes:Millisecondes	Heure
Interval	Durée sous la forme Jour:Heure:Minutes:Secondes:Millisecondes	Heure
Picture	Image PICT jusqu'à 2 Go	Image

La conversion entre les types de données numériques est automatique. Les chaînes qui représentent un nombre ne sont pas converties en valeur numérique. L'opérateur de transtypage CAST permet de convertir des valeurs d'un type en un autre.

Les types de données SQL suivants ne sont pas implémentés :

- NCHAR
- NCHAR VARYING

Valeurs NULL dans 4D

La valeur NULL est implémentée dans le langage SQL de 4D ainsi que dans le moteur de base de données de 4D. En revanche, cette valeur n'existe pas dans le langage de 4D.

Note Il est toutefois possible de lire et d'écrire la valeur NULL dans un champ 4D via les nouvelles commandes Valeur champ Null et FIXER VALEUR CHAMP NULL.

Compatibilité des traitements et option Traduire les NULL en valeurs vides

Pour des raisons de compatibilité dans 4D v11, les valeurs NULL stockées dans les tables des bases de données 4D sont automatiquement converties en valeurs par défaut lors des

manipulations effectuées via le langage de 4D. Par exemple, dans le cas de l'instruction suivante :
mavarAlpha:=[matable]MonchpAlpha

... si le champ *MonchpAlpha* contient la valeur NULL, la variable *mavarAlpha* contiendra "" (chaîne vide).

Les valeurs par défaut dépendent du type de données :

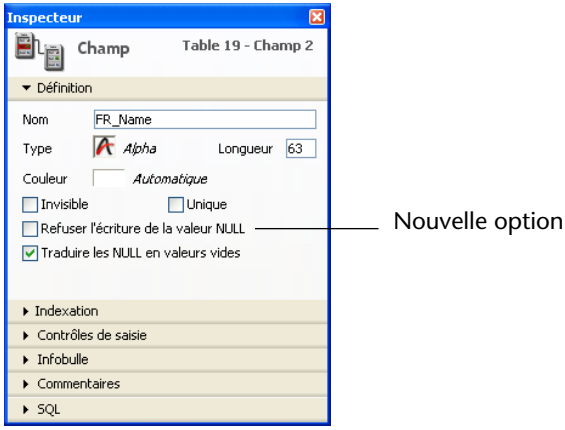
- Pour les types Alpha et Texte : ""
- Pour les types Numérique (réel), Entier et Entier long : 0
- Pour le type Date : "00/00/00"
- Pour le type Heure : "00:00:00"
- Pour le type Booléen : Faux
- Pour le type Image : Image vide
- Pour le type BLOB : BLOB vide

En revanche, ce mécanisme ne s'applique pas en principe aux traitements effectués au niveau du moteur de la base de données 4D, tels que les recherches. En effet, la recherche d'une valeur "vide" (par exemple *mavaleur=0*) ne trouvera pas les enregistrements stockant la valeur NULL, et inversement. Lorsque les deux types de valeurs (valeurs par défaut et NULL) cohabitent dans les enregistrements pour un même champ, certains traitements peuvent être faussés ou nécessiter du code supplémentaire.
Pour éviter ces désagréments, une nouvelle option permet d'uniformiser tous les traitements dans le langage de 4D v11 : **Traduire les NULL en valeurs vides**. Cette option, accessible dans l'Inspecteur des champs de l'éditeur de structure, permet d'étendre le principe d'usage des valeurs par défaut à tous les traitements. Les champs contenant la valeur NULL seront systématiquement considérés comme contenant la valeur par défaut. Cette option est cochée par défaut. Pour plus d'informations, reportez-vous au paragraphe "Options de prise en charge des NULL", page 121.

La propriété **Traduire les NULL en valeurs vides** est prise en compte à un niveau très bas du moteur de la base de données. Elle agit notamment sur la nouvelle routine Valeur champ Null, page 270.

Refuser l'écriture de la valeur NULL

La nouvelle propriété de champ **Refuser l'écriture de la valeur NULL** permet d'interdire le stockage de la valeur NULL :



Lorsque cet attribut est coché pour un champ, il ne sera pas possible de stocker la valeur NULL dans ce champ. Cette propriété de bas niveau correspond précisément à l'attribut NOT NULL du SQL.

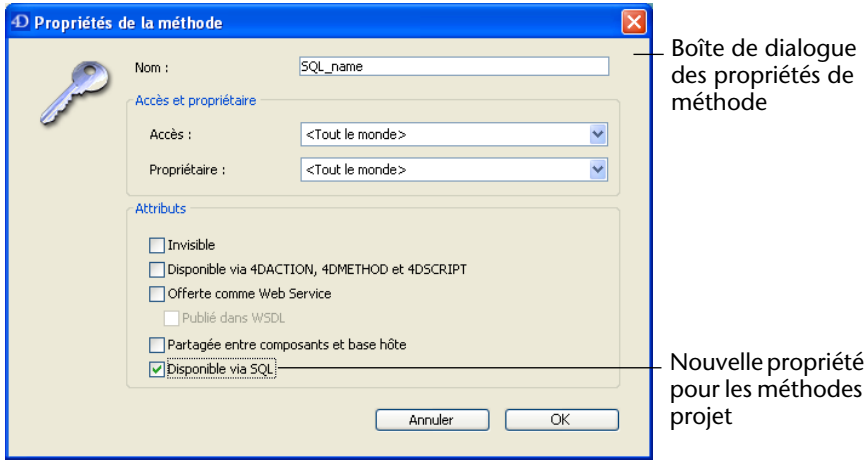
De manière générale, si vous souhaitez pouvoir utiliser des valeurs NULL dans votre base de données 4D, il est conseillé d'utiliser exclusivement le langage SQL de 4D.

Note Dans 4D, les champs peuvent également avoir l'attribut "Obligatoire". Les deux notions sont proches mais leur portée est différente : l'attribut "Obligatoire" est un contrôle de saisie, tandis que l'attribut "Refuser l'écriture de la valeur NULL" agit au niveau du moteur de la base de données.

Si un champ disposant de cet attribut reçoit la valeur NULL, une erreur est générée.

Option “Disponible via SQL”

Une propriété de sécurité a été ajoutée pour les méthodes projet de 4D : **Disponible via SQL** :



Lorsqu’elle est cochée, cette option autorise l’exécution de la méthode projet par le moteur SQL de 4D. Elle désélectionnée par défaut, ce qui signifie que, sauf autorisation explicite, les méthodes projet de 4D sont protégées et peuvent pas être appelées par le moteur SQL de 4D.

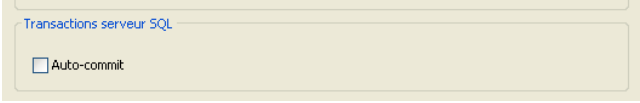
Cette propriété s’applique à toutes les requêtes SQL internes et externes — exécutées via le driver ODBC, le code SQL inséré dans les balises Debut SQL/Fin SQL ou la commande CHERCHER PAR SQL.

Notes

- Même si une méthode dispose de l’attribut “Disponible via SQL”, les accès définis au niveau des Préférences et des propriétés de la méthode sont pris en compte pour l’exécution de la méthode.
- La fonction ODBC *SQLProcedure* retourne uniquement les méthodes projet disposant de l’attribut “Disponible via SQL”.

Auto commit

Une option située dans la page SQL/Configuration des Préférences de 4D permet d’activer le mécanisme d’*auto-commit* dans le moteur SQL de 4D :



Le mode auto-commit a pour but de préserver l’intégrité référentielle des données. Lorsque cette option est cochée, toute requête SELECT, INSERT, UPDATE et DELETE (*SIUD*) effectuée en-dehors d’une transaction

est automatiquement incluse dans une transaction *ad hoc*. Ce principe garantit que les requêtes seront entièrement exécutées ou, en cas d'erreur, intégralement annulées.

Les requêtes incluses dans une transaction (gestion personnalisée de l'intégrité référentielle) ne sont pas affectées par cette option.

Lorsque cette option n'est pas cochée, aucune transaction automatique n'est générée (à l'exception des requêtes SELECT... FOR UPDATE, reportez-vous au paragraphe "Options de la commande SELECT", page 223). Par défaut, cette option n'est pas cochée.

Vous pouvez également gérer cette option par programmation à l'aide de la commande FIXER PARAMETRE BASE (cf. paragraphe "FIXER PARAMETRE BASE, Lire parametre base", page 359).

Note Seules les bases locales interrogées par le moteur SQL de 4D sont affectées par ce paramètre. Dans le cas de bases externes, le mécanisme d'auto-commit est pris en charge par les moteurs SQL distants.

Tables système

Le catalogue SQL de 4D comporte six tables système, accessible à tout utilisateur SQL disposant des droits d'accès en lecture : _USER_TABLES, _USER_COLUMNS, _USER_INDEXES, _USER_CONSTRAINTS, _USER_IND_COLUMNS et _USER _CONS_ COLUMNS.

Conformément aux usages dans le monde SQL, les tables système décrivent la structure de la base de données. Voici le descriptif de ces tables et de leurs champs :

_USER_TABLES		**Décrit les tables utilisateurs de la base**
TABLE_NAME	VARCHAR	Nom de table
TEMPORARY	BOOLEAN	Vrai si la table est temporaire, faux sinon
TABLE_ID	INT64	Numéro de table

_USER_COLUMNS		**Décrit les colonnes des tables utilisateurs de la base**
TABLE_NAME	VARCHAR	Nom de table
COLUMN_NAME	VARCHAR	Nom de colonne
DATA_TYPE	INT32	Type de colonne
DATA_LENGTH	INT32	Longueur de colonne

NULLABLE	BOOLEAN	Vrai si la colonne accepte des valeurs NULL, faux sinon
TABLE_ID	INT64	Numéro de table
COLUMN_ID	INT64	Numéro de colonne

_USER_INDEXES		**Décrit les index utilisateurs de la base**
INDEX_ID	VARCHAR	Numéro d'index
INDEX_NAME	VARCHAR	Nom d'index
INDEX_TYPE	INT32	Type d'index
TABLE_NAME	VARCHAR	Nom de table avec index
UNIQUENESS	BOOLEAN	Vrai si l'index impose une contrainte d'unicité, faux sinon
TABLE_ID	INT64	Numéro de table avec index

_USER_IND_COLUMNS		**Décrit les colonnes des index utilisateurs de la base**
INDEX_ID	VARCHAR	Numéro d'index
INDEX_NAME	VARCHAR	Nom d'index
TABLE_NAME	VARCHAR	Nom de table avec index
COLUMN_NAME	VARCHAR	Nom de colonne avec index
POSITION	INT32	Position de colonne dans l'index
TABLE_ID	INT64	Numéro de table avec index
COLUMN_ID	INT64	Numéro de colonne

_USER_CONSTRAINTS		**Décrit les contraintes utilisateurs de la base**
CONSTRAINT_ID	VARCHAR	Numéro de contrainte
CONSTRAINT_NAME	VARCHAR	Nom de contrainte
CONSTRAINT_TYPE	VARCHAR	Type de contrainte
TABLE_NAME	INT64	Nom de table avec contrainte
TABLE_ID	VARCHAR	Numéro de table avec contrainte
DELETE_RULE	VARCHAR	Règle de suppression – CASCADE ou RESTRICT

RELATED_TABLE_NAME	INT64	Nom de table liée
RELATED_TABLE_ID	VARCHAR	Numéro de table liée

_USER_CONS_COLUMNS		**Décrit les colonnes des contraintes utilisateurs de la base**
CONSTRAINT_ID	VARCHAR	Numéro de contrainte
CONSTRAINT_NAME	VARCHAR	Nom de contrainte
TABLE_NAME	VARCHAR	Nom de table avec contrainte
TABLE_ID	INT64	Numéro de table avec contrainte
COLUMN_NAME	VARCHAR	Nom de colonne avec contrainte
COLUMN_ID	INT64	Numéro de colonne avec contrainte
POSITION	INT32	Position de colonne dans une contrainte
RELATED_COLUMN_NAME	VARCHAR	Nom de colonne liée dans une contrainte
RELATED_COLUMN_ID	INT32	Numéro de colonne liée dans une contrainte

DML SQL

Ce paragraphe décrit les principales implémentations et limitation des fonctions de manipulation de données DML (*Data Manipulate Language*) dans le moteur SQL de 4D v11.

Pour une description des types de données pris en charge, reportez-vous au paragraphe “Types de données”, page 215.

Pour une description des commandes et opérateurs pris en charge, reportez-vous au paragraphe “Prise en charge des fonctions SQL”, page 228.

Noms de colonnes qualifiés

Il est possible de désigner une colonne SQL (un champ) en utilisant un nom de colonne qualifié. Par exemple, pour accéder à la colonne NAME de la table CUSTOMERS, vous pouvez écrire CUSTOMERS.NAME

Options de la commande SELECT

- Le requêtes mixant "*" et des champs spécifiques ne sont pas autorisées dans les instructions SELECT.
  - L'exemple suivant n'est PAS autorisé :
    SELECT *, SALES, TARGET FROM OFFICES
  - L'exemple suivant est en revanche parfaitement autorisé :
    SELECT * FROM OFFICES
- Il n'est pas possible d'insérer une expression à la place d'un nom de table dans la clause FROM.
- Les requêtes groupées (commande GROUP BY) acceptent les groupes de colonnes multiples.
- Conditions de recherche pour un groupe (clause HAVING) : vous pouvez utiliser la clause HAVING sans la clause GROUP BY.

Note Le moteur SQL de 4D dispose d'un mécanisme automatique incluant automatiquement les requêtes SELECT... FOR UPDATE au sein d'une transaction lorsqu'elles sont exécutées en-dehors de toute transaction. Ce principe permet de préserver l'intégrité des données.

INSERT, DELETE et UPDATE

- La commande INSERT est prise en charge dans les requêtes mono et multilignes.
  Cependant, une instruction INSERT multilignes n'autorise pas les opérations UNION et JOIN.
  - Par exemple, l'instruction suivante est parfaitement admise :
    INSERT INTO Table1 SELECT * FROM Table2
- La commande DELETE est prise en charge dans les requêtes. Toutefois, l'instruction DELETE positionnée n'est pas permise.
  Règles de suppression : les suppressions en cascade sont implémentées dans 4D, mais la règle de suppression SET DEFAULT n'est pas prise en charge.
- La commande UPDATE est prise en charge dans les requêtes et les sous-requêtes. Toutefois, l'instruction UPDATE positionnée n'est pas permise.
  Règles de mise à jour : les mises à jour en cascade sont implémentées

dans 4D, mais les règles de suppression SET NULL et SET DEFAULT ne sont pas prises en charge.

Triggers

Les triggers sont implémentés via le langage de 4D et sont pleinement pris en charge par le moteur SQL.

Contraintes

- Les **contraintes d'intégrité référentielle** suivantes sont implémentées :
  - Données obligatoires (colonnes obligatoire) : une colonne doit contenir une valeur non NULL.
  - Intégrité d'entité : implémenté dans le moteur de 4D.
- Les **contraintes SQL d'intégrité des données** *validity* et *checking* ne sont pas implémentées.
- Les quatre **contraintes de type** ne sont pas implémentées :
  - Contraintes de colonnes
  - Domains
  - Contraintes de tables
  - Assertions (ASSERTION)

TCL (Transactions) SQL

Les ordres COMMIT et ROLLBACK sont implémentés dans le SQL de 4D. Vous démarrez une transaction avec START TRANSACTION et la terminez soit avec l'ordre COMMIT, soit avec l'ordre ROLLBACK.

Les mécanismes des transactions 4D, intégrées au moteur de la base de données, sont également utilisables. Il est possible d'imbriquer des transactions jusqu'à 32767 niveaux.

Voici la correspondance des commandes :

SQL	4D
START (TRANSACTION)	DEBUT TRANSACTION
COMMIT (TRANSACTION)	VALIDER TRANSACTION
ROLLBACK (TRANSACTION)	ANNULER TRANSACTION

Au niveau de la base de données, il n'y a pas de différence entre une transaction 4D et une transaction SQL. Les deux types de transactions partagent les mêmes données et le même process. Les instructions SQL incluses dans une structure Debut SQL/Fin SQL, la commande CHERCHER PAR SQL et les commandes ODBC appliquées à la base locale

sont toujours exécutées dans le même contexte que les commandes 4D standard.

Les exemples suivants illustrent les différentes combinaisons de transactions.

- Ni “John” ni “Smith” ne seront ajoutés dans la table emp :

```
ODBC LOGIN(SQL_INTERNAL ;"";"")  `Initaliser le moteur SQL de 4D SQL
DEBUT TRANSACTION  `Démarrer une transaction dans le process courant
Debut SQL
  INSERT INTO emp
  (NAME)
  VALUES ('John');
Fin SQL
ODBC EXECUTER("START")  `Autre transaction dans le process courant
ODBC ANNULER CHARGEMENT
ODBC EXECUTER("INSERT INTO emp (NAME) VALUES ('Smith')")
  `Cette instruction est exécutée dans le même process
ODBC ANNULER CHARGEMENT
ODBC EXECUTER("ROLLBACK")  `Annuler la transaction interne du process
ANNULER TRANSACTION  `Annuler la transaction externe du process
ODBC LOGOUT
```

- Seul “John” sera ajouté dans la table emp :

```
ODBC LOGIN(SQL_INTERNAL ;"";"")
DEBUT TRANSACTION
Debut SQL
  INSERT INTO emp
  (NAME)
  VALUES ('John');
Fin SQL
ODBC EXECUTER("START")
ODBC ANNULER CHARGEMENT
ODBC EXECUTER("INSERT INTO emp (NAME) VALUES ('Smith')")
ODBC ANNULER CHARGEMENT
ODBC EXECUTER("ROLLBACK")  `Annuler la transaction interne du process
VALIDER TRANSACTION  `Valider la transaction externe du process
ODBC LOGOUT
```

- Ni “John” ni “Smith” ne seront ajoutés dans la table emp. La transaction externe annule la transaction interne :

```
ODBC LOGIN(SQL_INTERNAL ;"";"")
DEBUT TRANSACTION
Debut SQL
```

```
  INSERT INTO emp
  (NAME)
  VALUES ('John');
Fin SQL
ODBC EXECUTER("START")
ODBC ANNULER CHARGEMENT
ODBC EXECUTER("INSERT INTO emp (NAME) VALUES ('Smith')")
ODBC ANNULER CHARGEMENT
ODBC EXECUTER("COMMIT")  `Valider la transaction interne du process
ANNULER TRANSACTION  `Annuler la transaction externe du process
ODBC LOGOUT
```

▼ “John” et “Smith” seront ajoutés dans la table emp :

```
ODBC LOGIN(SQL_INTERNAL ;"";"")
DEBUT TRANSACTION
Debut SQL
  INSERT INTO emp
  (NAME)
  VALUES ('John');
Fin SQL
ODBC EXECUTER("START")
ODBC ANNULER CHARGEMENT
ODBC EXECUTER("INSERT INTO emp (NAME) VALUES ('Smith')")
ODBC ANNULER CHARGEMENT
ODBC EXECUTER("COMMIT")  `Valider la transaction interne du process
VALIDER TRANSACTION  `Valider la transaction externe du process
ODBC LOGOUT
```

DDL SQL

Ce paragraphe traite des implémentations et limitations des commandes DDL (Data Definition Language) dans le serveur SQL de 4D v11. Les commandes DDL permettent de définir et de manipuler la structure de la base.

Principales implémentations

Les commandes DDL suivantes sont implémentées, de façon complète ou partielle :

- CREATE TABLE est pris en charge. CREATE TABLE accepte la contrainte IF NOT EXISTS : la table est créée uniquement si une table de même nom n’existe pas déjà dans la base. Dans le cas contraire, elle n’est pas créée et aucune erreur n’est générée.
- DROP TABLE est pris en charge mais pas les concepts CASCADE et RESTRICT.

DROP TABLE accepte la contrainte IF EXISTS : si la table n'existe pas dans la base, la commande ne fait rien et aucune erreur n'est générée.

- ALTER TABLE est pris en charge mais pas les concepts CASCADE et RESTRICT.

Instructions DDL non prises en charges

Les commandes DDL suivantes ne sont pas implémentées dans le SQL de 4D v11 :

CREATE VIEW
DROP VIEW
ALTER VIEW
CREATE ALIAS
DROP ALIAS
CREATE SCHEMA
DROP SCHEMA
CREATE TRIGGER
DROP TRIGGER
ALTER TRIGGER
CREATE RULE
DROP RULE

DCL SQL (Privilèges)

Les concepts SQL relatifs aux droits d'accès (utilisateurs, objets de la base et privilèges) ne sont pas pris en charge. Dans l'implémentation actuelle, les concepts de gestion des accès de 4D sont utilisés (utilisateurs et groupes).

Connexions aux sources SQL

L'architecture multi-bases est implémentée au niveau du serveur SQL de 4D. Depuis 4D, il est possible :

- de se connecter à une base existante à l'aide de la commande ODBC LOGIN (pour plus d'informations, reportez-vous au paragraphe "Sources de données externes", page 260).
- de passer d'une base à l'autre en utilisant les nouvelles commandes 4D UTILISER BASE EXTERNE et UTILISER BASE INTERNE.

Les concepts SQL standard de Catalogues et de Schémas ne sont pas utilisés.

Limitations de la programmation SQL de 4D

Voici une liste récapitulative des concepts et instructions du SQL92 non pris en charge dans 4D v11:

- les instructions DELETE et UPDATE positionnées,
- les valeurs SQLCODE et SQLSTATE ; la gestion des erreurs est assurée via les mécanismes du langage de 4D (commande APPELER SUR ERREUR),
- la commande GET DIAGNOSTICS,
- l'instruction WHENEVER,
- la condition NOT FOUND,
- le concept de SQL dynamique — ces mécanismes pouvant toutefois être mis en place via le langage de 4D,
- le concept de Cursor et de Scroll cursor,
- le concept de procédure stockée externe. Cependant, il est possible d'exécuter depuis l'environnement SQL une méthode projet 4D (via un mécanisme semblable à celui des propriétés de méthodes pour les services Web) qui appelle elle-même une fonction appartenant à un plug-in externe.
- le concept de procédure stockée interne. Les procédures stockées de 4D doivent être utilisée.

Prise en charge des fonctions SQL

Les tableaux suivants listent les fonctions et opérateurs du SQL92 et indiquent leur prise en charge par le moteur SQL intégré de 4D. La colonne Commentaires fournit des précisions lorsque c'est nécessaire.

Fonctions statistiques

Fonction SQL	Description	SQL 4D	Commentaires
AVG	Moyenne	oui	Les valeurs NULL dans les colonnes sont ignorées par les fonctions de colonnes
COUNT, COUNT(*)	Nombre	oui	
MAX	Maximum	oui	
MIN	Minimum	oui	
SUM	Sum	oui	

Fonctions système

Fonction SQL	Description	SQL 4D	Commentaires
CURRENT_USER	Utilisateur courant	non	
SESSION_USER	Utilisateur autorisé	non	
SYSTEM_USER	Utilisateur système	non	
CURRENT_DATE	Date courante	oui	
CURRENT_TIME	Heure locale courante	oui	
CURRENT_TIMESTAMP	Date et heure locales courantes	oui	
CURDATE	Date courante	oui	Pas dans SQL92
CURTIME	Heure locale courante	oui	Pas dans SQL92

Fonctions standard

Fonction SQL	Description	SQL 4D	Commentaires
CAST	Transtypage	oui	
COALESCE	Valeur non NULL	oui	
NULLIF	Valeur NULL	oui	
OCTET_LENGTH	Longueur en octets	oui	

Fonctions chaîne

Fonction SQL	Description	SQL 4D	Commentaires
||	Concaténation	oui	
CHAR_LENGTH	Longueur de chaîne	oui	
CHARACTER_LENGTH	Longueur de chaîne	non	
LENGTH	Nombre de caractères	oui	
COLLATE	Remplacement d'une séquence de caractères	non	
CONCATENATE	Concaténation	oui	
CONCAT	Concaténation	oui	Pas dans SQL92
CONVERT	Conversion de type	non	
LOWER	Minuscule	oui	
POSITION	Position d'une chaîne dans une autre chaîne	oui	
SUBSTRING	Extraction d'une sous-chaîne	oui	
SUBSTR	Extraction d'une sous-chaîne	non	

TRANSLATE	Conversion du jeu de caractères	non	
TRIM	Suppression des caractères inutiles	oui	
LTRIM	Suppression des blancs à gauche	oui	Pas dans SQL92
RTRIM	Suppression des blancs à droite	oui	Pas dans SQL92
UPPER	Majuscule	oui	
CHAR	Retourne un caractère sur la base d'un code Ascii	oui	Pas dans SQL92
LOCATE	Position d'une chaîne dans une autre chaîne	oui	Pas dans SQL92
REPLACE	Remplacement de caractères	oui	Pas dans SQL92
SPACE	Génération d'espaces	oui	Pas dans SQL92
LEFT		oui	Pas dans SQL92
REPEAT		oui	Pas dans SQL92
RIGHT		oui	Pas dans SQL92
LIKE	Comparaison partielle	oui	• "%" est le caractère *joker* pour zéro ou N caractères. • "_" est le caractère *joker* pour un seul caractère.

Fonction bits

Fonction SQL	Description	SQL 4D	Commentaires
BIT_LENGTH	Longueur en bits	oui	Les chaînes 4D sont stockées en Unicode (1 caractère=2 octets)

Fonctions numériques

Fonction SQL	Description	SQL 4D	Commentaires
ABS	Valeur absolue	oui	Pas dans SQL92
ACOS	Arccosinus en radians	oui	Pas dans SQL92
ASCII	Code ascii	oui	Pas dans SQL92
ASIN	Arcsinus en radians	oui	Pas dans SQL92
ATAN	Arctangente en radians	oui	Pas dans SQL92
ATAN2	x y arctangente en radians	oui	Pas dans SQL92
CEILING	Plus petit entier supérieur ou égal au paramètre	oui	Pas dans SQL92
COS	Cosinus en radians	oui	Pas dans SQL92

COT	Cotangente en radians	oui	Pas dans SQL92
DEGREES	Nombre de degrés	oui	Pas dans SQL92
EXP	Valeur exponentielle	oui	Pas dans SQL92
FLOOR	Plus grand entier inférieur ou égal au paramètre	oui	Pas dans SQL92
LOG(float_exp)	Logarithme normal	oui	Pas dans SQL92
LOG10	Logarithme Base 10	oui	Pas dans SQL92
MOD	Reste (Modulo)	oui	Pas dans SQL92
PI	Valeur de Pi (Constante)	oui	Pas dans SQL92
POWER	Valeur à la puissance	oui	Pas dans SQL92
RADIANS	Nombre de radians convertis	oui	Pas dans SQL92
RAND	Valeur aléatoire	oui	Pas dans SQL92
ROUND	Valeur arrondie	oui	Pas dans SQL92
SIGN	Indicateur du signe du paramètre	oui	Pas dans SQL92
SIN	Sinus en radians	oui	Pas dans SQL92
SQRT	Racine carrée	oui	Pas dans SQL92
TAN	Tangente en radians	oui	Pas dans SQL92
TRUNC (TRUNCATE)	Troncature	oui	Pas dans SQL92
MILLISECOND	Date en millisecondes	oui	Pas dans SQL92

Fonctions heure

Fonction SQL	**Description**	**SQL 4D**	**Commentaires**
EXTRACT	Partie de date	oui	
INTERVAL (operating)	Durée	non	
OVERLAPS (Predicat)	Chevauchement de périodes	non	
DAY	Jour d'une date	oui	Pas dans SQL92
DAYNAME	Nom du jour	oui	Pas dans SQL92
DAYOFMONTH	Jour du mois	oui	Pas dans SQL92
DAYOFWEEK	Jour de la semaine	oui	Pas dans SQL92
DAYOFYEAR	Jour de l'année	oui	Pas dans SQL92
HOUR	Heure	oui	Pas dans SQL92
MINUTE	Minutes d'une date/heure	oui	Pas dans SQL92
MONTH	Mois d'une date	oui	Pas dans SQL92
MONTHNAME	Nom du mois	oui	Pas dans SQL92

QUARTER	Trimestre	oui	Pas dans SQL92
SECOND	Seconde	oui	Pas dans SQL92
WEEK	Semaine de l'année	oui	Pas dans SQL92
YEAR	Année d'une date	oui	Pas dans SQL92

Opérateurs arithmétiques

Fonction SQL	Description	SQL 4D	Commentaires
+ - * / ( )	Opérateurs arithmétiques et parenthèses	oui	La priorité de ces opérateurs est la suivante : * et / ont une priorité supérieure à + et -
%	Modulo	non	Pas dans SQL92

Opérateurs logiques et binaires

Fonction SQL	Description	SQL 4D	Commentaires
IS [NOT] TRUE	Vrai	non	
IS [NOT] FALSE	Faux	non	
IS [NOT] UNKNOWN	Inconnu	non	
IS [NOT] NULL	NULL	oui	TRUE AND NULL = NULL FALSE AND NULL = FALSE NULL AND NULL = NULL TRUE OR NULL = TRUE FALSE OR NULL = NULL NULL OR NULL = NULL
AND	Oui	oui	
OR	Ou	oui	
NOT	Non	oui	
&	"Et" logique	non	Pas dans SQL92
|	"Ou" logique	non	Pas dans SQL92
^	"Ou" logique exclusif	non	Pas dans SQL92

Opérateurs de comparaison

Fonction SQL	Description	SQL 4D	Commentaires
=	Egale	oui	
>	Supérieur à	oui	
<	Inférieur à	oui	
>=	Supérieur ou égal à	oui	

<=	Inférieur ou égal à	oui	
<>	Différent de	oui	

Predicate operators

Fonction SQL	Description	SQL 4D	Commentaires
[NOT] IN	Appartenance	oui	Si le test de l'expression retourne la valeur NULL, le test IN retourne NULL
[NOT] BETWEEN	Intervalle	oui	Les trois exceptions NULL sont traitées de la manière suivante : • Si les deux expressions définissant l'intervalle retournent des valeurs NULL, le test BETWEEN retourne un résultat NULL. • Si l'expression définissant la borne inférieure de l'intervalle produit une valeur NULL, le test BETWEEN retourne FAUX si la valeur testée est au-delà de la limite supérieure, et NULL sinon. • Si l'expression définissant la borne supérieure de l'intervalle produit une valeur NULL, le test BETWEEN retourne FAUX si la valeur testée est en-deçà de la limite inférieure, et NULL sinon.
[NOT] EXISTS	Existe	oui	
ALL	Comparaison à toutes les valeurs d'un ensemble	oui	
ANY	Comparaison à au moins une des valeurs d'un ensemble	oui	
SOME	Comparaison à au moins une des valeurs d'un ensemble	oui	

Manipulation des données

Fonction SQL	Description	SQL 4D	Commentaires
INTERSECT	Intersection (ensembles)	non	
UNION	Union (ensembles)	non	

EXCEPT	Différence (ensembles)	non	Identique à DIFFERENCE
DIFFERENCE	Différence (ensembles)	non	Identique à EXCEPT
INNER JOIN	Jointure interne	oui	
SELECT FROM WHERE		oui	
SELECT INTO		non	
INSERT INTO VALUES		oui	
INSERT INTO SELECT FROM		oui	
DELETE		oui	
UPDATE		oui	
DISTINCT	Valeurs distinctes	oui	

Définitions de données

Fonction SQL	Description	SQL 4D	Commentaires
CREATE TABLE		oui	La définition de stockage physique n'est pas prise en charge
DROP TABLE		oui	Les concepts CASCADE et RESTRICT ne sont pas pris en charge
ALTER TABLE		oui	Les concepts CASCADE et RESTRICT ne sont pas pris en charge
CREATE DATABASE		non	

Tri des données

Fonction SQL	Description	SQL 4D	Commentaires
ORDER BY	Tri	oui	
ORDER BY DESC		oui	
ORDER BY ASC		oui	
GROUP BY		oui	
GROUP BY HAVING		oui	

Jointure

Fonction SQL	Description	SQL 4D	Commentaires
INNER JOIN	Jointure interne	oui	
LEFT, RIGHT, FULL OUTER JOIN	Jointure externe	non	

NATURAL JOIN	Jointure naturelle	non	
UNION JOIN	Jointure union	non	
LEFT, RIGHT, FULL OUTER NATURAL JOIN	Jointure naturelle externe	non	

Expression conditionnelle

Fonction SQL	Description	SQL 4D	Commentaires
CASE	Structure de programmation	oui	
UNIQUE	Existe sans valeur dupliquée	non	
MATCH UNIQUE	Valeurs correspondantes	non	
LIMIT	Nombre de lignes retournées	oui	Pas dans SQL92 SELECT select_list [INTO new_table] FROM table_source [WHERE search_condition] [GROUP BY group_by_expression] [HAVING search_condition] [ORDER BY order_expression [ASC | DESC]] [LIMIT numeric-expression] [OFFSET numeric-expression] *Paramètres*: 1 select_list : liste de colonnes 2 new_table : table créée 3 table_source : table liée à la colonne 4 search_condition : condition de recherche 5 group_by_expression : expression groupée 6 search_condition : condition de recherche 7 order_expression : condition de tri 8 num-expression : valeur limite de selection 9 num-expression : valeur position de selection

Sous-requêtes

Fonction SQL		SQL 4D	Commentaires
dans clause SELECT	Imbriquées	non	
	Corrélées	non	
dans clause FROM	Imbriquées	non	
	Corrélées	non	
dans clause WHERE	Imbriquées	oui	
	Corrélées	oui	
dans clause HAVING	Imbriquées	oui	
	Corrélées	oui	
dans opérateur IN	Imbriquées	oui	
	Corrélées	oui	
dans opérateurs ALL, ANY, SOME	Imbriquées	oui	
	Corrélées	oui	
dans opérateur EXISTS	Imbriquées	oui	
	Corrélées	oui	

Index

Fonction SQL	SQL 4D	Commentaires
CREATE INDEX	oui	
DROP INDEX	oui	

Transactions

Fonction SQL	SQL 4D	Commentaires
START [TRANSACTION]	oui	
COMMIT [TRANSACTION]	oui	
ROLLBACK [TRANSACTION]	oui	

Codes d'erreur SQL

Le moteur SQL retourne des erreurs spécifiques, listées ci-dessous. Ces erreurs peuvent être interceptées à l'aide d'une méthode gestion d'erreurs installée par la commande APPELER SUR ERREUR.

- Erreurs génériques

1001 INVALID ARGUMENT
1002 INVALID INTERNAL STATE
1003 NOT RUNNING
1004 ACCESS DENIED
1005 FAILED TO LOCK SYNCHRONIZATION PRIMITIVE
1006 FAILED TO UNLOCK SYNCHRONIZATION PRIMITIVE
1007 SQL SERVER IS NOT AVAILABLE

- Erreurs sémantiques

1101 TABLE DOES NOT EXIST
1102 COLUMN DOES NOT EXIST
1103 TABLE NOT DECLARED IN FROM CLAUSE
1104 AMBIGUOUS COLUMN NAME
1105 TABLE ALIAS SAME AS TABLE NAME
1106 DUPLICATE TABLE ALIAS
1107 DUPLICATE TABLE IN FROM CLAUSE
1108 INCOMPATIBLE TYPES
1109 INVALID ORDER BY INDEX
1110 WRONG AMOUNT OF PARAMETERS
1111 INCOMPATIBLE PARAMETER TYPE
1112 UNKNOWN FUNCTION
1113 DIVISION BY ZERO
1114 ORDER BY INDEX NOT ALLOWED
1115 DISTINCT NOT ALLOWED
1116 NESTED COLUMN FUNCTIONS NOT ALLOWED
1117 COLUMN FUNCTIONS NOT ALLOWED
1118 CAN NOT MIX COLUMN AND SCALAR OPERATIONS
1119 INVALID GROUP BY INDEX
1120 GROUP BY INDEX NOT ALLOWED
1121 GROUP BY NOT ALLOWED WITH SELECT ALL
1122 NOT A COLUMN EXPRESSION
1123 NOT A GROUPING COLUMN IN AGGREGATE ORDER BY
1124 MIXED LITERAL TYPES IN PREDICATE
1125 LIKE ESCAPE IS NOT ONE CHAR

1126 BAD LIKE ESCAPE CHAR
1127 UNKNOWN ESCAPE SEQUENCE IN LIKE
1128 COLUMNS FROM MORE THAN ONE QUERY IN COLUMN FUNCTION
1129 SCALAR EXPRESSION WITH GROUP BY
1130 SUBQUERY HAS MORE THAN ONE COLUMN
1131 SUBQUERY MUST HAVE ONE ROW
1132 INSERT VALUE COUNT DOES NOT MATCH COLUMN COUNT
1133 DUPLICATE COLUMN IN INSERT
1134 COLUMN DOES NOT ALLOW NULLS
1135 DUPLICATE COLUMN IN UPDATE
1136 TABLE ALREADY EXISTS
1137 DUPLICATE COLUMN IN CREATE TABLE
1138 DUPLICATE COLUMN IN COLUMN LIST
1139 MORE THAN ONE PRIMARY KEY NOT ALLOWED
1140 AMBIGUOUS FOREIGN KEY NAME
1141 COLUMN COUNT MISMATCH IN FOREIGN KEY
1142 COLUMN TYPE MISMATCH IN FOREIGN KEY
1143 FAILED TO FIND MATCHING PRIMARY COLUMN
1144 UPDATE AND DELETE CONSTRAINTS MUST BE THE SAME
1145 FOREIGN KEY DOES NOT EXIST
1146 INVALID LIMIT VALUE IN SELECT
1147 INVALID OFFSET VALUE IN SELECT
1148 PRIMARY KEY DOES NOT EXIST
1149 FAILED TO CREATE FOREIGN KEY
1150 FIELD IS NOT IN PRIMARY KEY
1151 FIELD IS NOT UPDATEABLE
1153 BAD DATA TYPE LENGTH
1154 EXPECTED EXECUTE IMMEDIATE COMMAND

- Implémentations

1203 FUNCTIONALITY IS NOT IMPLEMENTED
1204 FAILED TO CREATE NEW RECORD
1205 FAILED TO UPDATE FIELD
1206 FAILED TO DELETE RECORD
1207 NO MORE JOIN SEEDS POSSIBLE

1208 FAILED TO CREATE TABLE
1209 FAILED TO DROP TABLE
1210 CANT BUILD BTREE FOR ZERO RECORDS
1211 COMMAND COUNT GREATER THAN ALLOWED
1212 FAILED TO CREATE DATABASE
1213 FAILED TO DROP COLUMN
1214 VALUE IS OUT OF BOUNDS
1215 FAILED TO STOP SQL_SERVER
1216 FAILED TO LOCALIZE
1217 FAILED TO LOCK TABLE FOR READING
1218 FAILED TO LOCK TABLE FOR WRITING
1219 TABLE STRUCTURE STAMP CHANGED
1220 FAILED TO LOAD RECORD
1221 FAILED TO LOCK RECORD FOR WRITING
1222 FAILED TO PUT SQL LOCK ON A TABLE

- Analyse

1301 PARSING FAILED

- Accès au langage runtime

1401 COMMAND NOT SPECIFIED
1402 ALREADY LOGGED IN
1403 SESSION DOES NOT EXIST
1404 UNKNOWN BIND ENTITY
1405 INCOMPATIBLE BIND ENTITIES
1406 REQUEST RESULT NOT AVAILABLE
1407 BINDING LOAD FAILED
1408 COULD NOT RECOVER FROM PREVIOUS ERRORS
1409 NO OPEN STATEMENT
1410 RESULT EOF
1411 BOUND VALUE IS NULL
1412 STATEMENT ALREADY OPENED
1413 FAILED TO GET PARAMETER VALUE
1414 INCOMPATIBLE PARAMETER ENTITIES
1415 PARAMETER VALUE NOT SPECIFIED
1416 COLUMN REFERENCE PARAMETERS FROM DIFFERENT TABLES

1417 EMPTY STATEMENT
1418 FAILED TO UPDATE VARIABLE
1419 FAILED TO GET TABLE REFERENCE
1420 FAILED TO GET TABLE CONTEXT
1421 COLUMNS NOT ALLOWED
1422 INVALID COMMAND COUNT
1423 INTO CLAUSE NOT ALLOWED
1424 EXECUTE IMMEDIATE NOT ALLOWED
1425 ARRAY NOT ALLOWED IN EXECUTE IMMEDIATE
1426 COLUMN NOT ALLOWED IN EXECUTE IMMEDIATE
1427 NESTED BEGIN END SQL NOT ALLOWED
1428 RESULT IS NOT A SELECTION
1429 INTO ITEM IS NOT A VARIABLE (LANGUAGE RUNTIME)
1430 VARIABLE WAS NOT FOUND (LANGUAGE RUNTIME)

- Analyse de date

1501 SEPARATOR_EXPECTED
1502 FAILED TO PARSE DAY OF MONTH
1503 FAILED TO PARSE MONTH
1504 FAILED TO PARSE YEAR
1505 FAILED TO PARSE HOUR
1506 FAILED TO PARSE MINUTE
1507 FAILED TO PARSE SECOND
1508 FAILED TO PARSE MILLISECOND
1509 INVALID AM PM USAGE
1510 FAILED TO PARSE TIME ZONE
1511 UNEXPECTED CHARACTER
1512 FAILED TO PARSE TIMESTAMP
1513 FAILED TO PARSE DURATION

- Formatage de date

1551 FAILED

- Erreurs lexer

1601 NULL INPUT STRING
1602 NON TERMINATED STRING

1603 NON TERMINATED COMMENT
1604 INVALID NUMBER
1605 UNKNOWN START OF TOKEN
1606 NON TERMINATED NAME
1607 NO VALID TOKENS

- Erreurs moteur 4D

1837 DB4D QUERY FAILED

- Erreurs de cache

2000 CACHEABLE NOT INITIALIZED
2001 VALUE ALREADY CACHED
2002 CACHED VALUE NOT FOUND

- Erreurs de protocole

3000 HEADER NOT FOUND
3001 UNKNOWN COMMAND
3002 ALREADY LOGGED IN
3003 NOT LOGGED IN
3004 UNKNOWN OUTPUT MODE
3005 INVALID STATEMENT ID
3006 UNKNOWN DATA TYPE
3007 STILL LOGGED IN
3008 SOCKET READ ERROR
3009 SOCKET WRITE ERROR
3010 BASE64 DECODING ERROR
3011 SESSION TIMEOUT
3012 FETCH TIMESTAMP ALREADY EXISTS
3013 BASE64 ENCODING ERROR
3014 INVALID HEADER TERMINATOR

11 Serveur Web

Plusieurs nouveautés proposées dans 4D version 11 sont relatives au serveur Web intégré de 4D :exceptionnel

- Il est désormais possible d'utiliser le mode "Digest" pour l'authentification des connexions au serveur Web.
- Différents formats standard sont maintenant proposés pour le fichier d'historique du serveur Web et un système de sauvegarde périodique du fichier est proposé.

Authentification en mode Digest

Principes

4D v11 permet désormais d'utiliser le mode **Digest** pour l'authentification des demandes de connexion au serveur Web intégré. Le mode d'authentification concerne la manière dont sont collectées et traitées les informations relatives au nom d'utilisateur et au mot de passe.

Dans les versions précédentes de 4D, seul le mode Basic était disponible. En mode Basic, le nom et le mot de passe saisis par l'utilisateur sont envoyés en clair dans les requêtes HTTP. Ce principe n'assure pas une sécurité totale au système dans la mesure où ces informations peuvent être interceptées et utilisées par un tiers.

Le mode Digest procure un niveau de sécurité plus élevé car les informations d'authentification sont traitées par un processus unidirectionnel appelé *hachage* qui rend leur contenu impossible à décrypter.

Côté utilisateur, l'emploi d'un mode d'authentification ou de l'autre est transparent.

Toutefois, l'authentification Digest est une fonction de HTTP1.1 et n'est pas prise en charge par tous les navigateurs. Par exemple, seules les versions 5.0 et suivantes du navigateur Microsoft Internet Explorer acceptent ce mode.
Si un navigateur ne prenant pas en charge cette fonctionnalité adresse une demande au serveur Web alors que l'authentification Digest est activée, le serveur rejette la demande et retourne un message d'erreur au navigateur.

Mise en oeuvre dans 4D

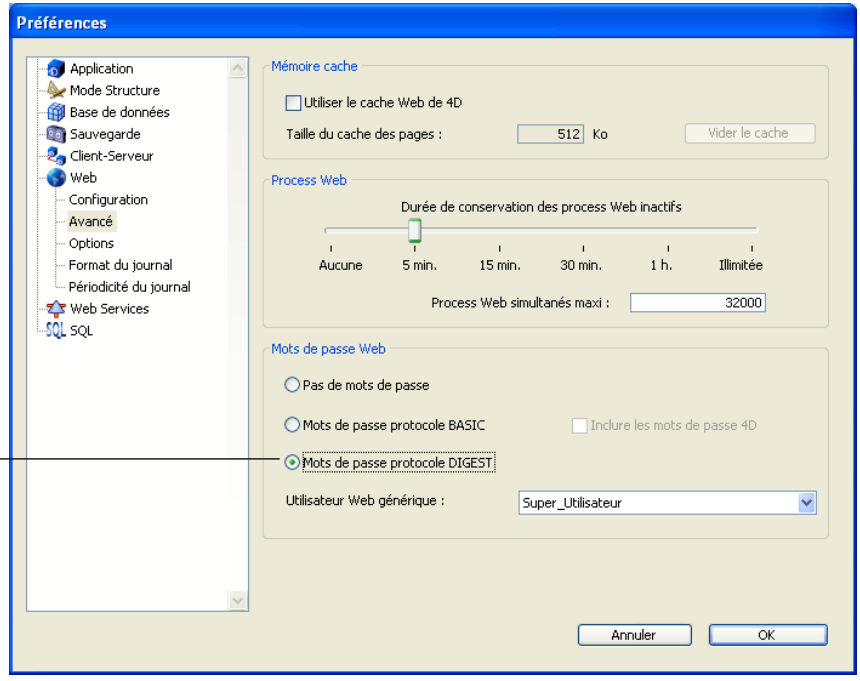
Pour des raisons de compatibilité, le mode d'authentification Basic est utilisé par défaut dans les bases 4D converties en version 11 (si l'option "Utiliser mots de passe" était cochée dans la version précédente). Vous devez activer explicitement le mode Digest.

Pour activer et utiliser le mode Digest, vous devez :

1. Cocher la nouvelle option "Mots de passe protocole DIGEST" dans la page Web/Avancé des Préférences.
2. Utiliser la nouvelle commande Valider mot de passe digest Web dans la Méthode base Sur authentification Web afin d'accepter ou de rejeter la connexion.

Préférences

L'activation du mode Digest s'effectue via l'option "Mots de passe protocole DIGEST" dans la page Web/Avancé des Préférences :



Activation du mode DIGEST

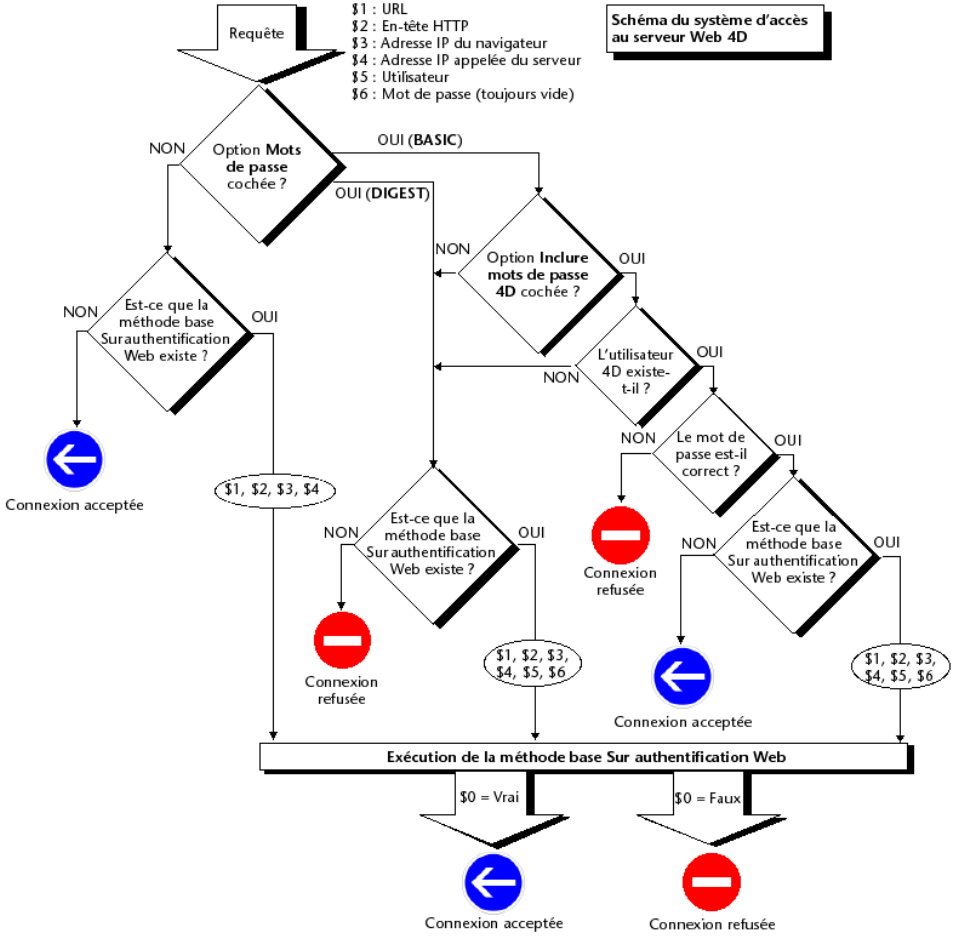
Dans 4D v11, le paramétrage de l'authentification Web s'effectue via trois options :

- **Pas de mots de passe** : aucune authentification n'est effectuée pour la connexion au serveur Web (équivaut à l'option "Mots de passe" non cochée dans les versions précédentes de 4D).
- **Mots de passe protocole BASIC** : authentification standard en mode BASIC (équivaut à l'option "Mots de passe" cochée dans les versions précédentes de 4D).
- **Mots de passe protocole DIGEST** : authentification en mode Digest. Ce nouveau mode est disponible à partir de 4D v11. Si vous cochez cette option, vous devez gérer les connexions à l'aide de la nouvelle commande Valider mot de passe digest Web dans la Méthode base Sur authentification Web.

Note Vous devez redémarrer le serveur Web pour que les modifications effectuées sur ces paramètres soient prises en compte.

A la différence du mode Basic, le mode Digest n'est pas compatible avec les mots de passe 4D standard : il n'est pas possible d'utiliser les mots de passe 4D comme identifiants Web. L'option "Inclure les mots de passe 4D" est grisée lorsque ce mode est sélectionné.
Les identifiants des utilisateurs Web doivent être gérés de façon personnalisée (par exemple via une table).

Voici le schéma d'accès au serveur Web de 4D v11 :



Méthode base Sur authentification Web

Lorsque le mode Digest est activé, le paramètre $6 (mot de passe) est retourné vide dans la Méthode base Sur authentification Web. En effet dans ce mode, cette information ne transite pas en clair par le réseau.

Vous devez impérativement dans ce cas évaluer la demande de connexion à l'aide de la nouvelle commande **Valider mot de passe digest Web**. Cette nouvelle commande est détaillée dans le paragraphe "Serveur Web", page 407.

Paramétrage de l'historique des requêtes (logweb.txt)

Les mécanismes de génération du fichier d'historique des requêtes Web (*logweb.txt*) sont désormais paramétrables dans 4D v11.

Le fichier *logweb.txt* permet d'analyser a posteriori les requêtes adressées au serveur Web de 4D. Il est automatiquement placé :

- avec 4D et 4D Server, à côté du fichier de structure,
- avec 4D Client ou une application exécutable, à côté de l'application (Windows) ou du progiciel (Mac OS).

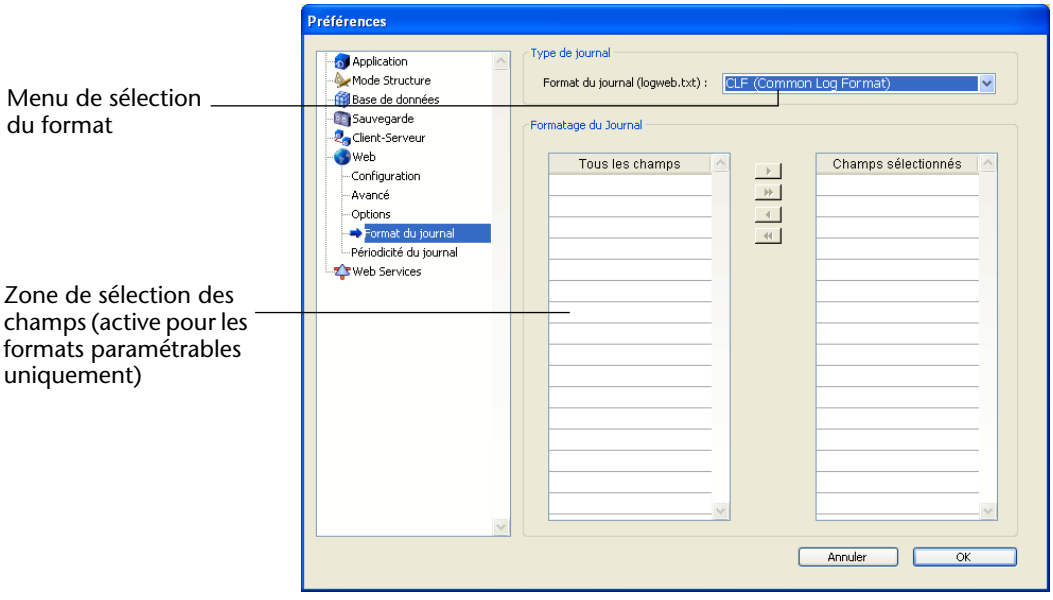
Dans les versions précédentes de 4D, il était seulement possible d'activer ou non l'enregistrement de ce fichier. Dans 4D v11, vous pouvez désormais configurer :

- son format interne,
- sa fréquence de sauvegarde.

Format du fichier

Dans les versions précédentes de 4D, seul le format CLF (*Common LogFile Format*) était possible pour le fichier *logweb.txt*. Ce format est toujours utilisable dans 4D v11, mais trois autres formats standard sont désormais proposés : DLF (*Combined Log Format*), ELF (*Extended Log format*) et WLF (*Webstar Log Format*). Les deux derniers formats sont paramétrables.

L'activation et la configuration du contenu du fichier d'historique s'effectue dans les Préférences de l'application, page Web/Format du journal :



Note L'activation et la désactivation du fichier d'historique des requêtes peut également être effectuée par programmation, à l'aide de la commande FIXER PARAMETRE BASE. Cette commande a été modifiée dans 4D v11 de manière à prendre en compte les nouveaux formats de fichiers. Pour plus d'informations, reportez-vous au paragraphe "FIXER PARAMETRE BASE, Lire parametre base", page 359.

Le menu de format du journal propose les options suivantes :

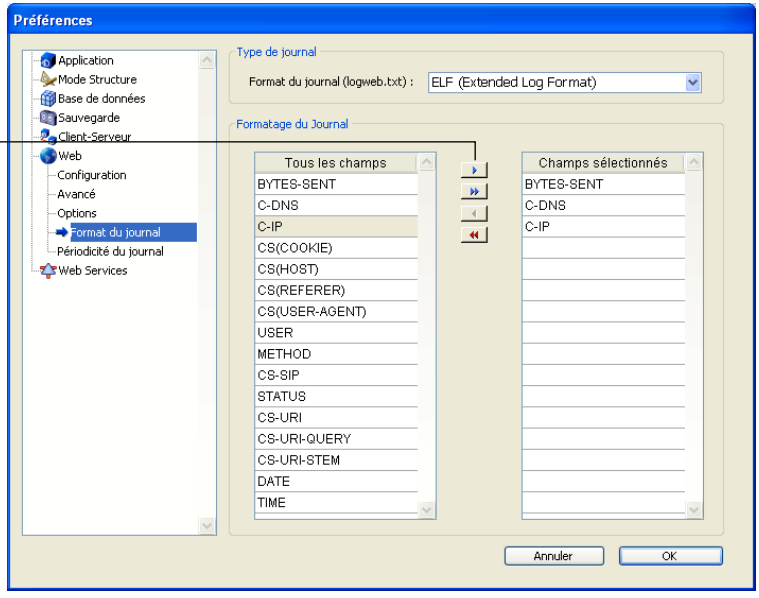
- **Pas de journal** : lorsque cette option est sélectionnée, 4D ne génère pas d'historique des requêtes.
  Dans les versions précédentes de 4D, ce fonctionnement était obtenu par la désélection de l'option "Enregistrer requêtes dans fichier (logweb.txt)".
- **CLF (Common Log Format)** : lorsque cette option est sélectionnée, l'historique des requêtes est généré au format CLF. Cette option correspond au format utilisé dans les versions précédentes de 4D. Pour plus d'informations sur le format CLF, reportez-vous au manuel *Langage* de 4D.
  Le format CLF ne peut pas être personnalisé.

- **DLF (Combined Log Format)** : lorsque cette option est sélectionnée, l'historique des requêtes est généré au format DLF.
  Le format DLF est semblable au format CLF dont il reprend exactement la structure. Il contient simplement deux champs HTTP supplémentaires à la fin de chaque requête : *Referer* et *User-agent*.
  - *Referer* : contient l'URL de la page pointant vers le document demandé.
  - *User-agent* : contient le nom et la version du navigateur ou du logiciel client à l'origine de la requête.

  Le format DLF ne peut pas être personnalisé.
- **ELF (Extended Log Format)** : lorsque cette option est sélectionnée, l'historique des requêtes est généré au format ELF.
  Le format ELF est largement répandu dans le monde des serveurs HTTP. Il permet de construire des historiques sophistiqués, répondant à des besoins spécifiques. Pour cette raison, le format ELF est personnalisable : il est possible de choisir les champs à enregistrer ainsi que leur ordre d'insertion dans le fichier.
- **WLF (WebStar Log Format)** : lorsque cette option est sélectionnée, l'historique des requêtes est généré au format WLF.
  Le format WLF a été développé spécifiquement pour le serveur 4D WebSTAR. Il est semblable au format ELF, il dispose simplement de champs supplémentaires. Comme le format ELF, il est personnalisable.

Configurer les champs Lorsque vous choisissez le format *ELF (Extended Log Format)* ou *WLF (WebStar Log Format)*, la zone "Formatage du journal" affiche les champs disponibles pour le format. Vous devez sélectionner chaque

champ à inclure dans l'historique. Pour cela, utilisez les flèches de commande ou procédez par glisser-déposer :



Note Il n'est pas possible de sélectionner deux fois le même champ.

Le tableau suivant liste les champs disponibles pour chaque format (par ordre alphabétique) et décrit son contenu :

Champ	ELF	WLF	Valeur
BYTES_RECEIVED		X	Nombre d'octets reçus par le serveur
BYTES_SENT	X	X	Nombre d'octets envoyés par le serveur au client
C_DNS	X	X	Adresse IP du DNS (ELF : champ identique au champ C_IP)
C_IP	X	X	Adresse IP du client (par exemple 192.100.100.10)
CONNECTION_ID		X	Numéro unique de la connexion
CS(COOKIE)	X	X	Informations sur les cookies contenus dans la requête HTTP
CS(HOST)	X	X	Champ Host de la requête HTTP
CS(REFERER)	X	X	URL de la page pointant vers le document demandé

Champ	ELF	WLF	Valeur
CS(USER_AGENT)	X	X	Informations sur le logiciel et le système d'exploitation du client
CS_SIP	X	X	Adresse IP du serveur
CS_URI	X	X	URI sur lequel la requête est effectuée
CS_URI_QUERY	X	X	Paramètres d'interrogation de la requête
CS_URI_STEM	X	X	Partie de la requête sans les paramètres d'interrogation
DATE	X	X	DD: jour, MMM: abréviation de 3 lettres pour le mois (Jan, Feb,...), YYYY: année
METHOD	X	X	Méthode HTTP utilisée pour la requête adressée au serveur
PATH_ARGS		X	Paramètres de la CGI : chaîne située après le caractère "$"
STATUS	X	X	Réponse fournie par le serveur
TIME	X	X	HH: heure, MM: minutes, SS: secondes
TRANSFER_TIME	X	X	Délai ayant été nécessaire au serveur pour générer la réponse
USER	X	X	Nom d'utilisateur s'il s'est authentifié, sinon - (signe moins). Si le nom d'utilisateur contient des espaces, ils sont remplacés par des _ (traits de soulignement)
URL		X	URL demandé par le client

Note La date et l'heure sont indiquées en GMT.

Périodicité de sauvegarde

Le fichier d'historique des requêtes Web pouvant atteindre une taille importante, il est possible de mettre en place un mécanisme d'archivage automatique. Le déclenchement de l'archivage peut être basé sur un délai (exprimé en heures, jours, semaines ou mois) ou une taille de fichier ; à chaque échéance, 4D referme et archive automatiquement le fichier d'historique courant et en crée un nouveau.

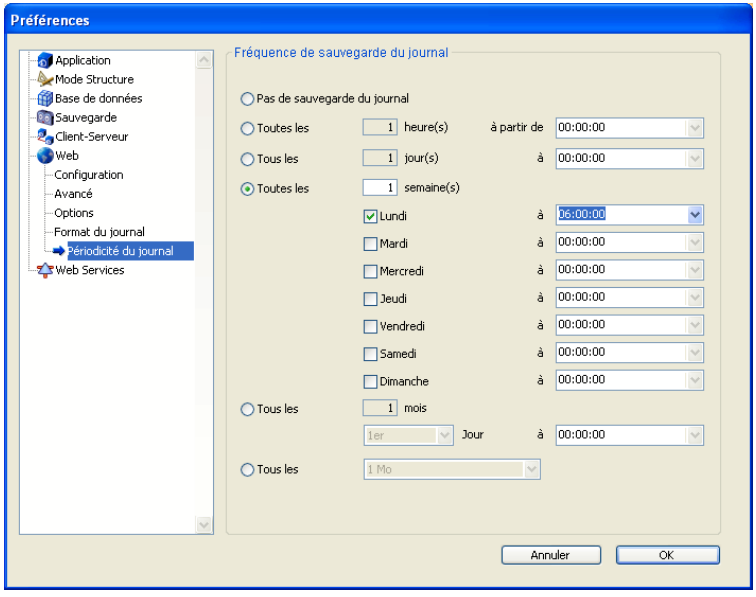
Nom des archives

Lorsque l'archivage du ficher d'historique se déclenche, le fichier d'historique est archivé dans un dossier nommé "**Logweb Archives**", créé au même niveau que le fichier *logweb.txt* (c'est-à-dire à côté du fichier de structure de la base).
Le fichier archivé est renommé sur le modèle

"DAAAA_MM_JJ_Thh_mm_ss.txt". Par exemple, pour un fichier archivé le 4 septembre 2006 à 15 heures 50 minutes et 7 secondes : "D2006_09_04_T15_50_07.txt"

Paramétrage des sauvegardes

Les paramètres d'archivage automatique de l'historique des requêtes sont définis dans la page **Web/Périodicité du journal** des Préférences de l'application :



Vous devez dans un premier temps choisir une échelle de fréquence (jours, semaines...) ou le critère de taille limite en cliquant sur le bouton radio correspondant. Vous devez ensuite éventuellement préciser le moment de la sauvegarde.

- **Pas de sauvegarde du journal** : la fonction de sauvegarde périodique est inactivée.
- **Toutes les *N* heure(s)** : cette option permet de programmer la sauvegarde sur une base horaire. Vous pouvez saisir une valeur comprise entre 1 et 24.
  - **à partir de** : permet de définir l'heure à laquelle débutera la première sauvegarde horaire.
- **Tous les *N* jour(s) à *N*** : cette option permet de programmer la sauvegarde sur une base journalière. Saisissez 1 si vous souhaitez une sauve-

garde quotidienne. Lorsque vous cochez cette option, vous devez indiquer l'heure à laquelle la sauvegarde doit être déclenchée.

- **Toutes les *N* semaine(s), *jour* à *N*** : cette option permet de programmer la sauvegarde sur une base hebdomadaire. Saisissez 1 si vous souhaitez une sauvegarde hebdomadaire. Lorsque vous cochez cette option, vous devez indiquer le ou les jour(s) de la semaine et l'heure à laquelle chaque sauvegarde doit être déclenchée. Vous pouvez cocher un ou plusieurs jour(s) de la semaine. Par exemple, vous pouvez utiliser cette option pour définir deux sauvegardes hebdomadaires : une le mercredi et une le vendredi.
- **Tous les *N* mois, *N*e jour à *N*** : cette option permet de programmer la sauvegarde sur une base mensuelle. Saisissez 1 si vous souhaitez une sauvegarde mensuelle. Lorsque vous cochez cette option, vous devez indiquer le jour de chaque mois auquel la sauvegarde doit être déclenchée, ainsi que l'heure de déclenchement.
- **Tous les *N* Mo** : cette option permet de programmer la sauvegarde sur la base de la taille du fichier d'historique des requêtes courant. La sauvegarde est automatiquement déclenchée lorsque le fichier atteint la taille définie. Vous pouvez fixer une taille limite de 1, 10, 100 ou 1000 Mo.

Note En cas de sauvegarde périodique, si le serveur Web n'était pas lancé au moment théorique de la sauvegarde, 4D considère au lancement suivant que la sauvegarde a échoué et applique les paramétrages adéquats, définis dans les Préférences de la base.

12 Langage

Le langage de 4D version 11 a été enrichi de nombreuses nouvelles commandes et constantes et les capacités de plusieurs commandes existantes ont été étendues.

- La prise en charge de l'Unicode dans le langage de 4D entraîne des changements dans la gestion des chaînes de caractères, détaillées dans le paragraphe "Modifications liées à l'Unicode", page 258.
- Les commandes du thème "Sources de données externes" ont été modifiées de manière à prendre en charge le nouveau moteur SQL intégré de 4D.
- Le nouveau thème "SQL" regroupe les commandes relatives au moteur SQL de 4D ainsi qu'à l'activation du serveur SQL intégré.
- Les commandes du thème "Recherches et tris" ont été modifiées afin notamment de tirer parti du nouveau moteur de 4D et des recherches par mots-clés.
- Le paragraphe "Formulaires projet", page 277 détaille les modifications relatives à la prise en charge des nouveaux formulaires projet.
- La gestion du glisser-déposer a été enrichie et étendue dans 4D v11. Le glisser-déposer est désormais pris en charge à la fois par les commandes du thème "Glisser-Déposer" mais également par les commandes du thème "Conteneur de données" (évolution du thème "Presse-Papiers").
- La gestion des menus a été profondément remaniée dans 4D v11 — il est désormais possible par exemple de créer des barres de menus utilisant des menus hiérarchiques. Par conséquent, le thème "Menus" contient de nouvelles commandes et des commandes existantes ont été modifiées.

- Les listes hiérarchiques ont également subi d'importantes modifications. Le thème "Listes hiérarchiques" contient donc de nouvelles commandes et des commandes existantes ont été modifiées. A noter également que plusieurs commandes d'interface existantes fonctionnent désormais avec les listes hiérarchiques (cf. paragraphe "Commandes existantes utilisables avec les listes hiérarchiques", page 324).
- Les list box bénéficiant de nouvelles capacités (contenu basé sur des sélections d'enregistrements), le thème "List box" contient des nouvelles commandes et des modifications ont été apportées à certaines commandes existantes.
- La commande Creer fenetre permet désormais de créer des fenêtres d'aspect "métal" sous Mac OS.
- Le thème "Impressions" accueille deux nouvelles commandes offrant un contrôle accru des tâches d'impression : OUVRIR TACHE IMPRESSION et FERMER TACHE IMPRESSION.
- Le thème "Environnement 4D" contient plusieurs nouvelles commandes utilitaires liées à la maintenance des fichiers des bases 4D et aux composants et diverses commandes existantes ont été modifiées.
- Le thème "Environnement système" comporte les nouvelles commandes Selectionner couleur RVB, LIRE FORMATAGE SYSTEME et les commandes de gestion des écrans fonctionnent désormais sous Windows. En outre, la commande PROPRIETES PLATE FORME a été modifiée.
- Une nouvelle commande a été ajoutée dans le thème "Interface utilisateur" et plusieurs commandes ont été modifiées.
- La commande Selectionner dossier (thème "Documents système") admet un paramètre supplémentaire.
- Le thème "Images" contient de nouvelles commande de gestion des images : LISTE CODECS IMAGES, TRANSFORMER IMAGE, COMBINER IMAGES et CONVERTIR IMAGE.
- Deux commandes du thème "Graphes" permettent de tirer parti du moteur de rendu SVG intégré à 4D.
- Le thème "Chaînes de caractères" a subi diverses modifications liées à la prise en charge de l'Unicode. Il comporte en outre les nouvelles commandes Lire traduction chaine permettant de tirer parti de la

technologie XLIFF et Trouver regex pour la prise en charge des expresssions régulières.

- Les commandes TEXTE VERS BLOB et BLOB vers texte du thème “BLOB” et leurs constantes associées ont été modifiées afin de prendre en charge les textes Unicode.
- La commande EXECUTER METHODE a été ajoutée dans le thème “Langage”.
- Deux commandes du thème “Ressources” ont été modifiées afin qu’elles puissent être utilisées dans le cadre du XLIFF.
- Les commandes ENVOYER PAQUET et RECEVOIR PAQUET du thème “Communications” acceptent désormais des BLOBs en paramètre, et la commande UTILISER FILTRE a été modifiée.
- Diverses modifications ont été apportées au thème “Définition structure” : les commandes Nombre de champs et Nombre de tables ont été renommées et les commandes Est un numero de table valide et Est un numero de champ valide ont été ajoutées. En outre, ce thème comporte plusieurs nouveautés relatives à la gestion des index.
- Le thème “Serveur Web” comporte la nouvelle commande Valider mot de passe digest Web et la commande FIXER RACINE HTML a été modifiée.
- Le thème “Outils” s’enrichit d’une nouvelle commande utilitaire (Choisir) et des commandes liées à la gestion des macros. En outre, ce thème regroupe des commandes auparavant situées dans d’autres thèmes.
- Le thème “Transactions” contient la nouvelle commande Niveau de la transaction permettant de gérer les transactions imbriquées.
- Plusieurs commandes du thème “XML” ont été modifiées et une commande DOM a été ajoutée.
- La commande APPELER WEB SERVICE du thème “Web Services (Client)” a été optimisée.
- Enfin, des modifications limitées ont été apportées à plusieurs commandes ou thèmes de commandes : ces informations sont regroupées dans le paragraphe “Modifications diverses”, page 423.

Modifications liées à l'Unicode

4D v11 propose une prise en charge étendue du jeu de caractères Unicode (cf. paragraphe "Prise en charge de l'Unicode", page 67). En particulier, le langage de programmation utilise désormais ce jeu de caractères pour le traitement et le stockage des expressions.

Cette nouveauté entraîne des modifications au niveau de certaines commandes, notamment les commandes relatives aux chaînes de caractères. Ce paragraphe décrit ces modifications lorsque le mode Unicode est activé pour la base.

Note Il est possible de désactiver le mode Unicode via une nouvelle option des Préférences (cf. paragraphe "Compatibilité de l'Unicode avec les bases 4D", page 68).

Déclaration et type des variables

En mode Unicode, les commandes de déclaration de variables sont modifiées.

- C_TEXTE et TABLEAU TEXTE déclarent des variables et tableaux en utilisant UTF-16.
- C_ALPHA et TABLEAU ALPHA font exactement la même chose que C_TEXTE et TABLEAU TEXTE. Le paramètre de longueur maximale de chaîne est ignoré. Cela signifie notamment que, dans le code suivant :

```
C_ALPHA(1;mavar)
mavar:="abc"
lataille:=Longueur(mavar)
```

  - En mode compatibilité (et dans les versions précédentes de 4D), *lataille* = 1
  - En mode Unicode, dans 4D version 11, *lataille* = 3

4D v11 peut donc retourner un type différent pour les tableaux et les variables alpha en fonction du mode d'exécution de la base. Le tableau suivant résume ces différences exprimées via la fonction **Type**(valeur) :

	Type(x) en mode compatibilité	**Type(x) en mode Unicode (v11)**
C_ALPHA(x;10)	Est une variable chaîne (24)	Est un texte (2)
C_TEXTE(x)	Est un texte (2)	
TABLEAU ALPHA(10;x;1)	Est un tableau chaîne (21)	Est un tableau texte (18)
TABLEAU TEXTE(x;1)	Est un tableau texte (18)	
[Table]Champ_Alpha	Est un champ alpha (0)	
[Table]Champ_Texte	Est un texte (2)	

La commande Lire parametre base permet de connaître par programmation le mode d'exécution courant de la base (cf. paragraphe "FIXER PARAMETRE BASE, Lire parametre base", page 359).

Note Les noms des sémaphores sont encodés en Unicode et leur taille est limitée à 255 caractères (30 caractères dans les versions précédentes de 4D).

Commandes du thème "Chaînes de caractères"

En mode Unicode :

- La commande Caractere ne requiert plus un code ASCII (0 à 255) en paramètre mais une valeur UTF-16 (comprise entre 1 et 65535)
- La commande Code ascii a été renommée Code de caractere. Elle ne retourne plus un code ASCII mais la valeur UTF-16 du premier caractère Unicode.
- Les commandes de manipulation de chaînes (Longueur, Remplacer chaine, Position, Supprimer chaine, Sous chaine, Remplacer caracteres) ne travaillent plus sur la base de positions d'octets dans la chaîne mais des positions de caractères UTF-16.
- L'accès au énième caractère d'une chaîne via la syntaxe machaîne[[n]] retourne le énième caractère UTF-16 et non plus le énième octet.
- Les commandes de conversion de chaînes (Mac vers Windows, Windows vers Mac, Mac vers ISO et ISO vers Mac) ne fonctionnent plus. En effet, ces commandes permettent d'exprimer un texte à l'aide d'un autre jeu de caractères. Or, en mode Unicode, toutes les variables texte sont exprimées dans le jeu de caractères Unicode, il n'est pas possible de le modifier. Par conséquent, ces commandes retournent une chaîne identique à la chaîne passée. De nouvelles commandes permettent la conversion de chaînes (cf. ci-dessous).
- Deux nouvelles commandes ont été ajoutées afin de permettre les conversions de chaînes de caractères entre les divers jeux de caractères : CONVERTIR DEPUIS TEXTE et Convertir vers texte. Ces commandes sont décrites dans le paragraphe "Chaînes de caractères", page 380.

Ces nouveaux mécanismes n'entraînent pas de modification particulière pour les langues dont les caractères sont codés sur un seul octet (langues occidentales) mais simplifiera les traitements dans les langues à caractères codés sur deux octets (telles que le japonais).

Note En Unicode, les codes de caractères suivants sont réservés et ne doivent jamais être inclus dans un texte :
0
65534 (FFFE)
65535 (FFFF)

Commandes du thème "BLOB"

Par compatibilité, les commandes BLOB vers texte et TEXTE VERS BLOB acceptent toujours du texte exprimé en ASCII (Mac Roman) mais, en mode Unicode, peuvent également prendre en charge le texte exprimé en Unicode. Pour permettre ces deux utilisations, de nouvelles constantes ont été ajoutées et les constantes existantes ont été renommées. Pour plus d'informations, reportez-vous au paragraphe "BLOB", page 391.

Commandes du thème "Communications"

Par défaut, en mode Unicode, les commandes de ce thème travaillent avec le jeu de caractères UTF-8. La commande UTILISER FILTRE attend désormais un nom "IANA" comme filtre de caractères.

Note En mode compatibilité, les textes sont toujours limités à 32 Ko.

Sources de données externes

4D v11 est doté d'un moteur SQL natif (cf. chapitre "Utiliser le moteur SQL de 4D", page 205).

Le thème "Sources de données externes", créé dans 4D 2004, contient un ensemble de commandes de haut niveau vous permettant d'accéder aux sources de données compatibles ODBC via le SQL. Certaines de ces commandes ont été modifiées en version 11 afin de vous permettre de les utiliser avec le moteur SQL interne de 4D. Pour cela, la nouvelle constante SQL_INTERNAL a été ajoutée dans le thème "Source de données externes".

En outre, de nouvelles commandes vous permettent désormais d'accéder directement aux sources de données SQL externes disponibles sur le poste et d'exécuter des requêtes depuis l'éditeur de méthodes. Reportez-vous au paragraphe "Connexion directe aux sources de données SQL", page 262.

ODBC LOGIN

ODBC LOGIN {(nomSource;utilisateur;motDePasse)}

La commande ODBC LOGIN permet désormais d'ouvrir une connexion avec le moteur SQL interne de 4D.

Pour ouvrir ce type de connexion, il suffit de passer la constante SQL_INTERNAL dans le paramètre *nomSource*.

Il n'est pas nécessaire d'ouvrir une connexion à l'aide de cette commande si vous avez l'intention d'accéder à l'environnement SQL de 4D via les mots-clés Debut SQL/Fin SQL (cf. paragraphe "Saisir du code SQL dans l'éditeur de méthodes", page 135) ou la commande CHERCHER PAR SQL. En revanche, vous devez initialiser la connexion pour pouvoir utiliser les autres commandes ODBC de 4D.

Une fois la connexion terminée, utilisez la commande ODBC LOGOUT pour la refermer.

▼ Cet exemple initialise une connexion avec le moteur SQL interne de 4D :

ODBC LOGIN(SQL_INTERNAL;$utilisateur;$motdepasse)

ODBC FIXER OPTION

ODBC FIXER OPTION (option; valeur)

Lorsque vous travaillez avec le moteur SQL interne de 4D, l'*option* ODBC Asynchrone est inutile. En effet, ce type de connexion est toujours synchrone.

ODBC EXECUTER

ODBC EXECUTER (instructionSQL{; objetLié}{; objetLié2;...;objetLiéN})

La commande ODBC EXECUTER fonctionne de la même manière avec le moteur SQL interne de 4D qu'avec un moteur externe. A noter cependant les précisions suivantes.

Référencer des expressions 4D

Il est possible de faire référence à tout type d'expression 4D valide (variable, champ, tableau, expression...) dans le paramètre *instructionSQL*. Pour désigner une référence 4D au sein de la requête SQL, vous pouvez utiliser indifféremment l'une des deux notations suivantes :

- placer la référence entre chevrons "<<" et ">>"
- faire précéder la référence de deux-points ":"

Par exemple :

```
C_ALPHA(80;vNom)
vNom:=Demander("Nom :")
ODBC EXECUTER("SELECT age FROM PEOPLE WHERE name=<<vNom>>")
```

équivaut strictement à :

```
C_ALPHA(80;vNom)
vNom:=Demander("Nom :")
ODBC EXECUTER("SELECT age FROM PEOPLE WHERE name= :vNom")
```

Cette notation est optimisée : dans le cas de l'exécution de la requête au sein d'une boucle 4D, seule la référence sera calculée sur le serveur à chaque passage, la requête ne sera évaluée que la première fois.

Récupération des valeurs dans 4D

La récupération dans 4D de valeurs issues de requêtes SQL peut être effectuée de deux manières :

- en utilisant les paramètres supplémentaires de la commande ODBC EXECUTER (solution préconisée par 4D) :

```
ODBC EXECUTER("SELECT ename FROM emp";[Employés]Nom)
```

- en utilisant la clause INTO dans la requête SQL elle-même (solution alternative à réserver aux cas particuliers) :

```
ODBC EXECUTER("SELECT ename FROM emp INTO :[Employés]Nom")
```

Pour plus d'informations sur les interactions avec le serveur SQL de 4D, reportez-vous au paragraphe "Echanger des données entre 4D et le moteur SQL", page 206.

ODBC IMPORTER, ODBC EXPORTER

Ces commandes ne peuvent pas être utilisées en cas de connexion avec le moteur SQL interne de 4D.

SQL

Ce nouveau thème regroupe les commandes spécifiques permettant le démarrage et l'utilisation du nouveau moteur SQL de 4D v11.

Connexion directe aux sources de données SQL

4D v11 vous permet de vous connecter directement via le langage à une source de données ODBC externe et d'exécuter des requêtes SQL au sein d'une structure Debut SQL/Fin SQL.

Le principe d'utilisation est le suivant : la nouvelle commande LISTE SOURCES DONNEES permet d'obtenir la liste des sources de données présentes sur le poste. La commande UTILISER BASE EXTERNE permet ensuite de désigner la source externe à utiliser (requêtes de type *SQL pass-through*). Vous pouvez ensuite exécuter des requêtes SQL dans une structure Debut SQL/Fin SQL sur la source "courante". Pour plus d'informations sur les commandes SQL dans l'éditeur de méthodes, reportez-vous au paragraphe "Saisir du code SQL dans l'éditeur de méthodes", page 135.

LISTE SOURCES DONNEES

LISTE SOURCES DONNEES(typeSource; tabNomsSources; tabPilotes)

Paramètres	Type		Description
typeSource	Entier long	→	Type de source : utilisateur ou système
tabNomsSources	Tab Texte	←	Tableau des noms de sources de données
tabPilotes	Tab Texte	←	Tableau des pilotes des sources

La commande LISTE SOURCES DONNEES retourne dans les tableaux *tabNomsSources* et *tabPilotes* les noms et les pilotes des sources de données de type *typeSource* définies dans le gestionnaire ODBC du système d'exploitation.

Passez dans *typeSource* le type de source de données que vous souhaitez obtenir. Vous pouvez utiliser l'une des constantes suivantes, placées dans le thème "Sources de données externes" :

Constante	Type	Valeur
Source de données utilisateur	Entier long	1
Source de données système	Entier long	2

Note Cette commande ne prend pas en compte les sources de données de type fichier.

La commande remplit et dimensionne les tableaux *tabNomsSources* et *tabPilotes* avec les valeurs correspondantes.

Si la commande est correctement exécutée, la variable système OK prend la valeur 1. Sinon, elle prend la valeur 0 et une erreur est générée.

▼ Exemple utilisant une source de données utilisateur :

TABLEAU TEXTE(tdsn;0)
TABLEAU TEXTE(tdsnPilotes;0)
LISTE SOURCES DONNEES(Source de données utilisateur;tdsn;tdsnPilotes)

Référence : UTILISER BASE EXTERNE

UTILISER BASE EXTERNE

UTILISER BASE EXTERNE (nomSource{;utilisateur{;motDePasse}})

Paramètres	Type		Description
nomSource	Chaîne	→	Nom de la source de données ODBC à laquelle se connecter
utilisateur	Chaîne	→	Nom d'utilisateur
motDePasse	Chaîne	→	Mot de passe de l'utilisateur

La commande UTILISER BASE EXTERNE établit une connexion entre l'application 4D et la source de données désignée par le paramètre *nomSource*.

Note Vous pouvez obtenir la liste des sources de données disponibles sur le poste à l'aide de la commande LISTE SOURCES DONNEES.

Une fois la connexion établie, toutes les instructions SQL exécutées par la suite au sein de structures Debut SQL/Fin SQL seront envoyées à cette source externe (*SQL Pass-through*) dans le process courant, jusqu'à ce que la commande UTILISER BASE INTERNE ou une autre instruction UTILISER BASE EXTERNE soit exécutée.

Passez dans les paramètres *utilisateur* et *motDePasse* les identifiants requis par la source de données, le cas échéant.

Si la commande est correctement exécutée, la variable système OK prend la valeur 1. Sinon, elle prend la valeur 0 et une erreur est générée.

▼ Exemple

```
C_TEXTE(tdsn;tdsnPilotes;0)
LISTE SOURCES DONNEES(1;tdsn;tdsnPilotes) `Sources utilisateur
Si (Chercher dans tableau(tdsn;"emp")#-1)
      `Si la source emp existe bien
  UTILISER BASE EXTERNE("emp";"tiger";"scott")
  Debut SQL
    ... `Instructions SQL
  Fin SQL
Fin de si
```

Référence : LISTE SOURCES DONNEES, UTILISER BASE INTERNE

UTILISER BASE INTERNE

UTILISER BASE INTERNE

Paramètres	Type	Description
Cette commande ne requiert pas de paramètre		

La commande UTILISER BASE INTERNE permet de refermer la connexion externe éventuellement ouverte via la commande UTILISER BASE EXTERNE et de reconnecter l'application à la base de données 4D locale. Les éventuelles instructions SQL exécutées au sein de structures Debut SQL/Fin SQL seront envoyées à la base de données 4D locale.

Si aucune connexion externe n'avait été ouverte à l'aide de la commande UTILISER BASE EXTERNE, la commande ne fait rien.

Si la commande est correctement exécutée, la variable système OK prend la valeur 1. Sinon, elle prend la valeur 0 et une erreur est générée.

Référence : UTILISER BASE EXTERNE

Lire source donnees courante

Lire source donnees courante → Chaîne

Paramètres	Type		Description
Résultat	Chaîne	←	Nom de la source de données en cours d'utilisation

La commande Lire source donnees courante retourne le nom de la source de données courante de l'application. La source de données courante reçoit les requêtes SQL exécutées au sein de structures Debut SQL/Fin SQL.

Lorsque la source de données courante est la base 4D locale, la commande retourne la chaîne ";DB4D_SQL_LOCAL;", correspondant à la valeur de la constante SQL_INTERNAL.

Cette commande vous permet de contrôler la source de données courante, par exemple avant d'exécuter une requête SQL.

Référence : UTILISER BASE EXTERNE, UTILISER BASE INTERNE, LISTE SOURCES DONNEES

CHERCHER PAR SQL

CHERCHER PAR SQL({table; }formuleSQL)

Paramètres	Type		Description
table	Table	→	Table de laquelle retourner une sélection d'enregistrements ou Table par défaut si ce paramètre est omis
formuleSQL	Chaîne	→	Formule de recherche SQL valide représentant la clause WHERE de la requête SELECT

4D v11 est doté d'un moteur SQL natif (cf. chapitre "Utiliser le moteur SQL de 4D", page 205). La nouvelle commande CHERCHER PAR SQL est l'un des moyens permettant de tirer parti de ce moteur.
Les deux autres moyens sont les commandes ODBC (cf. paragraphe "Sources de données externes", page 260) et l'utilisation des balises Debut SQL/Fin SQL dans l'éditeur de méthodes (cf. paragraphe "Saisir du code SQL dans l'éditeur de méthodes", page 135).

La commande CHERCHER PAR SQL exécute une requête SELECT simple qui peut être écrite ainsi :

```
SELECT *
  FROM table
  WHERE <formuleSQL>
```

table est le nom de la table passé en premier paramètre et *formuleSQL* la chaîne de recherche passée en deuxième paramètre.

Par exemple, l'instruction suivante :

```
  CHERCHER PAR SQL([Employees];"name='smith'")
```

équivaut à la requête SQL :

```
  SELECT * FROM Employees WHERE "name='smith'"
```

CHERCHER PAR SQL est semblable à la commande CHERCHER PAR FORMULE.

La commande CHERCHER PAR SQL effectue une recherche parmi les enregistrements de la *table* définie. Elle modifie la sélection courante de *table* pour le process courant et fait du premier enregistrement de la nouvelle sélection le nouvel enregistrement courant.

CHERCHER PAR SQL applique *formuleSQL* à chaque enregistrement de la sélection de la *table*. *formuleSQL* est une expression booléenne qui doit retourner VRAI ou FAUX. Comme vous le savez peut-être, dans la norme SQL-2, une condition de recherche peut avoir un résultat VRAI, FAUX ou NULL. Tous les enregistrements (rows) pour lesquels la condition de recherche retourne VRAI sont inclus dans la nouvelle sélection courante.

L'expression *formuleSQL* peut être simple, comme par exemple la comparaison d'un champ (colonne) à une valeur ; elle peut également être complexe, comme la réalisation d'un calcul. Comme CHERCHER PAR FORMULE, CHERCHER PAR SQL peut évaluer des valeurs dans les tables liées (cf. exemple 4). *formuleSQL* doit être une instruction SQL valide, conforme à la norme SQL-2 et tenant compte des limitations actuelles de l'implémentation SQL dans 4D. Pour plus d'information la prise en charge du SQL dans 4D, reportez-vous au chapitre "Utiliser le moteur SQL de 4D", page 205.

Le paramètre *formuleSQL* peut contenir des références à des expressions 4D. La syntaxe à utiliser est la même que pour les commandes ODBC intégrées ou le code inclus dans les balises Debut SQL/Fin SQL, c'est-à-dire :

<<MaVar>> ou :MaVar

Pour plus d'informations, reportez-vous au paragraphe "Echanger des données entre 4D et le moteur SQL", page 206.

Si le format de la condition de recherche est correct, la variable système *OK* prend la valeur 1. Sinon, elle prend la valeur 0, le résultat de la commande est une sélection vide et une erreur est retournée. Cette erreur peut être interceptée par une méthode installée à l'aide de la commande APPELER SUR ERREUR.

Note Cette commande est compatible avec les commandes FIXER LIMITE RECHERCHE et FIXER DESTINATION RECHERCHE.

A propos des liens

CHERCHER PAR SQL n'utilise pas les liens entre les tables définis dans l'éditeur de structure de 4D. Si vous souhaitez tirer parti des données liées, vous devez ajouter une clause JOIN dans la requête.

Par exemple, considérons la structure suivante, dans laquelle un lien N vers 1 relie les champs [PERSONNES]Ville à [VILLES]Nom :

```
[PERSONNES]
    Nom
    Ville
[VILLES]
    Nom
    Population
```

Avec la commande CHERCHER PAR FORMULE, vous pourriez écrire :

```
CHERCHER PAR FORMULE([PERSONNES];[VILLES]Population>1000)
```

Avec CHERCHER PAR SQL, vous devez écrire l'instruction suivante, que le lien existe ou non :

```
CHERCHER PAR SQL([PERSONNES];"personnes.ville=villes.nom AND
                                                villes.population>1000")
```

Note Les liens 1 vers N et N vers N sont également traités par CHERCHER PAR SQL d'une manière différente de CHERCHER PAR FORMULE.

▼ Cet exemple recherche les bureaux dont les ventes sont supérieures à 100.
La requête SQL est :

```
SELECT *
    FROM BUREAUX
    WHERE VENTES > 100
```

En utilisant la commande CHERCHER PAR SQL :

```
C_ALPHA(30;$formuleRequete)
$formuleRequete:="VENTES > 100"
CHERCHER PAR SQL([BUREAUX];$formuleRequete)
```

▼ Cet exemple recherche les commandes comprises entre 3000 et 4000.
La requête SQL est :

```
SELECT *
    FROM COMMANDES
    WHERE TOTAL BETWEEN 3000 AND 4000
```

En utilisant la commande CHERCHER PAR SQL :

```
C_ALPHA(40;$formuleRequete)
$formuleRequete:="TOTAL BETWEEN 3000 AND 4000"
CHERCHER PAR SQL([VENTES];$formuleRequete)
```

- Cet exemple montre comment trier le résultat de la requête sur un critère spécifique. La requête SQL est :

```
SELECT *
    FROM PERSONNES
    WHERE VILLE ='Paris'
      ORDER BY NOM
```

En utilisant la commande CHERCHER PAR SQL :

```
C_ALPHA(40;$formuleRequete)
$formuleRequete:="VILLE = 'Paris' ORDER BY NOM"
CHERCHER PAR SQL([PERSONNES];$formuleRequete)
```

- Cet exemple montre une requête utilisant des tables liées dans 4D. Via le SQL vous devez utiliser un JOIN pour recréer cette relation.
Considérons les deux tables suivantes :
[FACTURES] avec les champs (colonnes) suivants :
    ID_FACT : Entier long
    DATE_FACT : Date
    TOTAL : Réel

[LIGNES_FACTURES] avec les champs (colonnes) suivants :
    ID_LIGNE : Entier long
    ID_FACT : Entier long
    CODE : Alpha (10)

Un lien de N vers 1 relie le champ [LIGNES_FACTURES]ID_FACT au champ [FACTURES]ID_FACT.

Avec la commande CHERCHER PAR FORMULE, vous pourriez écrire :

```
CHERCHER PAR FORMULE([LIGNES_FACTURES];
                                    ([LIGNES_FACTURES]Code="FX-200")
                                & (Mois de([FACTURES]DATE_FACT)=4))
```

La requête SQL est :

```
SELECT ID_LIGNE
    FROM LIGNES_FACTURES, FACTURES
    WHERE LIGNES_FACTURES.ID_FACT=FACTURES.ID_FACT
      AND LIGNES_FACTURES.CODE='FX-200'
      AND MONTH(FACTURES.DATE_FACT) = 4
```

En utilisant la commande CHERCHER PAR SQL :

```
C_ALPHA(30;$formuleRequete)
$formuleRequete:="LIGNES_FACTURES.ID_FACT=FACTURES.ID_FACT
                                    AND LIGNES_FACTURES.CODE='FX-200'
                                    AND MONTH(FACTURES.DATE_FACT)=4"
CHERCHER PAR SQL([LIGNES_FACTURES];$formuleRequete)
```

Valeur champ Null

Valeur champ Null (champ) → Booléen

Paramètres	Type		Description
champ	champ	→	Champ à évaluer
Résultat	Booléen	←	Vrai = le champ est NULL Faux = le champ n'est pas NULL

La commande Valeur champ Null retourne Vrai si le champ désigné par le paramètre *champ* contient la valeur NULL, et Faux sinon.

La valeur NULL est exploitée par le moteur SQL de 4D. Pour plus d'informations, reportez-vous au paragraphe "Valeurs NULL dans 4D", page 216.

Référence : FIXER VALEUR CHAMP NULL

FIXER VALEUR CHAMP NULL

FIXER VALEUR CHAMP NULL (champ)

Paramètres	Type		Description
champ	champ	→	Champ auquel attribuer la valeur NULL

La commande FIXER VALEUR CHAMP NULL attribue la valeur NULL au champ désigné par le paramètre *champ*.

La valeur NULL est exploitée par le moteur SQL de 4D. Pour plus d'informations, reportez-vous au paragraphe "Valeurs NULL dans 4D", page 216.

A noter qu'il est possible d'interdire la valeur Null pour les champs 4D au niveau de l'éditeur de structure (cf. paragraphe "Refuser l'écriture de la valeur NULL", page 218).

Référence : Valeur champ Null

LIRE DERNIERE ERREUR SQL

LIRE DERNIERE ERREUR SQL(tabCodes; tabComposants; tabLibellés)

Paramètres	Type		Description
tabCodes	Tab num	←	Numéros d'erreurs
tabCompInterne	Tab chaîne	←	Codes de composants internes
tabLibellés	Tab chaîne	←	Libellés d'erreurs

La commande LIRE DERNIERE ERREUR SQL retourne les informations de la "pile" d'erreurs courante relative à l'exécution d'une instruction SQL.
Cette commande doit être appelée dans le cadre d'une méthode d'appel sur erreur installée par la commande APPELER SUR ERREUR.

Les informations sont retournées sous la forme de trois tableaux synchronisés :

- *tabCodes* : ce tableau reçoit la liste des codes d'erreurs générés.
- *tabCompInterne* : ce tableau contient les codes des composants internes associés à chaque erreur.
- *tabLibellés* : ce tableau contient les libellés des erreurs.

La liste des codes d'erreurs et de leurs libellés est fournie dans le paragraphe "Codes d'erreur SQL", page 236.

LANCER SERVEUR SQL

LANCER SERVEUR SQL

Paramètres	Type	Description
Cette commande ne requiert pas de paramètre		

La commande LANCER SERVEUR SQL démarre le serveur SQL intégré de l'application 4D sur laquelle elle a été exécutée. Une fois lancé, le serveur SQL peut répondre aux requêtes SQL reçues sur le port TCP défini dans les Préférences de l'application.

Si le serveur SQL a été correctement lancé, la variable système *OK* prend la valeur 1, sinon elle prend la valeur 0.

Note Cette commande n'influe pas sur le fonctionnement du moteur SQL interne de 4D. Le moteur SQL est toujours disponible pour les requêtes internes.

Référence : ARRETER SERVEUR SQL.

ARRETER SERVEUR SQL

ARRETER SERVEUR SQL

Paramètres	Type	Description
Cette commande ne requiert pas de paramètre		

La commande ARRETER SERVEUR SQL stoppe le serveur SQL intégré de l'application 4D sur laquelle elle a été exécutée.

Si le serveur SQL était lancé, toutes les connexions SQL sont interrompues et le serveur n'accepte plus aucune requête SQL externe. Si le serveur SQL n'était pas lancé, la commande ne fait rien.

Note Cette commande n'influe pas sur le fonctionnement du moteur SQL interne de 4D. Le moteur SQL est toujours disponible pour les requêtes internes.

Référence : LANCER SERVEUR SQL

Recherches et tris

La nouvelle commande FIXER RECHERCHE ET VERROUILLAGE permet de paramétrer le verrouillage des enregistrements trouvés.

La syntaxe et le fonctionnement des commandes CHERCHER PAR FORMULE et CHERCHER PAR FORMULE DANS SELECTION ont été modifiés.

Les commandes CHERCHER PAR TABLEAU et Trouver clef index peuvent désormais fonctionner avec des champs non indexés — la commande Trouver clef index a d'ailleurs été renommée Trouver dans champ.

Enfin, 4D v11 permet d'effectuer des recherches par mots-clés. Cette nouveauté est détaillée dans le paragraphe "Recherches par mots-clés", page 275.

FIXER RECHERCHE ET VERROUILLAGE

FIXER RECHERCHE ET VERROUILLAGE(verrou)

Paramètres	Type		Description
verrou	Booléen	→	Vrai = verrouiller les enregistrements trouvés par les recherches Faux = ne pas verrouiller les enregistrements

La nouvelle commande FIXER RECHERCHE ET VERROUILLAGE vous permet de demander le verrouillage automatique des enregistrements trouvés par toutes les recherches qui suivent l'appel de cette commande dans la transaction courante. Ce mécanisme permet de s'assurer que les enregistrements ne puissent pas être modifiés par un process autre que le process courant entre une recherche et la manipulation des résultats.

Par défaut, les enregistrements trouvés par les recherches ne sont pas verrouillés. Passez Vrai dans le paramètre *verrou* pour activer le verrouillage.

Cette commande doit impérativement être utilisée à l'intérieur d'une transaction. Si elle est appelée hors du contexte d'une transaction, une erreur est générée. Ce principe permet un meilleur contrôle du verrouillage des enregistrements.
Les enregistrements trouvés restent verrouillés tant que la transaction n'a pas été terminée (qu'elle ait été validée ou annulée). A l'issue de la transaction, tous les enregistrements sont déverrouillés.

Le verrouillage des enregistrements est effectif pour toutes les tables dans la transaction courante. Appelez FIXER RECHERCHE ET VERROUILLAGE(Faux) afin de désactiver le mécanisme après usage.

▼ Dans cet exemple, il n'est pas possible de supprimer un client qui aurait été passé de la catégorie "C" à la catégorie "A" dans un autre process entre le CHERCHER et le SUPPRIMER SELECTION :

```
DEBUT TRANSACTION
FIXER RECHERCHE ET VERROUILLAGE(Vrai)
CHERCHER([Clients];[Clients]Catégorie="C")
  ` A cet instant, les enregistrements trouvés sont automatiquement
  ` verrouillés pour tous les autres process
SUPPRIMER SELECTION([Clients])
FIXER RECHERCHE ET VERROUILLAGE(Faux)
VALIDER TRANSACTION
```

CHERCHER PAR FORMULE, CHERCHER PAR FORMULE DANS SELECTION

CHERCHER PAR FORMULE (*table*{; formule})

CHERCHER PAR FORMULE DANS SELECTION(*table*{; formule)

Plusieurs modifications importantes sont à signaler pour ces deux commandes dans 4D v11 :

■ Le paramètre *table* est désormais obligatoire.

- Ces commandes ont été largement optimisées et peuvent notamment tirer parti des index.
  Lorsque le type de requête le permet, ces commandes exécutent des requêtes équivalentes à un CHERCHER. Par exemple, l'instruction CHERCHER PAR FORMULE([matable]; [matable]monchamp=valeur) sera exécutée comme CHERCHER([matable]; [matable]monchamp=valeur), ce qui permettra d'utiliser l'index.
  4D pourra également optimiser les requêtes contenant des parties non "optimisables", en exécutant d'abord les parties optimisables puis en combinant les résultats avec le reste de la requête.
  Par exemple, l'instruction CHERCHER PAR FORMULE([matable]; Longueur(monchamp)=valeur) ne sera pas optimisée. En revanche, CHERCHER PAR FORMULE([matable]; Longueur(monchamp)=valeur1 | monchamp=valeur2) sera partiellement optimisée.
- En client-serveur, ces commandes sont désormais exécutées sur le serveur. A noter qu'en cas d'appel direct de variables dans la formule, la requête est calculée avec la valeur de la variables sur le poste client. Par exemple, l'instruction CHERCHER PAR FORMULE([matable]; [matable]monchamp=mavariable) sera exécutée sur le serveur mais avec le contenu de la variable *mavariable* du client.

CHERCHER PAR TABLEAU

CHERCHER PAR TABLEAU (*champCible*; tableau)

Paramètres	Type		Description
champCible	Champ	→	Champ duquel comparer les valeurs
tableau	Tableau	→	Tableau des valeurs recherchées

La commande CHERCHER PAR TABLEAU peut désormais fonctionner avec un champ non indexé.

Trouver dans champ

Trouver dans champ (*champCible*; valeur) → Entier long

Paramètres	Type		Description
champCible	Champ	→	Champ sur lequel effectuer la recherche
valeur	Champ | Var	→	Valeur à rechercher
		←	Valeur trouvée
Résultat	Entier long	←	Numéro de l'enregistrement trouvé, ou -1 si pas d'enregistrement trouvé

Note La commande Trouver dans champ était nommée Trouver clef index dans les versions précédentes de 4D.

La commande Trouver dans champ peut désormais fonctionner avec un champ non indexé.

Recherches par mots-clés

4D v11 permet d'effectuer des recherches par mots-clés dans les champs alpha et texte. Schématiquement, une recherche par mots-clés recherche des "mots" dans des "textes". Par extension dans 4D, une recherche par mots-clés recherche les enregistrements dont les champs alpha ou texte contiennent une ou plusieurs chaînes de caractères spécifiques.

La recherche porte sur des "mots" entiers mais pas sur des suites de caractères, chiffres ou symboles.

Indexé ou non

La recherche par mots-clés est possible même lorsque les champs alpha ou texte ne sont pas indexés. Cependant, les temps de recherche seront très nettement inférieurs si le champ est indexé, en particulier avec le type "Index de mots-clés". Pour plus d'informations sur ce type d'index, reportez-vous au paragraphe "Index de mots-clés", page 41.

Opérateur de comparaison "Contient mot-clé"

Le nouvel opérateur "contient mot-clé", symbolisé par le caractère **%**, a été créé spécialement pour les recherches par mots-clés.
Cet opérateur permet de tester si un texte contient ou non un "mot", et retourne une valeur booléenne (Vrai ou Faux).

Cet opérateur est disponible dans les commandes de recherche CHERCHER, CHERCHER PAR FORMULE, CHERCHER DANS SELECTION et CHERCHER PAR FORMULE DANS SELECTION.

Note Des recherches par mots-clés peuvent être définies dans la boîte de dialogue standard de recherche à l'aide du nouveau comparateur *contient mot-clé*. Pour plus d'informations, reportez-vous au paragraphe "Requêtes sur les enregistrements", page 107.

▼ L'exemple suivant recherche dans toute la table [Produits] les enregistrements dont le champ Description contient le mot "facile" :

```
CHERCHER([Produits];[Produits]Description%"facile")
```

Règles d'usage des recherches par mots-clés

Voici les règles d'usage et de fonctionnement des recherches par mots-clés :

- **Seuls les mots sont pris en compte par l'opérateur %**
  Les mots sont évalués individuellement et dans leur globalité. L'opérateur % retournera toujours Faux si la recherche porte sur plusieurs mots ou une partie de mot (par exemple une syllabe). Les "mots" sont des chaînes de caractères encadrées par des "séparateurs", qui sont les espaces et les caractères de ponctuation. Les chiffres peuvent être recherchés car ils sont évalués comme des chaînes, toutefois les séparateurs décimaux (. ,) ainsi que les autres symboles (monnaie, température, etc.) seront ignorés.

  Exemples :
  "Alpha Bravo Charlie" % "Alpha" retourne Vrai

  Mais
  "Alpha Bravo Charlie" % "Bravo Charlie" retourne Faux
  "Alpha Bravo Charlie" % "ravo" retourne Faux

- **L'opérateur % ne tient compte ni de la casse des caractères ni des caractères diacritiques**
  Exemples :
  "ALPHA BRAVO" % "alpha" retourne Vrai

  et
  "Alpha Bravo" % "ALPHA" retourne Vrai

  et
  "Chaîne bûche hôte" % "chaine" retourne Vrai

- **L'opérateur % admet des recherches du type "commence par" à l'aide du caractère @ placé à la fin du mot recherché**
  Exemple :
  "Software and Computers" % "Comput@" retourne Vrai

Formulaires projet

Dans 4D v11, il est désormais possible de créer et d'utiliser des formulaires projet. Ces formulaires, non rattachés à une table, constituent des outils précieux pour le développement de composants et la construction d'interfaces utilisateur.

La gestion de ces formulaires est détaillée dans le paragraphe "Formulaires projet", page 143.

Cette section décrit la prise en charge de cette nouveauté par les commandes du langage 4D.

Compatibilité avec les commandes et les événements

La plupart des commandes du langage de 4D relatives aux formulaires peuvent être utilisées avec les formulaires projet. Cependant, certaines commandes ne peuvent pas être utilisées dans ce contexte.

- Les commandes suivantes fonctionnent sans restriction avec les formulaires projet :
  - toutes les commandes du thème "Formulaires" sauf FORMULAIRE ENTREE et FORMULAIRE SORTIE (cf. ci-dessous),
  - Imprimer ligne et UTILISER PARAMETRES IMPRESSION (thème "Impressions"),
  - Creer fenetre formulaire (thème "Fenêtres"),
  - DIALOGUE (thème "Saisie").
- Pour des raisons structurelles, les commandes FORMULAIRE ENTREE et FORMULAIRE SORTIE (thème "Formulaires") ne sont pas compatibles avec les formulaires projet. Si vous passez un formulaire projet à l'une de ces commandes, elle est ignorée.
- La commande Table du formulaire courant (thème "Table") retourne Nil lorsqu'elle est exécutée dans le contexte d'un formulaire projet.
- **Formulaires utilisateurs** : les mécanismes des formulaires utilisateurs ne sont pas compatibles avec les formulaires projet. Les commandes du thème "Formulaires utilisateurs" ne peuvent donc pas être utilisées avec les formulaires projet.

Evénements formulaire

Tous les événements formulaires peuvent être activés dans les formulaires projet, à l'exception des événements spécifiques des formulaires de sortie :

- Sur affichage corps

- Sur ouverture corps
- Sur fermeture corps
- Sur chargement ligne
- Sur entête
- Sur impression corps
- Sur impression sous total
- Sur impression pied de page

Appel des formulaires projet

La plupart des commandes relatives aux formulaires (hors formulaires utilisateurs) acceptent un paramètre *table* facultatif comme premier paramètre. C'est par exemple le cas des commandes LIRE PARAMETRE FORMULAIRE, Creer fenetre formulaire ou DIALOGUE.
Comme un formulaire projet et un formulaire table peuvent avoir le même nom, ce paramètre permet de déterminer le formulaire à utiliser : passez le paramètre *table* lorsque vous souhaitez adresser un formulaire table et ne le passez pas dans le cas d'un formulaire projet.

▼ Dans une base contenant un formulaire projet nommé "LeForm" et un formulaire table du même nom pour la table [Table1] :

DIALOGUE([Table1];"LeForm") `4D utilise le formulaire table
DIALOGUE("LeForm") `4D utilise le formulaire projet

▼ Dans une base contenant un formulaire projet nommé "LeForm"mais pas de formulaire table "LeForm" pour la table [Table1] :

TABLE PAR DEFAUT([Table1])
DIALOGUE("LeForm") `4D utilise le formulaire projet

Ce principe est toutefois caduc lorsque la commande TABLE PAR DEFAUT est exécutée et que la base contient un formulaire projet et un formulaire table du même nom. En effet, dans ce cas 4D utilisera le formulaire table de la table par défaut, même si le paramètre *table* n'est pas passé. Dans ce cas, pour garantir l'utilisation de formulaires projet, il suffit d'utiliser la nouvelle commande PAS DE TABLE PAR DEFAUT (décrite ci-dessous).

PAS DE TABLE PAR DEFAUT

PAS DE TABLE PAR DEFAUT

Paramètres	Type	Description
Cette commande ne requiert pas de paramètre		

La commande PAS DE TABLE PAR DEFAUT permet d'annuler l'effet de la commande TABLE PAR DEFAUT. Après l'exécution de cette commande, il n'y a plus de table par défaut définie pour le process.

Si la commande TABLE PAR DEFAUT n'a pas été appelée auparavant, cette commande ne fait rien.

▼ Dans une base contenant un formulaire projet nommé "LeForm" et un formulaire table du même nom pour la table [Table1] :

```
TABLE PAR DEFAUT([Table1])
DIALOGUE("LeForm") `4D utilise le formulaire table

PAS DE TABLE PAR DEFAUT
DIALOGUE("LeForm") `4D utilise le formulaire projet
```

Thème : Table.

Glisser-Déposer

La prise en charge du glisser-déposer a été étendue dans 4D v11, tout en restant compatible avec les mécanismes existants.
Le glisser-déposer automatique est désormais configurable. De nouveaux événements et commandes permettent de mettre en place des interfaces utilisant notamment le glisser-déposer de documents du Bureau vers les formulaires 4D ou inversement.

La plupart des nouvelles fonctions de glisser-déposer sont prises en charge via les commandes du thème "Conteneur de données" (nouveau nom du thème "Presse-papiers" dans 4D v11). Les commandes de ce thème ont été modifiées et de nouvelles commandes ont été ajoutées. Ces nouveautés sont détaillées dans le paragraphe "Conteneur de données", page 285.

Note De nouvelles fonctions intégrées permettent également de gérer le déplacement d'objets en mode Développement (menus, énumérations, etc.) entre deux bases de données 4D par glisser-déposer ou copier-coller. Ce point est traité dans le paragraphe "Déplacement d'objets en mode Développement", page 84.

Vue d'ensemble des nouvelles fonctions

4D v11 élargit le champ d'action des interfaces utilisant le glisser-déposer :

- De nouvelles options permettent de contrôler le glisser-déposer automatique dans les zones en texte natif.
- Il est désormais possible de glisser-déposer des objets entre les formulaires 4D et d'autres applications ou le bureau du système d'exploitation, ou d'effectuer l'opération inverse.
  Par exemple, il est possible de glisser-déposer un fichier image GIF dans un champ image 4D. Il est également possible de sélectionner du texte dans une application de traitement de texte et de le déposer dans une variable texte de 4D.
- Il est possible de déposer des objets directement sur l'application 4D, sans qu'un formulaire soit nécessairement au premier plan. La nouvelle méthode base Sur déposer permet dans ce cas de gérer le glisser-déposer. Ce principe permet par exemple d'ouvrir un document 4D Write en le déposant sur l'icône de l'application 4D.

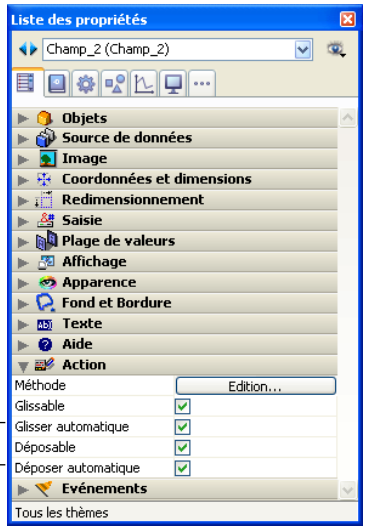
Note Dans 4D v11, la gestion graphique du glisser standard (cadre pointillé autour de l'objet déplacé) n'est plus prise en charge.

Options de glisser-déposer pour les zones natives texte et image

La gestion du glisser-déposer automatique (appelé "glisser-déposer système" dans les versions précédentes de 4D) a été modifiée dans 4D v11 afin de permettre un contrôle accru sur les mécanismes mis en oeuvre.

Le glisser-déposer automatique permet de copier directement du texte ou des images entre deux zones de formulaires. Ce type de glisser-déposer ne peut être utilisé qu'entre des objets simples en contrôle natif, c'est-à-dire les **champs**, les **variables** et les **combo box**.

De nouvelles options de contrôle du glisser-déposer ont été ajoutées pour ces objets. Elles sont accessibles pour chacun de ces objets dans la Liste des propriétés de l'éditeur de formulaires, thème "Action" :



Glissable

Cette option était déjà présente dans les versions précédentes de 4D. Lorsqu'elle est cochée, le nouvel événement formulaire Sur début glisser est généré (cf. paragraphe "Nouvel événement formulaire Sur début glisser", page 284) en cas de glisser de l'objet, hormis lorsque le glisser automatique est activé (cf. ci-dessous). Dans le cas des variables et champs image, l'image et sa référence sont glissées.
Pour activer le glisser standard, appuyez sur la touche **Alt** (Windows) ou **Option** (Mac OS) pendant l'opération. Dans le cas des images, seule la référence du champ ou de la variable est alors glissée.

Glisser automatique

Lorsque cette option est cochée, le mode de glisser automatique est activé (équivaut au fonctionnement précédent de 4D) pour les zones de texte. Ce mode est prioritaire, même si l'option **Glissable** est cochée. Pour "forcer" le glisser standard dans ce cas, appuyez sur la touche **Alt** (Windows) ou **Option** (Mac OS) pendant l'opération.

Note Le glisser automatique provoque le déplacement de la sélection de texte. Pour effectuer une copie de la sélection, appuyez sur la touche **Ctrl** (Windows) ou **Commande** (Mac OS) pendant l'opération.

Par défaut, cette option est cochée pour les bases de données converties depuis une version précédente de 4D.

Déposable

Cette option était déjà présente dans les versions précédentes de 4D. Elle indique que l'objet accepte le déposer de données et provoque l'appel des événements formulaire Sur glisser et Sur déposer, quel que soit le type des données glissées.

Toutefois, si la nouvelle préférence de compatibilité "Interdire de glisser des données ne provenant pas de 4D" est cochée, en cas de glisser d'un objet externe à 4D le déposer est refusé et les événements ne sont pas générés. Pour plus d'informations, reportez-vous au paragraphe "Nouvelle préférence de compatibilité", page 283.

Déposer automatique

Cette option permet d'activer le mode de déposer automatique. Dans ce mode, 4D gère automatiquement — si possible — l'insertion des données glissées de type texte ou image et déposées sur l'objet (les données sont collées dans l'objet). Les événements Sur glisser et Sur déposer dans ce cas ne sont pas générés.
En cas de déposer de données autres que du texte ou des images (autre objet 4D, fichier, etc.) ou de données complexes, l'application se réfère à la valeur de l'option **Déposable** : si elle est cochée, les événements Sur glisser et Sur déposer sont générés, dans le cas contraire le déposer est refusé. Ce principe dépend également de la valeur de l'option "Interdire de glisser des données ne provenant pas de 4D".

Position deposer

Position deposer {(numColonne)} → Numérique

Paramètres	Type		Description
numColonne	Entier long	←	Numéro de colonne de list box ou -1 si le déposer a lieu après la dernière colonne
Résultat	Numérique	←	• Numéro (tableau/list box), ou • Position (liste), *ou* • *Position dans la chaîne (texte/combo box)* de l'élément de destination, ou • -1 si le déposer a eu lieu après le dernier élément de tableau ou de liste

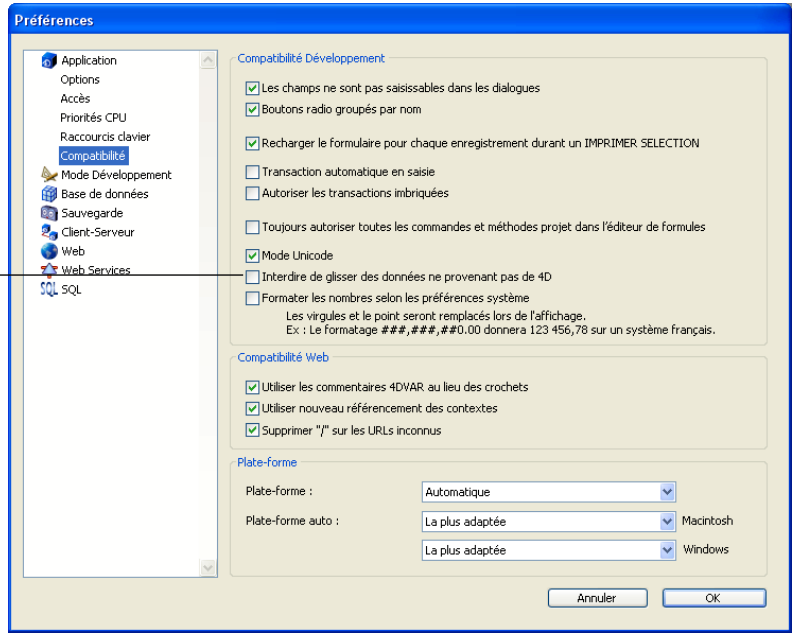
La commande Position deposer fonctionne désormais avec les variables et les champs de type texte, ainsi que les combo box. Dans ce contexte, la commande retourne la position du caractère auquel le déposer a lieu à l'intérieur de la chaîne.

Nouvelle préférence de compatibilité

4D v11 permet le glisser-déposer de sélections, d'objets ou de fichiers extérieurs à 4D, comme par exemple des fichiers image. Cette possibilité doit être prise en charge par le code de la base.

Dans les bases de données converties depuis une version précédente de 4D, cette possibilité peut entraîner des dysfonctionnements si le code existant n'est pas adapté.

Pour cette raison, une nouvelle option des Préférences permet d'interdire le déposer d'objets externes dans la base : **Interdire de glisser des données ne provenant pas de 4D**. Cette option est placée dans la page **Application/Compatibilité** :



Option de gestion du glisser-déposer externe

Lorsque cette option est cochée, le déposer d'objets externes est refusé dans les formulaires 4D. A noter toutefois que l'insertion d'objets externes reste possible dans les objets disposant de l'option **Déposer automatique**, lorsque l'application peut interpréter les données déposées (texte ou image). Pour plus d'informations, reportez-vous au paragraphe "Déposer automatique", page 282.

Cette option est cochée par défaut dans les bases de données converties et non cochée par défaut pour les bases de données créées en version 11.

Nouvel événement formulaire Sur début glisser

Le nouvel événement formulaire **Sur début glisser** est disponible dans 4D v11. Cet événement est sélectionnable pour tous les objets de formulaire pouvant être glissés. Il est généré dans tous les cas lorsque l'objet dispose de la propriété **Glissable**.

A la différence de l'événement formulaire existant Sur glisser, **Sur début glisser** appelle la méthode de l'objet source ou la méthode du formulaire de l'objet source.
En cas de glisser-déposer interprocess, comme Sur glisser, **Sur début glisser** est généré dans le contexte de l'objet source.

Ce nouvel événement est utile pour la gestion avancée du glisser. Il permet de :

- lire les données et les signatures présentes dans le conteneur (via la commande LIRE DONNEES CONTENEUR).
- ajouter des données et des signatures dans le conteneur (via la commande AJOUTER DONNEES AU CONTENEUR).
- accepter ou refuser le glisser via $0 dans la méthode de l'objet glissé. Passer $0=0 pour accepter le glisser ou $0=-1 pour refuser le glisser.

Les données de 4D sont placées dans le conteneur de données avant l'appel de l'événement. Par exemple, pour un glisser sans l'option **Glisser automatique**, le texte glissé est déjà dans le conteneur au moment de l'appel de l'événement.

Nouvelle Méthode base Sur déposer

Une nouvelle méthode base Sur déposer a été ajoutée dans 4D v11. Elle est disponible dans les applications 4D monopostes et 4D Client.

Cette méthode base est exécutée automatiquement en cas de déposer d'objets dans l'application 4D en-dehors de tout formulaire ou fenêtre, c'est-à-dire :

- dans une zone vide de la fenêtre MDI (Windows),
- sur l'icône 4D dans le Dock (Mac OS) ou sur le bureau du système.

Lors d'un déposer sur l'icône de l'application 4D sur le bureau, la méthode base Sur déposer est appelée uniquement si l'application est déjà lancée, sauf dans le cas des applications fusionnées avec 4D Desktop. Dans ce cas la méthode base est appelée même lorsqu'elle n'est pas lancée. Ce principe permet de définir des signatures de documents personnalisées.

▼ Cet exemple permet d'ouvrir un document 4D Write déposé en-dehors de tout formulaire :

```
  `Méthode base Sur Déposer
fichierDéposé:=Lire fichier dans conteneur(1)
Si(Position(".4W7";fichierDéposé)=Longueur(fichierDéposé)-3)
  zexterne:=Creer fenetre externe(100;100;500;500;0;fichierDéposé;
                                                    "_4D Write")
  WR OUVRIR DOCUMENT(zexterne;fichierDéposé)
Fin de si
```

Conteneur de données

Le thème "Conteneur de données" est le nouveau nom du thème "Presse-Papiers" dans 4D v11. Ce changement a été effectué afin de refléter les nouvelles capacités des commandes de ce thème en matière de gestion du glisser-déposer.

En effet, dans 4D v11 les commandes de ce thème prennent en charge les nouvelles fonctions relatives au glisser-déposer. Les nouvelles possibilités de glisser-déposer sont décrites dans le paragraphe "Glisser-Déposer", page 279.

Les commandes du thème "Conteneur de données" ont également été renommées et de nouvelles commandes ont été ajoutées.

Conteneurs de copier-coller et de glisser-déposer

Les commandes du thème "Conteneur de données" peuvent désormais être utilisées pour gérer le glisser-déposer en plus du copier-coller.

Pour cela, 4D v11 exploite deux conteneurs de données : l'un pour les données copiées (ou coupées), qui est en fait celui du Presse-papiers, déjà présent dans les versions précédentes de 4D et l'autre pour les données en cours de glisser-déposer.

Ces deux conteneurs sont gérés à l'aide des mêmes commandes. Vous accédez à l'un ou à l'autre en fonction du contexte :

- Le conteneur de glisser-déposer est accessible uniquement dans le cadre des événements formulaire Sur début glisser, Sur glisser ou Sur déposer et dans la Méthode base Sur déposer.
  En-dehors de ces contextes, le conteneur de glisser-déposer n'est pas disponible.
- Le conteneur de copier-coller est accessible dans tous les autres cas. A la différence du conteneur de glisser-déposer, il conserve durant la

session les données qui y ont été placées, tant qu'il n'a pas été effacé ou réutilisé.

Noms des commandes

Afin de refléter leurs nouvelles capacités liées au glisser-déposer, toutes les commandes du thème "Presse-Papiers" ont été renommées. La plupart ont vu leurs capacités étendues. Le tableau suivant liste ces changements de noms :

Ancien nom (4D 2004.x)	Nouveau nom (4D v11.x)
Tester presse papiers	**Tester conteneur**
LIRE PRESSE PAPIERS	**LIRE DONNEES CONTENEUR**
EFFACER PRESSE PAPIERS	**EFFACER CONTENEUR**
AJOUTER A PRESSE PAPIERS	**AJOUTER DONNEES AU CONTENEUR**
ECRIRE IMAGE DANS PRESSE PAPIERS	**FIXER IMAGE DANS CONTENEUR**
LIRE IMAGE DANS PRESSE PAPIERS	**LIRE IMAGE DANS CONTENEUR**
ECRIRE TEXTE DANS PRESSE PAPIERS	**FIXER TEXTE DANS CONTENEUR**
Lire texte dans presse papiers	**Lire texte dans conteneur**

Noms des constantes

Par soucis de cohérence, les modifications suivantes ont été apportées aux constantes associées :

- le thème "Presse-Papiers" a été renommé "**Conteneur de données**".
- la constante Données absentes presse papiers a été renommée **Données absentes conteneur**.

Types de données

Lors du glisser-déposer, différents types de données peuvent être placés et lus dans le conteneur de données. Vous pouvez accéder à un type de données de plusieurs manières :

- via sa **signature 4D** : la signature 4D est une chaîne de caractères indiquant un type de données référencé par 4D. L'emploi de signatures 4D facilite le développement d'application multi plates-formes, car ces signatures sont identiques sous Mac OS et Windows.
  Vous trouverez ci-dessous la liste des signatures 4D.
- via un **UTI** (*Uniform Type Identifier*, Mac OS uniquement) : la norme UTI, définie par Apple, associe une chaîne de caractères à chaque type d'objet natif. Par exemple, les images GIF ont le type UTI

"com.apple.gif". Les types UTI sont publiés dans les documentations Apple ainsi que par les éditeurs concernés.

- via son **numéro** ou son **nom de format** (Windows uniquement) : sous Windows, chaque type de donnée natif est référencé par un numéro ("3", "12", etc.) et un nom ("Rich Text Edit"). Par défaut, Microsoft définit plusieurs types natifs appelés *formats de données standard*. En outre, tout éditeur tiers peut "enregistrer" des noms de formats auprès du système, qui leur attribue un numéro en retour. Pour plus d'informations sur ce principe et sur les types natifs, reportez-vous à la documentation développeur de Microsoft (en particulier http://msdn2.microsoft.com/en-us/library/ms649013.aspx).

Note Dans les commandes de 4D, les numéros de formats Windows sont manipulés sous forme de textes.

Toutes les commandes du thème "Conteneur de données" peuvent travailler avec chacun de ces types de données. Vous pouvez connaître les types de données présents dans le conteneur dans chacun de ces formats à l'aide de la nouvelle commande LIRE TYPE DONNEES DANS CONTENEUR.

Note Les types sur 4 caractères (TEXT, PICT ou types personnalisés) sont conservés par compatibilité avec les versions précédentes de 4D.

Signatures 4D

Voici la liste des signatures 4D standard ainsi que leur description :

Signature	Description
"com.4d.private.text.native"	Texte en jeu de caractères natif
"com.4d.private.text.utf16"	Texte en jeu de caractères unicode
"com.4d.private.text.rtf"	Texte enrichi
"com.4d.private.picture.pict"	Image format PICT
"com.4d.private.picture.pgn"	Image format PGN
"com.4d.private.picture.gif"	Image format GIF
"com.4d.private.picture.jfif"	Image format JPEG
"com.4d.private.picture.emf"	Image format EMF
"com.4d.private.picture.bitmap"	Image format BITMAP
"com.4d.private.picture.tiff"	Image format TIFF
"com.4d.private.picture.pdf"	Document PDF
"com.4d.private.file.url"	Chemin d'accès de fichier

Commandes modifiées

La plupart des commandes existantes du thème “Presse-Papiers” ont été modifiées afin de gérer les données issues du glisser-déposer.

LIRE DONNEES CONTENEUR

LIRE DONNEES CONTENEUR(typeDonnées; données)

Paramètres	Type		Description
typeDonnées	Alpha	→	Type de données du conteneur *ou signature 4D pour le glisser-déposer*
données	BLOB	←	Données extraites du conteneur

Note Dans les versions précédentes de 4D, cette commande était intitulée LIRE PRESSE PAPIERS.

La commande LIRE DONNEES CONTENEUR peut désormais retourner des données issues d’un glisser-déposer.

Dans le cadre d’un glisser-déposer, vous pouvez passer dans *typeDonnées* l’un des types définis dans le paragraphe “Types de données”, page 286.

Il n’est pas possible de lire les données de type fichier avec cette commande, vous devez utiliser la commande Lire fichier dans conteneur.

AJOUTER DONNEES AU CONTENEUR

AJOUTER DONNEES AU CONTENEUR(typeDonnées; données)

Paramètres	Type		Description
typeDonnées	Alpha	→	Type de données (4 caractères) du conteneur *ou signature 4D pour le glisser-déposer*
données	BLOB	→	Données à ajouter au conteneur

Note Dans les versions précédentes de 4D, cette commande était intitulée AJOUTER A PRESSE PAPIERS.

La commande AJOUTER DONNEES AU CONTENEUR peut désormais ajouter des données dans le contexte d’un glisser-déposer.

Dans le cadre d’un glisser-déposer, vous pouvez passer dans *typeDonnées* l’un des types définis dans le paragraphe “Types de données”, page 286.

Note Le conteneur de données est en lecture seule pendant l'événement formulaire Sur glisser. Il n'est pas possible d'utiliser cette commande dans ce contexte.

Lire texte dans conteneur

Lire texte dans conteneur → Chaîne

Paramètres	Type		Description
Résultat	Chaîne	←	Texte éventuellement présent dans le conteneur de données

Note Dans les versions précédentes de 4D, cette commande était intitulée Lire texte dans presse papiers.

La commande Lire texte dans conteneur peut peut être utilisée pour les données texte du conteneur dans le contexte d'un glisser-déposer.

Note Dans 4D v11, la taille des données de type texte peut désormais atteindre 2 Go.

Dans le cas où le conteneur de données contient du texte enrichi (par exemple au format RTF), le texte est déposé avec ses attributs si la zone de déposer est compatible.

FIXER TEXTE DANS CONTENEUR

FIXER TEXTE DANS CONTENEUR(texte)

Paramètres	Type		Description
texte	Chaîne	→	Texte à placer dans le conteneur de données

Note Dans les versions précédentes de 4D, cette commande était intitulée ECRIRE TEXTE DANS PRESSE PAPIERS.

La commande FIXER TEXTE DANS CONTENEUR peut être utilisée pour placer du texte dans le conteneur dans le contexte d'un glisser-déposer.

Note Le conteneur de données est en lecture seule pendant l'événement formulaire Sur glisser. Il n'est pas possible d'utiliser cette commande dans ce contexte.

LIRE IMAGE DANS CONTENEUR

LIRE IMAGE DANS CONTENEUR(image)

Paramètres	Type		Description
image	Image	←	Image présente dans le conteneur de données

Note Dans les versions précédentes de 4D, cette commande était intitulée LIRE IMAGE DANS PRESSE PAPIERS.

La commande LIRE IMAGE DANS CONTENEUR peut être utilisée pour retourner l'image présente dans le conteneur, dans le contexte d'un glisser-déposer.

Note Dans 4D v11, les images sont stockées dans leur format natif (jpeg, tif, png...).

Dans le cas où la zone de déposer accepte les formats natifs, l'image conserve son format, sinon elle est convertie au format PICT.

FIXER IMAGE DANS CONTENEUR

FIXER IMAGE DANS CONTENEUR(image)

Paramètres	Type		Description
image	Image	→	Image à placer dans le conteneur de données

Note Dans les versions précédentes de 4D, cette commande était intitulée ECRIRE IMAGE DANS PRESSE PAPIERS.

La commande FIXER IMAGE DANS CONTENEUR peut être utilisée pour placer une image dans le conteneur, dans le contexte d'un glisser-déposer.

Note Le conteneur de données est en lecture seule pendant l'événement formulaire Sur glisser. Il n'est pas possible d'utiliser cette commande dans ce contexte.

Tester conteneur

Tester conteneur (typeDonnées) → Réel

Paramètres	Type		Description
typeDonnées	Chaîne	→	Type de données (4 caractères) du conteneur *ou signature 4D pour le glisser-déposer*
Résultat	Réel	←	Taille (en octets) des données contenues dans le Presse-papiers ou code d'erreur

Note Dans les versions précédentes de 4D, cette commande était intitulée Tester presse papiers.

La commande Tester conteneur peut désormais retourner la taille des données dans le contexte d'un glisser-déposer.
Dans le cadre d'un glisser-déposer, vous pouvez passer dans *typeDonnées* l'un des types définis dans le paragraphe "Types de données", page 286.

EFFACER CONTENEUR

EFFACER CONTENEUR

Paramètres	Type	Description
Cette commande ne requiert pas de paramètre		

Note Dans les versions précédentes de 4D, cette commande était intitulée EFFACER PRESSE PAPIERS.

La commande EFFACER CONTENEUR peut effacer le contenu du conteneur, dans le contexte d'un glisser-déposer.

Nouvelles commandes

Pour les besoins liés à la prise en charge du glisser-déposer, des nouvelles commandes ont été ajoutées au thème "Presse-papiers".

FIXER FICHIER DANS CONTENEUR

FIXER FICHIER DANS CONTENEUR(cheminFichier)

Paramètres	Type		Description
cheminFichier	Texte	→	Chemin d'accès complet de fichier

La commande FIXER FICHIER DANS CONTENEUR ajoute dans le conteneur de données de glisser-déposer le chemin d'accès complet passé dans le paramètre *cheminFichier*.

Cette commande permet de mettre en place des interfaces autorisant le glisser-déposer d'objets 4D vers des fichiers sur le bureau par exemple.

Note Le conteneur de données est en lecture seule pendant l'événement formulaire Sur glisser. Il n'est pas possible d'utiliser cette commande dans ce contexte.

Lire fichier dans conteneur

Lire fichier dans conteneur (indiceN) → Texte

Paramètres	Type		Description
indiceN	Numérique	→	Nième fichier inclus dans le glisser
Résultat	Texte	←	Chemin d'accès de fichier extrait du conteneur de données

La commande Lire fichier dans conteneur retourne le chemin d'accès absolu d'un fichier inclus dans une opération de glisser-déposer. Le paramètre *indiceN* permet de désigner un fichier parmi l'ensemble des fichiers sélectionnés.

S'il n'y a pas de Nième fichier dans le conteneur de données, la commande retourne une chaîne vide.

▼ L'exemple suivant permet de récupérer dans un tableau tous les chemins d'accès des fichiers inclus dans le glisser-déposer :

```
TABLEAU TEXTE($tabFichiers;0)
C_TEXTE($vtfichier)
C_ENTIER($n)
$n:=1
Repeter
   $vtfichier:=Lire fichier dans conteneur($n)
   Si($vtfichier#"")
      AJOUTER A TABLEAU($tabFichiers;$vtfichier)
      $n:=$n+1
   Fin de si
Jusque($vtfichier="")
```

LIRE TYPE DONNEES DANS CONTENEUR

LIRE TYPE DONNEES DANS CONTENEUR(signatures4D; typesNatifs{; nomsFormats})

Paramètres	Type		Description
signatures4D	Tab Texte	←	Signatures 4D des types de données
typesNatifs	Tab Texte	←	Types de données natifs
nomsFormats	Tab Texte	←	Noms des formats (Windows uniquement), chaînes vides sous Mac OS

La commande LIRE TYPE DONNEES DANS CONTENEUR permet d'obtenir la liste des types de données présents dans le conteneur. Cette commande doit généralement être utilisée dans le cadre des événements formulaire Sur déposer ou Sur glisser de l'objet de destination. Elle permet notamment de vérifier la présence d'un type de données spécifique dans le conteneur.

Cette commande retourne les types de données sous plusieurs formes différentes via deux (ou trois) tableaux :

- le tableau *signatures4D* contient les types de données exprimés à l'aide de leur signature 4D interne (par exemple "com.4d.picture.gif"). Si un type de données présent n'est pas reconnu par 4D, une chaîne vide ("") est retournée dans le tableau.
- le tableau *typesNatifs* contient les types de données exprimés à l'aide de leur type natif. Le format des types natifs diffère entre Mac OS et Windows :
  - Sous Mac OS, les types natifs sont exprimés sous forme d'UTI (*Uniform Type Identifier*).
  - Sous Windows, les types natifs sont exprimés sous forme de numéros, chaque numéro étant associé à un nom de format. Le tableau *typesNatifs* contient ces numéros sous forme de chaîne ("3", "12", etc.). Si vous souhaitez utiliser des libellés plus explicites, il est recommandé d'utiliser le tableau facultatif *nomsFormats* (cf. ci-dessous), qui contient le nom de format des types natifs sous Windows.

  Le tableau *typesNatifs* permet de prendre en charge tout type de données présent dans le conteneur, y compris des données dont le type n'est pas référencé par 4D.
- Sous Windows, vous pouvez également passer le tableau *nomsFormats*, qui reçoit les noms des types de données présents dans le conteneur. Les valeurs retournées dans ce tableau peuvent être utilisées par

exemple pour construire un pop up menu de sélection de format. Sous Mac OS, le tableau *nomsFormats* retourne des chaînes vides.

Note Pour plus d'informations sur les types de données pris en charge, reportez-vous au paragraphe "Types de données", page 286.

Menus

La gestion des menus a été profondément modifiée dans 4D v11. Les modifications effectuées répondent à un double objectif :

- permettre la création et la manipulation de menus hiérarchiques,
- offrir un contrôle dynamique plus poussé sur les menus et les lignes de menus.

Pour cela, l'éditeur de menus a été modifié (cf. paragraphe "Editeur de menus", page 175). En outre, le thème "Menus" s'est enrichi de nouvelles commandes et des commandes existantes ont été modifiées.

Présentation

4D v11 permet de créer des menus ou des barres de menus "à la volée", sans qu'il soit nécessaire que ces objets aient été préalablement définis dans l'éditeur de menus. Il est également possible de supprimer à la volée toutes les instances des menus et des barres de menus créées par programmation.

Dans 4D v11, il n'y a plus de différence de nature entre un menu et une barre de menus (lorsque ces objets sont créés par programmation). Il s'agit dans les deux cas de listes de libellés. Seul leur usage diffère. Dans le cas d'une barre, chaque libellé correspond à un menu, composé de libellés. C'est également sur ce principe que repose la définition de menus hiérarchiques : chaque libellé peut à son tour être un menu, et ainsi de suite.

Lorsqu'un menu est créé par programmation via la nouvelle commande Creer menu, toute modification effectuée sur ce menu durant la session est immédiatement répercutée à l'ensemble des instances de ce menu et ce, dans tous les process de la base.

RefMenu

A l'image des listes hiérarchiques, un menu créé par programmation reçoit un **identifiant unique**, grâce auquel il pourra être référencé durant toute la session. Cet identifiant, nommé *RefMenu* dans ce manuel, est un alphanumérique de 16 caractères. Toutes les

commandes du thème “Menus” acceptent soit cet identifiant, soit le numéro du menu (fonctionnement standard, désignant un menu créé en mode Développement).

Compatibilité avec les menus créés en Développement

Les deux types de menus (menus et barres de menus “ancienne génération” créés en mode Développement d’une part, menus créés par programmation avec 4D v11 d’autre part) sont compatibles et peuvent être utilisés simultanément.

Il est possible de combiner les deux types d’objets dans une même barre de menus. Du point de vue de l’utilisateur, leur apparence et leur mode de fonctionnement sont identiques.

Les commandes existantes du thème “Menus” ont été adaptées afin de pouvoir prendre en charge les menus créés par programmation. En particulier, elles acceptent un identifiant unique de menu comme premier paramètre.

En revanche, tous les nouveaux mécanismes ne peuvent être appliqués aux menus “ancienne génération”. Par exemple, la nouvelle commande EFFACER MENU ne supprimera pas un menu de l’éditeur en mode Développement.

ligneMenu=-1

Afin de faciliter la manipulation des lignes de menus, 4D propose un raccourci permettant de désigner la dernière ligne ajoutée au menu : il suffit pour cela de passer -1 dans le paramètre *ligneMenu*. Ce principe est utilisable dans toutes les commandes du thème “Menus” manipulant des lignes de menus.

Nouvelles commandes

Plusieurs nouvelles commandes permettent de gérer les menus dans 4D v11. Ces commandes prennent en charge les menus et barres de menus créés par progammation (paramètre de type *RefMenu*, chaîne de 16 caractères) ou dans l’éditeur de menus (paramètre de type Entier long).

Creer menu

Creer menu {(menu)} → RefMenu

Paramètres	Type		Description
menu	RefMenu | Entier long | Alpha	→	Référence de menu ou Numéro ou Nom de barre de menus
Résultat	RefMenu	←	Référence du menu

La commande Creer menu permet de créer un nouveau menu en mémoire. Ce menu n'existera qu'en mémoire et ne sera pas ajouté dans l'éditeur de menus en mode Développement. Toute modification effectuée sur ce menu durant la session sera immédiatement répercutée à l'ensemble des instances de ce menu et ce, dans tous les process de la base.

La commande retourne un identifiant unique de type *RefMenu* pour le nouveau menu.

- Si vous ne passez pas le paramètre facultatif *menu*, le menu sera créé vide. Vous devrez le construire et le gérer à l'aide des commandes AJOUTER LIGNE MENU, FIXER TEXTE LIGNE MENU, etc.
- Si vous passez le paramètre *menu*, le menu créé sera une copie exacte du menu source désigné par ce paramètre. Toutes les propriétés du menu source, y compris les éventuels sous-menus associés, seront appliquées au nouveau menu. A noter qu'une nouvelle référence *RefMenu* est créée pour le menu source ainsi que pour chaque sous-menu associé.
  Vous pouvez passer dans *menu* soit une référence de menu valide, soit un numéro ou un nom de barre de menus défini en mode Développement. Dans ce dernier cas, le nouveau menu sera constitué des menus et sous-menus de la barre d'origine.

  Un menu créé par cette commande peut être utilisé en tant que barre de menus à l'aide de la commande FIXER BARRE MENUS.

EFFACER MENU

EFFACER MENU(menu)

Paramètres	Type		Description
menu	RefMenu	→	Référence de menu

La commande EFFACER MENU efface de la mémoire le menu dont vous avez passé l'identifiant dans *menu*. Ce menu doit avoir été créé par la commande Creer menu.

La commande efface toutes les instances du *menu* dans toutes les barres de menus et tous les process.

Si le menu appartient à une barre de menus en cours d'utilisation, il continuera à fonctionner mais ne pourra plus être modifié. Il ne sera réellement effacé de la mémoire que lorsque la dernière barre de menus dans laquelle il figure ne sera plus utilisée.

Cette commande peut être appliquée aux menus utilisés comme barres de menus.
Les sous-menus éventuellement utilisés par *menu* ne sont pas effacés. Si vous souhaitez effacer un menu et ses sous-menus, vous devez effacer chaque sous-menu individuellement.

LIRE LIGNES MENU

LIRE LIGNES MENU (menu; tabTitresMenus; tabRefsMenus})

Paramètres	Type		Description
menu	RefMenu | Entier long	→	Référence de menu ou Numéro de menu
tabTitresMenu	Tab alpha (32)	←	Tableau des libellés du menu
tabRefsMenu	Tab alpha (16)	←	Tableau des références du menu

La commande LIRE LIGNES MENU retourne dans les tableaux *tabTitresMenu* et *tabRefsMenu* les libellés et les identifiants de toutes les lignes du menu ou de la barre de menus désigné(e) par le paramètre *menu*.

Vous pouvez passer dans *menu* un identifiant unique de menu (*RefMenu*), un numéro de barre de menus ou une référence de barre de menus obtenue via la commande Lire reference barre menu.

Si aucune référence de menu n'est rattachée à une ligne, une chaîne vide est retournée dans l'élément de tableau correspondant.

▼ Vous souhaitez connaître le contenu de la barre de menus du process courant :

```
TABLEAU ALPHA(32;tabTitresMenu;0)
TABLEAU ALPHA(16;tabRefsMenu;0)
RefBarreMenu:=Lire reference barre menu(Process de premier plan)
LIRE LIGNES MENU(RefBarreMenu;tabTitresMenu;tabRefsMenu)
```

Lire reference barre menu

Lire reference barre menu {(process)} → RefMenu

Paramètres	Type		Description
process	Numérique	→	Numéro de référence du process
Résultat	RefMenu	←	Identifiant de la barre de menus

La commande Lire reference barre menu renvoie l'identifiant unique de la barre de menus courante ou de la barre de menus d'un *process* spécifique.

Si la barre de menus a été créée par la commande Creer menu, cet identifiant correspond à la référence unique du menu créé. Sinon, la commande retourne un identifiant interne spécifique. Dans tous les cas, cet identifiant pourra être utilisé pour référencer la barre de menus par toutes les autres commandes du thème.

Le paramètre *process* permet de désigner le process duquel vous souhaitez obtenir l'identifiant de la barre de menus courante. Si vous omettez ce paramètre, la commande retourne l'identifiant de la barre de menus du process courant.

LIRE ICONE LIGNE MENU

LIRE ICONE LIGNE MENU (menu; ligneMenu; refIcône{; process})

Paramètres	Type		Description
menu	RefMenu | Num	→	Référence de menu ou Numéro de menu
ligneMenu	Numérique	→	Numéro de ligne de menu ou -1 pour la dernière ligne ajoutée au menu
refIcône	Var texte | Var entier long	←	Nom ou numéro d'image de la bibliothèque de l'icône associée à la ligne de menu
process	Numérique	→	Numéro de référence de process

La commande LIRE ICONE LIGNE MENU retourne dans la variable *refIcône* la référence de l'icône éventuellement associée à la ligne de menu désignée par les paramètres *menu* et *ligneMenu*. Cette référence est le nom ou le numéro de l'image dans la bibliothèque d'images.

L'icône associée à une ligne de menu est ajoutée dans la barre d'outils de l'application.

Vous pouvez passer -1 dans *ligneMenu* afin de désigner la dernière ligne ajoutée au *menu*.

Vous pouvez passer dans *menu* un identifiant unique de menu (*RefMenu*) ou un numéro de menu.
Si vous passez un identifiant unique, le paramètre *process* est inutile et sera ignoré s'il est passé.
Si vous passez un numéro de menu, la commande prendra en compte le menu correspondant dans la barre de menus principale du process courant. Si vous souhaitez désigner un autre process, passez son numéro dans le paramètre facultatif *process*.

La commande retourne dans *refIcône* soit le nom, soit le numéro de l'image en fonction du type de la variable passée dans ce paramètre. Si vous n'attribuez pas de type spécifique à la variable *refIcône*, par défaut le nom de l'image est retourné (type texte).

Si aucune icône n'est associée à la ligne, la commande retourne une valeur vide.

Référence : FIXER ICONE LIGNE MENU

FIXER ICONE LIGNE MENU

FIXER ICONE LIGNE MENU (menu; ligneMenu; refIcône{; process})

Paramètres	Type		Description
menu	RefMenu | Num	→	Référence de menu ou Numéro de menu
ligneMenu	Numérique	→	Numéro de ligne de menu ou -1 pour la dernière ligne ajoutée au menu
refIcône	Texte | Entier long	→	Nom ou numéro de l'image de la bibliothèque à associer à la ligne de menu
process	Numérique	→	Numéro de référence de process

La commande FIXER ICONE LIGNE MENU permet de modifier l'icône associée à la ligne de menu désignée par les paramètres *menu* et *ligneMenu*.

Vous pouvez passer -1 dans *ligneMenu* afin de désigner la dernière ligne ajoutée au *menu*.

Vous pouvez passer dans *menu* un identifiant unique de menu (*RefMenu*) ou un numéro de menu.
Si vous passez un identifiant unique, la commande s'appliquera à toutes les instances du menu dans tous les process. Dans ce cas, le paramètre *process* est ignoré s'il est passé.
Si vous passez un numéro de menu, la commande prendra en compte le menu correspondant dans la barre de menus principale du process courant. Si vous souhaitez désigner un autre process, passez son numéro dans le paramètre facultatif *process*.

L'icône associée à une ligne de menu est ajoutée dans la barre d'outils de l'application. L'image sera affichée dans un cadre de 20 x 20 pixels.

Vous pouvez passer dans le paramètre *refIcône* soit le nom soit le numéro de l'image de la bibliothèque devant être utilisée comme icône. Il est généralement préférable d'utiliser le numéro plutôt que le nom, car les numéros d'images sont des identifiants uniques, ce qui n'est pas le cas des noms.

Référence : LIRE ICONE LIGNE MENU

Lire methode ligne menu

Lire methode ligne menu (menu; ligneMenu{; process}) → Chaîne

Paramètres	Type		Description
menu	RefMenu | Num	→	Référence de menu ou Numéro de menu
ligneMenu	Entier long	→	Numéro de ligne de menu ou -1 pour la dernière ligne ajoutée au menu
process	Entier long	→	Numéro de référence de process
Résultat	Chaîne	←	Nom de la méthode

La commande Lire methode ligne menu retourne le nom de la méthode projet 4D associée à la ligne de menu désignée par les paramètres *menu* et *ligneMenu*.

Vous pouvez passer -1 dans *ligneMenu* afin de désigner la dernière ligne ajoutée au *menu*.

Vous pouvez passer dans *menu* un identifiant unique de menu (*RefMenu*) ou un numéro de menu.
Si vous passez un identifiant unique, le paramètre *process* est inutile et sera ignoré s'il est passé.
Si vous passez un numéro de menu, la commande prendra en compte le menu correspondant dans la barre de menus principale du process courant. Si vous souhaitez désigner un autre process, passez son numéro dans le paramètre facultatif *process*.

La commande retourne le nom de la méthode 4D sous la forme d'une chaîne de caractères (expression). Si aucune méthode n'est associée à la ligne de menu, la commande retourne une chaîne vide.

Référence : FIXER METHODE LIGNE MENU

FIXER METHODE LIGNE MENU

FIXER METHODE LIGNE MENU (menu; ligneMenu; méthode{; process})

Paramètres	Type		Description
menu	RefMenu | Entier long	→	Référence de menu ou Numéro de menu
ligneMenu	Entier long	→	Numéro de ligne de menu ou -1 pour la dernière ligne ajoutée au menu
méthode	Chaîne	→	Nom de la méthode
process	Entier long	→	Numéro de référence de process

La commande FIXER METHODE LIGNE MENU permet de modifier la méthode projet 4D associée à la ligne de menu désignée par les paramètres *menu* et *ligneMenu*.

Vous pouvez passer -1 dans *ligneMenu* afin de désigner la dernière ligne ajoutée au *menu*.

Vous pouvez passer dans *menu* un identifiant unique de menu (*RefMenu*) ou un numéro de menu.
Si vous passez un identifiant unique, la commande s'appliquera à toutes les instances du menu dans tous les process. Dans ce cas, le paramètre *process* est ignoré s'il est passé.
Si vous passez un numéro de menu, la commande s'appliquera au menu correspondant dans la barre de menus principale du process courant. Si vous souhaitez désigner un autre process, passez son numéro dans le paramètre facultatif *process*.

Passez dans *méthode* le nom de la méthode 4D sous la forme d'une chaîne de caractères (expression).

Note Si la ligne de menu correspond au libellé d'un sous-menu hiérarchique, la méthode ne sera pas appelée lorsque la ligne de menu sera sélectionnée.

Référence : Lire methode ligne menu

LIRE PROPRIETE LIGNE MENU

LIRE PROPRIETE LIGNE MENU (menu; ligneMenu; propriété; valeur{; process}) → Entier long

Paramètres	Type		Description
menu	RefMenu | Entier long	→	Référence de menu ou Numéro de menu
ligneMenu	Entier long	→	Numéro de ligne de menu ou -1 pour la dernière ligne ajoutée au menu
propriété	Chaîne	→	Type de propriété
valeur	Expression	←	Valeur de la propriété
process	Entier long	→	Numéro de référence de process

La commande LIRE PROPRIETE LIGNE MENU retourne dans le paramètre *valeur* la valeur courante de la *propriété* de la ligne de menu désignée par les paramètres *menu* et *ligneMenu*.

Vous pouvez passer -1 dans *ligneMenu* afin de désigner la dernière ligne ajoutée au *menu*.

Vous pouvez passer dans *menu* un identifiant unique de menu (*RefMenu*) ou un numéro de menu.
Si vous passez un identifiant unique, le paramètre *process* est inutile et sera ignoré s'il est passé.
Si vous passez un numéro de menu, la commande prendra en compte le menu correspondant dans la barre de menus principale du process courant. Si vous souhaitez désigner un autre process, passez son numéro dans le paramètre facultatif *process*.

Passez dans le paramètre *propriété* la propriété dont vous souhaitez obtenir la *valeur*. Vous pouvez utiliser l'une des constantes du thème "Propriétés des lignes de menu" ou une chaîne correspondant à une propriété personnalisée. Pour plus d'informations sur les propriétés des menus et leurs valeurs, reportez-vous à la description de la commande FIXER PROPRIETE LIGNE MENU.

Référence : FIXER PROPRIETE LIGNE MENU

FIXER PROPRIETE LIGNE MENU

FIXER PROPRIETE LIGNE MENU (menu; ligneMenu; propriété; valeur{; process})

Paramètres	Type		Description
menu	RefMenu | Entier long	→	Référence de menu ou Numéro de menu
ligneMenu	Entier long	→	Numéro de ligne de menu ou -1 pour la dernière ligne ajoutée au menu
propriété	Chaîne	→	Type de propriété
valeur	Expression	→	Valeur de propriété
process	Entier long	→	Numéro de référence de process

La commande FIXER PROPRIETE LIGNE MENU permet de fixer la *valeur* de la *propriété* pour la ligne de menu désignée par les paramètres *menu* et *ligneMenu*.

Vous pouvez passer -1 dans *ligneMenu* afin de désigner la dernière ligne ajoutée au *menu*.

Vous pouvez passer dans *menu* un identifiant unique de menu (*RefMenu*) ou un numéro de menu.
Si vous passez un identifiant unique, la commande s'appliquera à toutes les instances du menu dans tous les process. Dans ce cas, le paramètre *process* est ignoré s'il est passé.
Si vous passez un numéro de menu, la commande s'appliquera au menu correspondant dans la barre de menus principale du process courant. Si vous souhaitez désigner un autre process, passez son numéro dans le paramètre facultatif *process*.

Passez dans le paramètre *propriété* la propriété dont vous souhaitez modifier la valeur et dans *valeur*, la nouvelle valeur.
Pour le paramètre *propriété*, vous pouvez utiliser l'une des constantes du thème "Propriétés des lignes de menu" ou toute valeur personnalisée :

- Propriété standard : les constantes du thème “Propriétés des lignes de menu” ainsi que leurs valeurs possibles sont décrites ci-dessous. A noter que dans le cas de la propriété Action standard associée, vous pouvez passer une des constantes du thème “Valeurs pour actions standard associée” dans le paramètre *valeur*

propriété **Constante**	*valeur* **Valeurs possibles**
Action standard associée Permet d’associer une action standard à la ligne de menu.	0 = Pas d’action 1 = Action ne pas valider 2 = Action valider 3 = Action enregistrement suivant 4 = Action enregistrement précédent 5 = Action premier enregistrement 6 = Action dernier enregistrement 7 = Action supprimer enregistrement 8 = Action page suivante 9 = Action page précédente 10 = Action première page 11 = Action dernière page 12 = Action modifier sous enreg 13 = Action supprimer sous enreg 14 = Action ajouter sous enreg 17 = Action annuler 18 = Action couper 19 = Action copier 20 = Action coller 21 = Action effacer 22 = Action tout sélectionner 23 = Action afficher Presse papiers 26 = Action test application 27 = Action quitter 31 = Action répéter 32 = Action préférences 35 = Retour au mode Développement 36 = Action CSM
Démarrer un process Permet d’activer l’option Démarrer un nouveau process.	0 = Oui 1 = Non
Autorisations d’accès Permet d’affecter un groupe d’accès à la commande.	0 = Sans restriction >0 = Numéro de groupe

Pour plus d'informations sur les propriétés standard des lignes de menus, reportez-vous au chapitre "Créer des menus personnalisés" dans le manuel *Développement*.

- Propriété personnalisée : vous pouvez passer dans *propriété* tout texte personnalisé et lui associer une *valeur* de type texte, numérique ou booléen. Cette *valeur* sera stockée avec l'élément et pourra être récupérée via la commande LIRE PROPRIETE LIGNE MENU.
Vous pouvez utiliser toute chaîne personnalisée dans le paramètre propriété, veillez simplement à ne pas utiliser de libellé utilisé par 4D (par convention, les propriétés définies par 4D débutent par les caractères "4D_").

Note Si la ligne de menu correspond au libellé d'un sous-menu hiérarchique, l'action standard ne sera pas appelée lorsque la ligne de menu sera sélectionnée.

Référence : LIRE PROPRIETE LIGNE MENU

Lire modificateurs ligne menu

Lire modificateurs ligne menu(menu; ligneMenu{; process}) → Nombre

Paramètres	Type		Description
menu	RefMenu | Num	→	Référence de menu ou Numéro de menu
ligneMenu	Entier long	→	Numéro de ligne de menu ou -1 pour la dernière ligne ajoutée au menu
process	Entier long	→	Numéro de référence de process
Résultat	Nombre	←	Touche(s) de modification associée(s) à la ligne de menu

La commande Lire modificateurs ligne menu retourne le ou les modificateur(s) additionnel(s) associé(s) au raccourci standard de la ligne de menu désignée par les paramètres *menu* et *ligneMenu*.

Le raccourci standard est composé de la touche **Commande** (Mac OS) ou **Ctrl** (Windows) et d'une touche personnalisée. Le raccourci standard est géré via les commandes FIXER RACCOURCI LIGNE MENU et Lire touche ligne menu.

Les modificateurs additionnels sont la touche **Majuscule** et la touche **Option** (Mac OS) / **Alt** (Windows). Ces modificateurs ne sont utilisables que si un raccourci standard a été défini au préalable.

La valeur numérique retournée par la commande correspond au code de la ou des touche(s) de modification additionnelles. Les codes des touches sont les suivants :

- Majuscule = 512
- Option (Mac OS) ou Alt (Windows) = 2048

Si les deux touches sont utilisées, leur valeur est cumulée.

Note Vous pouvez évaluer la valeur retournée à l'aide des constantes Masque touche majuscule et Masque touche option du thème "Evénements (Modifiers)".

Si la ligne de menu n'a pas de touche de modification associée, la commande retourne 0.

Vous pouvez passer -1 dans *ligneMenu* afin de désigner la dernière ligne ajoutée au *menu*.

Vous pouvez passer dans *menu* un identifiant unique de menu (*RefMenu*) ou un numéro de menu.
Si vous passez un identifiant unique, le paramètre *process* est inutile et sera ignoré s'il est passé.
Si vous passez un numéro de menu, la commande prendra en compte le menu correspondant dans la barre de menus principale du process courant. Si vous souhaitez désigner un autre process, passez son numéro dans le paramètre facultatif *process*.

▼ Reportez-vous à l'exemple de la commande Lire touche ligne menu.

Référence : FIXER RACCOURCI LIGNE MENU, Lire touche ligne menu.

Pop up menu dynamique

Pop up menu dynamique (menu{; parDéfaut{; coordX; coordY}}) → RefLigne

Paramètres	Type		Description
menu	RefMenu	→	Référence de menu
parDéfaut	RefLigne	→	Référence de l'élément sélectionné par défaut
coordX	Numérique	→	Coordonnée X du coin supérieur gauche
coordY	Numérique	→	Coordonnée Y du coin supérieur gauche
Résultat	RefLigne	←	Référence de l'élément de menu sélectionné

La commande Pop up menu dynamique fait apparaître un pop up menu hiérarchique à l'emplacement courant de la souris ou à l'emplacement défini par les paramètres facultatifs *coordX* et *coordY*.
Le menu hiérarchique utilisé doit avoir été créé à l'aide de la nouvelle commande Creer menu. La référence retournée par Creer menu doit être passée dans le paramètre *menu*.

Conformément aux règles standard d'interface, cette commande doit généralement être appelée en réponse à un clic droit, ou lorsque le bouton reste enfoncé un certain laps de temps (menu contextuel par exemple).

Le paramètre facultatif *parDéfaut* vous permet de définir un élément du pop up menu sélectionné par défaut lorsque celui-ci apparaît. Passez dans ce paramètre une référence d'élément de menu. Cette référence doit avoir été préalablement définie à l'aide de la nouvelle commande FIXER REFERENCE LIGNE MENU.
Si vous ne passez pas ce paramètre, le premier élément du menu sera sélectionné par défaut.

Les paramètres facultatifs *coordX* et *coordY* permettent de désigner l'emplacement du pop up menu à afficher. Passez respectivement dans *coordX* et *coordY* les coordonnées horizontale et verticale du coin supérieur gauche du menu. Ces coordonnées doivent être exprimées en pixels dans le système de coordonnées local au formulaire courant. Ces deux paramètres doivent être passés ensemble ; si un seul est passé, il est ignoré.

Si vous souhaitez afficher un pop up associé à un bouton 3D, il suffit de ne pas passer les paramètres facultatifs *coordX* et *coordY*. Dans ce cas, 4D calcule automatiquement l'emplacement du menu par rapport au bouton en fonction des normes d'interface de la plate-forme courante.

Si une ligne de menu a été sélectionnée, la commande retourne sa référence (telle que définie à l'aide de la nouvelle commande FIXER REFERENCE LIGNE MENU). Sinon, la commande retourne une chaîne vide.

Note La commande existante Pop up menu (thème "Interface utilisateur") permet de créer des pop up menus basés sur du texte. Elle n'est pas modifiée dans 4D v11.

Référence : FIXER REFERENCE LIGNE MENU

FIXER REFERENCE LIGNE MENU

FIXER REFERENCE LIGNE MENU (menu; ligneMenu; refLigne})

Paramètres	Type		Description
menu	RefMenu | Entier long	→	Référence de menu ou Numéro de menu
ligneMenu	Entier long	→	Numéro de ligne de menu ou -1 pour la dernière ligne ajoutée au menu
refLigne	Chaîne	→	Chaîne à associer en tant que référence

La commande FIXER REFERENCE LIGNE MENU vous permet d'associer une référence personnalisée à la ligne de menu désignée par les paramètres *menu* et *ligneMenu*.

Cette référence sera principalement utilisée par la commande Pop up menu dynamique.

Note Il est également possible d'associer des références aux lignes de menus définies dans l'éditeur de menus (cf. paragraphe "Référence de ligne de menu", page 182).

Référence : Pop up menu dynamique

Lire reference ligne menu

Lire reference ligne menu (menu; ligneMenu) → refLigne

Paramètres	Type		Description
menu	RefMenu | Num	→	Référence de menu ou Numéro de menu
ligneMenu	Entier long	→	Numéro de ligne de menu ou -1 pour la dernière ligne ajoutée au menu
Résultat	refLigne	←	Référence de la ligne de menu

La commande Lire reference ligne menu retourne la référence de la ligne de menu désignée par les paramètres *menu* et *ligneMenu*. Cette référence doit avoir été préalablement définie à l'aide de la commande FIXER REFERENCE LIGNE MENU.

Référence : FIXER REFERENCE LIGNE MENU

Lire reference ligne menu selectionnee

Lire reference ligne menu selectionnee → refLigne

Paramètres	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	refLigne	←	Référence de la ligne de menu

La commande Lire reference ligne menu selectionnee retourne la référence de la ligne de menu sélectionnée. Cette référence doit avoir été préalablement définie à l'aide de la commande FIXER REFERENCE LIGNE MENU.

Si aucune ligne de menu n'a été sélectionnée, la commande retourne une chaîne vide "".

Référence : FIXER REFERENCE LIGNE MENU, Menu choisi

Commandes modifiées

Conformément aux principes de gestion des menus dans 4D v11, toutes les commandes existantes du thème "Menus" acceptent désormais une chaîne de 16 caractères retournée par la commande Creer menu comme référence dans le paramètre *menu*. Dans ce cas, la commande s'applique à toutes les instances du menu dans tous les process.

Bien entendu, il est toujours possible de référencer un menu ou une barre via son numéro. Toutefois, cette syntaxe ne permet pas de prendre en charge les sous-menus hiérarchiques. Dans ce cas également, la commande s'applique uniquement à la barre de menus principale du process courant ou du process désigné par le paramètre *process*, le cas échéant.

Cette section détaille uniquement les commandes du thème "Menus" ayant subi des modifications supplémentaires.

Changement des noms des commandes

Pour des raisons de cohérence et de clarté, certaines commandes existantes du thème "Menus" ont été renommées ou déplacées :

Anciens noms (4D 2004.x)	Nouveaux noms (4D v11.x)
CHANGER BARRE	**FIXER BARRE MENUS**
CACHER BARRE DE MENUS	**CACHER BARRE MENUS** *Transférée dans le thème "Interface utilisateur"*

Anciens noms (4D 2004.x)	Nouveaux noms (4D v11.x)
AFFICHER BARRE DE MENUS	**AFFICHER BARRE MENUS** *Transférée dans le thème "Interface utilisateur"*
APPELER SUR A PROPOS	*Transférée dans le thème "Interface utilisateur"*
Menu choisi	*Pas de changement*
Nombre de menus	*Pas de changement*
Nombre de lignes de menus	*Pas de changement*
Titre menu	**Lire titre menu**
Texte ligne menu	**Lire texte ligne menu**
CHANGER TEXTE LIGNE MENU	**FIXER TEXTE LIGNE MENU**
Style ligne menu	**Lire style ligne menu**
CHANGER STYLE LIGNE MENU	**FIXER STYLE LIGNE MENU**
Marque ligne menu	**Lire marque ligne menu**
MARQUER LIGNE MENU	**FIXER MARQUE LIGNE MENU**
Raccourci clavier	**Lire touche ligne menu**
CHANGER RACCOURCI CLAVIER	**FIXER RACCOURCI LIGNE MENU**
INACTIVER LIGNE MENU	*Pas de changement*
ACTIVER LIGNE MENU	*Pas de changement*
AJOUTER LIGNE MENU	*Pas de changement*
INSERER LIGNE MENU	*Pas de changement*
SUPPRIMER LIGNE MENU	*Pas de changement*

FIXER BARRE MENUS

FIXER BARRE MENUS(barre{;process}{;*})

Paramètres	Type		Description
barre	Num | Alpha | *RefMenu*	→	Numéro ou Nom de barre de menus *ou Référence de menu*
process	Entier long	→	Numéro de référence du process
*	*	→	Conserver l'état de la barre de menus

Note Dans les versions précédentes de 4D, cette commande était intitulée CHANGER BARRE.

La commande FIXER BARRE MENUS accepte désormais une référence unique de menu (type *RefMenu*, chaîne de 16 caractères) dans le paramètre *barre*. Dans 4D v11, les menus peuvent être utilisés comme

barres de menus et inversement (cf. paragraphe “Présentation”, page 294). A noter que si vous passez un paramètre *RefMenu* dans *barre*, le paramètre *process* est inutile et sera ignoré.

Vous pouvez générer une référence unique de menu à l’aide de la commande Creer menu.

Menu choisi

Menu choisi *{(sousMenu)}* → Entier long

Paramètres	Type		Description
sousMenu	*RefMenu*	←	*Référence du menu contenant la ligne sélectionnée*
Résultat	Entier long	←	Commande de menu sélectionnée Mot machine haut = n° de menu Mot machine bas = n° de ligne

La commande Menu choisi permet désormais de travailler avec des sous-menus hiérarchiques.

En cas de sélection d’une ligne d’un menu hiérarchique au-delà du premier niveau, la commande retourne dans le paramètre facultatif *sousMenu* la référence (type *RefMenu,* chaîne de 16 caractères) du sous-menu auquel appartient la ligne sélectionnée. Ce paramètre permet de gérer les sous-menus hiérarchiques. Si la commande du menu ne contient pas de sous-menu hiérarchique, ce paramètre reçoit une chaîne vide.

Référence : Lire reference ligne menu selectionnee

FIXER TEXTE LIGNE MENU

FIXER TEXTE LIGNE MENU (menu; ligneMenu; texteLigne{; process})

Paramètres	Type		Description
menu	Num I *RefMenu*	→	Numéro de menu *ou Référence de menu*
ligneMenu	Entier long	→	Numéro de ligne de menu *ou* *-1 pour la dernière ligne ajoutée au menu*
texteLigne	Alpha	→	Nouveau libellé de la ligne de menu
process	Entier long	→	Numéro de référence de process

Note Dans les versions précédentes de 4D, cette commande était intitulée CHANGER TEXTE LIGNE MENU.

La commande FIXER TEXTE LIGNE MENU accepte désormais une chaîne de type *RefMenu* dans le paramètre *menu*. A noter que si vous passez un paramètre *RefMenu* dans *menu*, le paramètre *process* est inutile et sera ignoré.

Conformément au nouveau fonctionnement du paramètre *ligneMenu*, pouvez passer -1 afin de désigner la dernière ligne ajoutée au *menu*.

Pour définir une ligne de menu comme ligne de séparation, passez le caractère "-" dans le paramètre *texteLigne*.

AJOUTER LIGNE MENU

AJOUTER LIGNE MENU (menu; libelléLigne*{; sousMenu*{; process}})

Paramètres	Type		Description
menu	Num | *RefMenu*	→	Numéro de menu *ou Référence de menu*
libelléLigne	Alpha	→	Libellé(s) de la ou des nouvelle(s) ligne(s) de menu
sousMenu	*RefMenu*	→	*Référence du sous-menu associé à la ligne*
process	Entier long	→	Numéro de référence de process

La commande AJOUTER LIGNE MENU accepte désormais une chaîne de type *RefMenu* dans le paramètre *menu* et comporte un nouveau paramètre facultatif, *sousMenu*. A noter que si vous passez un paramètre *RefMenu* dans *menu*, le paramètre *process* est inutile et sera ignoré.

Vous pouvez passer dans le paramètre *sousMenu* une chaîne de type *RefMenu* désignant un menu créé à l'aide de la commande Creer menu. Dans ce cas, ce menu est associé en tant que sous-menu à la ligne ajoutée.

La commande fonctionne également avec une barre de menus créée via la commande Creer menu et installée avec la commande FIXER BARRE MENUS.

INSERER LIGNE MENU

INSERER LIGNE MENU (menu; aprèsLigne; libelléLigne*{; sousMenu*{; process}})

Paramètres	Type		Description
menu	Num | *RefMenu*	→	Numéro de menu *ou Référence de menu*
aprèsLigne	Entier long	→	Numéro de ligne de menu*ou* *-1 pour la dernière ligne ajoutée au menu*
libelléLigne	Alpha	→	Libellé(s) de la ou des nouvelle(s) ligne(s) de menu
sousMenu	*RefMenu*	→	*Référence du sous-menu associé à la ligne*
process	Entier long	→	Numéro de référence de process

La commande INSERER LIGNE MENU accepte désormais une chaîne de type *RefMenu* dans le paramètre *menu* et comporte un nouveau paramètre facultatif, *sousMenu*. A noter que si vous passez un paramètre *RefMenu* dans *menu*, le paramètre *process* est inutile et sera ignoré.

Vous pouvez passer dans le paramètre *sousMenu* une chaîne de type *RefMenu* désignant un menu créé à l'aide de la commande Creer menu. Dans ce cas, ce menu est associé en tant que sous-menu à la ligne insérée. Si le paramètre *libelléLigne* contient plusieurs lignes, le menu est associé à la première ligne insérée.

La commande fonctionne également avec une barre de menus créée via la commande Creer menu et installée avec la commande FIXER BARRE MENUS.

▼ L'exemple suivant crée un menu constitué de deux commandes auxquelles il affecte une méthode :

```
refMenu:=Creer menu
AJOUTER LIGNE MENU(refMenu;"Caractères")
FIXER METHODE LIGNE MENU(refMenu;1;"GestCaracDial")
INSERER LIGNE MENU(refMenu;1;"Paragraphes")
FIXER METHODE LIGNE MENU(refMenu;2;"GestParDial")
```

SUPPRIMER LIGNE MENU

SUPPRIMER LIGNE MENU (menu; ligneMenu{; process})

Paramètres	Type		Description
menu	Num | *RefMenu*	→	Numéro de menu *ou Référence de menu*
ligneMenu	Entier long	→	Numéro de ligne de menu
process	Entier long	→	Numéro de référence de process

La commande SUPPRIMER LIGNE MENU accepte désormais une chaîne de type *RefMenu* dans le paramètre *menu*. A noter que si vous utilisez ce type de paramètre, le paramètre *process* est inutile et sera ignoré.

Si la ligne de menu désignée par *menu* et *ligneMenu* est elle-même un menu créé à l'aide la commande Creer menu, SUPPRIMER LIGNE MENU supprimera uniquement l'instance de *ligneMenu* dans *menu*. Le sous-menu référencé par *ligneMenu* continuera d'exister en mémoire. Vous devez utiliser la nouvelle commande EFFACER MENU afin de supprimer un menu créé en mémoire.

La commande fonctionne également avec une barre de menus créée avec la commande Creer menu et installée avec la commande FIXER BARRE MENUS.

▼ Cet exemple efface le 2e menu de la barre de menus du process courant :

```
SUPPRIMER LIGNE MENU(Lire reference barre de menus;2)
```

▼ Soit la configuration de menus suivante :

- Barre de menus : Fichier/Edition/Ventes/Contacts
- Menu *Ventes* :
  Clients
  Commandes
  Livraisons
- Pour chaque ligne du menu *Ventes*, un sous-menu hiérarchique propose les deux commandes suivantes : Ajouter/Modifier

Vous souhaitez déplacer le sous-menu *Clients* du menu *Ventes* vers le menu *Contacts* :

```
TABLEAU ALPHA(16;$MenusRef;0)
TABLEAU ALPHA(32;$MenusTitres;0)
C_ENTIER LONG($RefMenuVentes;$RefMenuContacts;$PosMenuLiv;
                                                $RefMenuLiv)
LIRE LIGNES MENU($MenusTitres;$MenusRef)
```

```
$RefMenuVentes:=$MenusRef{Chercher dans tableau($MenusTitres;
                                                "Ventes")}
$RefMenuContacts:=$MenusRef{Chercher dans tableau($MenusTitres;
                                                "Contacts")}
LIRE LIGNES MENU($RefMenuVentes;$MenusTitres;$MenusRef)
$PosMenuLiv:=Chercher dans tableau($MenusTitres;"Livraisons")
$RefMenuLiv:=$MenusRef{$PosMenuLiv}
AJOUTER LIGNE MENU($RefMenuContacts;"Clients";$RefMenuLiv)
SUPPRIMER LIGNE MENU($RefMenuVentes;$PosMenuLiv)
```

ACTIVER LIGNE MENU

ACTIVER LIGNE MENU (menu; ligneMenu{; process})

Paramètres	Type		Description
menu	Num | *RefMenu*	→	Numéro de menu *ou Référence de menu*
ligneMenu	Entier long	→	Numéro de ligne de menu *ou -1 pour la dernière ligne ajoutée au menu*
process	Entier long	→	Numéro de référence de process

La commande ACTIVER LIGNE MENU accepte désormais une chaîne de type *RefMenu* dans le paramètre *menu*. A noter que si vous utilisez ce type de paramètre, le paramètre *process* est inutile et sera ignoré.

Conformément au nouveau fonctionnement du paramètre *ligneMenu*, vous pouvez passer -1 afin de désigner la dernière ligne ajoutée au *menu*. Si le paramètre *ligneMenu* est lui-même un menu, toutes les commandes de ce menu et de ses éventuels sous-menus sont activées.

La commande fonctionne également avec une barre de menus créée avec la commande Creer menu et installée avec la commande FIXER BARRE MENUS.

INACTIVER LIGNE MENU

INACTIVER LIGNE MENU (menu; ligneMenu{; process}})

Paramètres	Type		Description
menu	Num | *RefMenu*	→	Numéro de menu *ou Référence de menu*
ligneMenu	Entier long	→	Numéro de ligne de menu *ou -1 pour la dernière ligne ajoutée au menu*
process	Entier long	→	Numéro de référence de process

La commande INACTIVER LIGNE MENU accepte désormais une chaîne de type *RefMenu* dans le paramètre *menu*. A noter que si vous utilisez ce type de paramètre, le paramètre *process* est inutile et sera ignoré.

Conformément au nouveau fonctionnement du paramètre *ligneMenu*, vous pouvez passer -1 afin de désigner la dernière ligne ajoutée au *menu*. Si le paramètre *ligneMenu* est lui-même un menu, toutes les commandes de ce menu et de ses éventuels sous-menus sont inactivées.

La commande fonctionne également avec une barre de menus créée avec la commande Creer menu et installée avec la commande FIXER BARRE MENUS.

FIXER MARQUE LIGNE MENU

FIXER MARQUE LIGNE MENU (menu; ligneMenu; marque{; process}})

Paramètres	Type		Description
menu	Num | *RefMenu*	→	Numéro de menu *ou Référence de menu*
ligneMenu	Entier long	→	Numéro de ligne de menu *ou -1 pour la dernière ligne ajoutée au menu*
marque	Alpha	→	Nouvelle marque de ligne de menu
process	Entier long	→	Numéro de référence de process

Note Dans les versions précédentes de 4D, cette commande était intitulée MARQUER LIGNE MENU.

La commande FIXER MARQUE LIGNE MENU accepte désormais une chaîne de type *RefMenu* dans le paramètre *menu*. A noter que si vous utilisez ce type de paramètre, le paramètre *process* est inutile et sera ignoré.

Conformément au nouveau fonctionnement du paramètre *ligneMenu*, vous pouvez passer -1 afin de désigner la dernière ligne ajoutée au *menu*. Si le paramètre *ligneMenu* est lui-même un menu, la commande sera sans effet (il n'est pas possible d'associer une marque à un sous-menu).

Lire marque ligne menu

Lire marque ligne menu *(*menu; ligneMenu{; process}) → Alpha

Paramètres	Type		Description
menu	Num I *RefMenu*	→	Numéro de menu *ou Référence de menu*
ligneMenu	Entier long	→	Numéro de ligne de menu *ou -1 pour la dernière ligne ajoutée au menu*
process	Entier long	→	Numéro de référence de process
Résultat	Alpha	←	Marque de la ligne de menu

Note Dans les versions précédentes de 4D, cette commande était intitulée Marque ligne menu.

La commande Lire marque ligne menu accepte désormais une chaîne de type *RefMenu* dans le paramètre *menu*. A noter que si vous utilisez ce type de paramètre, le paramètre *process* est inutile et sera ignoré.

Conformément au nouveau fonctionnement du paramètre *ligneMenu*, vous pouvez passer -1 afin de désigner la dernière ligne ajoutée au *menu*. Si le paramètre *ligneMenu* est lui-même un menu, la commande retourne une chaîne vide.

Lire touche ligne menu

Lire touche ligne menu *(*menu; ligneMenu{; process}) → Numérique

Paramètres	Type		Description
menu	Num I *RefMenu*	→	Numéro de menu *ou Référence de menu*
ligneMenu	Entier long	→	Numéro de ligne de menu *ou -1 pour la dernière ligne ajoutée au menu*
process	Entier long	→	Numéro de référence de process
Résultat	Numérique	←	Code de caractère de la touche de raccourci standard associé à la ligne de menu

Note Dans les versions précédentes de 4D, cette commande était intitulée Raccourci clavier.

La commande Lire touche ligne menu accepte désormais une chaîne de type *RefMenu* dans le paramètre *menu*. A noter que si vous utilisez ce type de paramètre, le paramètre *process* est inutile et sera ignoré.

Conformément au nouveau fonctionnement du paramètre *ligneMenu*, vous pouvez passer -1 afin de désigner la dernière ligne ajoutée au *menu*. Si le paramètre *ligneMenu* est lui-même un menu, la commande retourne 0.

- Dans 4D v11, pour obtenir le raccourci clavier associé à une ligne de menu, il est utile de mettre en place une structure de programmation du type suivant :

```
Si(Lire touche ligne menu(monmenu;1) # 0)
  $modifiers:=Lire modificateurs ligne menu(monmenu;1)
  Au cas ou
  : ($modifiers=Masque touche option)
     ...
  : ($modifiers=Masque touche majuscule)
     ...
  : ($modifiers=Masque touche option + Masque touche majuscule)
     ...
  Fin de cas
Fin de si
```

Référence : FIXER RACCOURCI LIGNE MENU, Lire modificateurs ligne menu

FIXER RACCOURCI LIGNE MENU

FIXER RACCOURCI LIGNE MENU (menu; ligneMenu; codeTouche | *nomTouche{; modificateurs}* {; process}})

Paramètres	Type		Description
menu	Num | *RefMenu*	→	Numéro de menu *ou Référence de menu*
ligneMenu	Entier long	→	Numéro de ligne de menu *ou -1 pour la dernière ligne ajoutée au menu*
codeTouche | *nomTouche*	Num | *Texte*	→	Code Ascii du raccourci clavier *ou Lettre du raccourci clavier*
modificateurs	*Entier long*	→	*Modificateur(s) à associer au raccourci (ignoré si codeTouche passé)*
process	Entier long	→	Numéro de référence de process

Note Dans les versions précédentes de 4D, cette commande était intitulée CHANGER RACCOURCI CLAVIER.

La commande FIXER RACCOURCI LIGNE MENU accepte désormais une chaîne de type *RefMenu* dans le paramètre *menu*. A noter que si vous passez un paramètre *RefMenu* dans *menu*, le paramètre *process* est inutile et sera ignoré.

Conformément au nouveau fonctionnement du paramètre *ligneMenu*, vous pouvez passer -1 afin de désigner la dernière ligne ajoutée au *menu*.

Par ailleurs, la commande FIXER RACCOURCI LIGNE MENU admet désormais un texte comme paramètre de définition de la touche de modification (*nomTouche*). Lorsque cette nouvelle syntaxe est utilisée, le paramètre facultatif *modificateurs* permet d'associer un ou deux modificateurs additionnels au raccourci standard.

Vous pouvez passer directement un nom de touche sous forme de chaîne dans le paramètre *nomTouche*, par exemple "U" pour définir le raccourci Ctrl+U (Windows) ou Commande+U (Mac OS).

Lorsque vous utilisez cette syntaxe, vous pouvez également passer le paramètre facultatif *modificateurs* afin d'associer des modificateurs additionnels au raccourci. Vous pouvez ainsi définir des raccourcis du type Ctrl+Alt+Maj+Z (Windows) ou Cmd+Option+Maj+Z (Mac OS). Pour cela, passez dans *modificateurs* les valeurs suivantes :

- 512 pour la touche Majuscule
- 2048 pour la touche Option (Mac OS) ou Alt (Windows)
- Pour associer les deux touches, cumulez leurs valeurs.

A noter que les touches Ctrl (Windows) et Commande (Mac OS) sont automatiquement ajoutées par 4D au raccourci clavier.

Note Vous pouvez définir la valeur à passer à l'aide des constantes Masque touche majuscule et Masque touche option du thème "Evénements (Modifiers)".

Le paramètre *modificateurs* n'est pas pris en compte s'il est passé lorsque la touche de modification est définie via son code Ascii (ancienne syntaxe).

Note de compatibilité La précédente syntaxe (basée sur le code Ascii de la touche) est conservée par compatibilité uniquement. L'usage de la nouvelle syntaxe, permettant notamment de définir des modificateurs additionnels, est désormais fortement recommandé.

▼ Définition du raccourci Ctrl+Maj+U (Windows) et Cmd+Maj+U (Mac OS) pour la ligne “Souligné” :

FIXER TEXTE LIGNE MENU(menuRef;1;"Souligné")
FIXER RACCOURCI LIGNE MENU(menuRef;1;"U";Masque touche majuscule)

Référence : Lire touche ligne menu, Lire modificateurs ligne menu

Listes hiérarchiques

Les listes hiérarchiques ont été structurellement modifiées dans 4D v11 afin de leur procurer davantage de puissance et de souplesse d’utilisation ainsi qu’une gestion plus conforme à celle des autres objets de formulaire de 4D.

Leur fonctionnement dans 4D v11 est globalement identique à celui des versions précédentes. Toutefois, les anciennes limitations disparaissent :

- Il est désormais possible de faire figurer plusieurs listes hiérarchiques dans un même formulaire.
- Il n’est plus nécessaire d’exécuter la commande REDESSINER LISTE pour mettre à jour une liste modifiée ; cette mise à jour est effectuée automatiquement.
- La longueur maximale d’un élément de liste passe de 255 caractères à plus de 2 milliards (taille maximale d’un champ texte).
- Il est désormais possible d’affecter l’attribut “transparent” à une liste.
- Les barres de défilement horizontales sont désormais gérées.

En outre, de nouvelles fonctions sont proposées :

- multi-représentation d’une même liste dans un ou plusieurs formulaires,
- possibilité de modifier certaines propriétés par défaut (sélection, police, barres de défilement, etc.) via les commandes existantes de gestion des objets,
- possibilité de modifier la police d’un élément de liste,
- recherches dans les listes,
- possibilité d’utiliser des variables pour les icônes des éléments.

Ces nouveautés sont décrites dans les paragraphes suivants.

Multi-représentation

4D v11 établit une distinction entre l'*objet de langage* liste hiérarchique (référencé par son identifiant unique **RéfListe**) et sa représentation en tant qu'*objet de formulaire* (référencé par son nom d'objet).

Il n'existe qu'une seule instance en mémoire de l'objet de langage et toute modification effectuée sur cet objet est immédiatement répercutée dans tous les endroits où il est utilisé.

En revanche, il peut exister plusieurs représentations d'une même liste hiérarchique dans un même formulaire ou dans des formulaires différents. Chaque représentation de liste dispose de caractéristiques propres et partage des caractéristiques communes avec l'ensemble des représentations.

Les caractéristiques propres à chaque représentation de liste sont les suivantes :

- la sélection,
- l'état déployé/contracté des éléments,
- la position du curseur de défilement.

Les autres caractéristiques (police, style, filtre de saisie, couleur, contenu de la liste, icônes, etc.) sont communes à toutes les représentations et ne peuvent pas être modifiées séparément.

Nouvelle syntaxe pour les commandes liées à la représentation

La multi-représentation implique la modification de l'utilisation des commandes fonctionnant avec la position des éléments (par exemple SELECTIONNER ELEMENTS PAR POSITION) ou l'élément courant. En effet, lorsqu'une même liste est représentée plusieurs fois, chaque représentation peut disposer de sa propre configuration d'éléments déployés/contractés et d'élément courant. Or, certaines commandes se basent précisément sur cette configuration, par exemple Nombre elements (lorsque le paramètre * final n'est pas passé). Il importe donc de pouvoir désigner sans ambiguïté la représentation à utiliser.

Pour cela, les commandes du thème "Listes hiérarchiques" liées à la représentation acceptent désormais le **nom d'objet** de formulaire de la liste (chaîne) comme premier paramètre au lieu de la référence **RéfListe** (Entier long). La nouvelle syntaxe est identique à celle des commandes s'appliquant aux objets de formulaire :

```
COMMANDE(*;"NomObjet"; paramètres...)
```

Par exemple :
SELECTIONNER ELEMENTS PAR POSITION(*;"maliste1";5;$posTab)

Vous devez utiliser cette syntaxe lorsque vous travaillez avec des listes hiérarchiques multiples dans les formulaires et souhaitez effectuer des actions ou obtenir des informations sur une seule représentation. Dans ce contexte, la syntaxe basée sur la *RéfListe* est à proscrire car elle ne permet pas de cibler précisément une représentation.

Les commandes de liste hiérarchique acceptant la syntaxe objet sont les suivantes :

CHANGER ELEMENT
INFORMATION ELEMENT
SUPPRIMER DANS LISTE (*cf. Note*)
SELECTIONNER ELEMENTS PAR POSITION
Nombre elements
Elements selectionnes
INSERER DANS LISTE (*cf. Note*)
Element parent
Position element liste
CHANGER PROPRIETES ELEMENT
LIRE PROPRIETES ELEMENT
FIXER POLICE ELEMENT (*nouvelle commande 4D v11*)
Lire police element (*nouvelle commande 4D v11*)
Chercher dans liste (*nouvelle commande 4D v11*)
FIXER ICONE ELEMENT (*nouvelle commande 4D v11*)
LIRE ICONE ELEMENT (*nouvelle commande 4D v11*)
FIXER PARAMETRE ELEMENT (*nouvelle commande 4D v11*)
LIRE PARAMETRE ELEMENT (*nouvelle commande 4D v11*)

Note Les commandes SUPPRIMER DANS LISTE et INSERER DANS LISTE sont les nouveaux noms des commandes SUPPRIMER ELEMENT et INSERER ELEMENT. Pour plus d'informations sur ce point, reportez-vous au paragraphe "Changements de noms", page 429.

Attention, dans le cas des commandes définissant des propriétés, la syntaxe basée sur le nom d'objet ne signifie pas que seul l'objet de formulaire désigné sera modifié par la commande, mais que l'action de la commande sera basée sur l'état de cet objet. Les caractérisques communes des listes hiérarchiques sont toujours modifiées dans toutes les représentations (cf. paragraphe "Multi-représentation", page 321). Par exemple, si vous passez l'instruction FIXER POLICE ELEMENT(*;"maliste1";*;lapolice), vous indiquez que vous souhaitez modifier la police d'un élément de la liste hiérarchique associée à

l'objet de formulaire *maliste1*. La commande tiendra compte de l'élément courant de l'objet *maliste1* pour définir l'élément à modifier, mais cette modification sera reportée dans toutes les représentations de la liste dans tous les process.

Prise en compte du @

Comme pour les autres commandes de gestion des propriété d'objets, il est possible d'utiliser le caractère "@" dans le paramètre *NomObjet*. En principe, cette syntaxe permet de désigner un ensemble d'objets dans le formulaire. Toutefois, dans le contexte des commandes de liste hiérarchique, ce principe n'est pas applicable dans tous les cas. Cette syntaxe aura deux effets différents en fonction du type de commande :

- Pour les commandes fixant des propriétés, cette syntaxe désigne tous les objets dont le nom correspond (fonctionnement standard). Par exemple, le paramètre "LH@" désigne tous les objets de type liste hiérarchique dont le nom débute par "LH". Ces commandes sont :
SUPPRIMER DANS LISTE
INSERER DANS LISTE
SELECTIONNER ELEMENTS PAR POSITION
CHANGER ELEMENT
CHANGER PROPRIETES ELEMENT
FIXER ICONE ELEMENT
FIXER POLICE ELEMENT
FIXER PARAMETRE ELEMENT

- Pour les commandes récupérant des propriétés, cette syntaxe désigne le premier objet dont le nom correspond. Ces commandes sont :
Nombre elements
Elements selectionnes
Element parent
Position element liste
LIRE PROPRIETES ELEMENT
INFORMATION ELEMENT
LIRE PARAMETRE ELEMENT
LIRE ICONE ELEMENT
Lire police element
Chercher dans liste

Priorité des commandes de propriété

Certaines propriétés d'une liste hiérarchique (par exemple l'attribut saisissable ou la couleur) peuvent être définies de trois manières : via la Liste des propriétés en mode Développement, via une commande du thème "Propriétés des objets" ou via une commande du thème "Liste hiérarchique".

Lorsque ces trois moyens sont utilisés pour définir les propriétés d'une liste, l'ordre de priorité suivant est appliqué :
1. Commandes du thème "Liste hiérarchique"
2. Commandes de gestion des objets (thème "Propriétés des objets" ou "Interface utilisateur")
3. Paramètres de la Liste des propriétés

Ce principe est appliqué quel que soit l'ordre d'appel des commandes. Si une propriété d'élément est modifiée individuellement via une commande de liste hiérarchique, la commande de propriété d'objet équivalente sera sans effet sur cet élément même si elle est appelée ultérieurement. Par exemple, si vous modifiez la couleur d'un élément via la commande CHANGER PROPRIETES ELEMENT, la commande CHOIX COULEUR n'aura aucun effet sur cet élément.

Commandes existantes utilisables avec les listes hiérarchiques

La gestion de l'interface des listes hiérarchiques dans les formulaires a été enrichie dans 4D v11. Il est désormais possible de modifier l'apparence des listes hiérarchiques à l'aide de plusieurs commandes 4D génériques. Vous devez passer à ces commandes soit le nom d'objet de la liste hiérarchique (en utilisant le paramètre *), soit son nom de variable (syntaxe standard).

Note Dans le cas des listes hiérarchiques, la variable de formulaire contient la valeur de *RéfListe*. Si vous exécutez une commande de modification d'attribut en lui passant la variable associée à la liste hiérarchique, cela revient à lui passer *RéfListe*. Seul le nom d'objet permet donc de désigner une représentation.

CHANGER JEU DE CARACTERES, CHANGER STYLE, CHANGER TAILLE

CHANGER JEU DE CARACTERES({*; }objet; police)
CHANGER STYLE({*; }objet; style)
CHANGER TAILLE({*; }objet; taille)

Les commandes CHANGER JEU DE CARACTERES, CHANGER STYLE et CHANGER TAILLE permettent désormais de modifier la police, le style et la taille de police (respectivement) d'une liste hiérarchique dans un formulaire. La liste et toutes ses sous-listes sont modifiées.

Ces commandes remplacent les propriétés de police correspondantes éventuellement définies pour la liste hiérarchique au niveau du formulaire. En revanche, elles n'auront pas d'effet sur les éléments ayant été éventuellement modifiés par les commandes du thème "Listes hiérarchiques".

Thème : Propriétés des objets.

CHOIX VISIBLE BARRES DEFILEMENT

CHOIX VISIBLE BARRES DEFILEMENT({*; }objet; horizontal; vertical)

Vous pouvez désormais afficher ou masquer la barre de défilement horizontale et/ou verticale d'une liste hiérarchique dans un formulaire à l'aide de la commande CHOIX VISIBLE BARRES DEFILEMENT.

Thème : Propriétés des objets.

DEFILER LIGNES

DEFILER LIGNES({*; }objet{; position{; *}})

La commande DEFILER LIGNES permet désormais de faire défiler les lignes d'une représentation de liste hiérarchique dans un formulaire de manière à afficher le premier élément sélectionné ou un élément spécifique. Si vous utilisez la syntaxe objet, seul l'objet de formulaire correspondant sera modifié (la position du curseur de défilement est une caractéristique propre à chaque représentation de liste).

Si vous utilisez le paramètre *position* pour désigner un numéro de ligne, la commande tient compte de l'état déployé/contracté des éléments.

Thème : Interface utilisateur.

Note Comme dans les versions précédentes de 4D, les commandes CHOIX COULEUR, FIXER COULEURS RVB, CHOIX FILTRE SAISIE et CHOIX SAISISSABLE fonctionnent avec les listes hiérarchiques, en fonction des règles de priorité évoquées ci-dessus.

Commandes modifiées

Les commandes suivantes du thème "Listes hiérarchiques" ont été modifiées dans 4D v11.

REDESSINER LISTE

REDESSINER LISTE(liste)

La commande REDESSINER LISTE est inutile dans 4D v11. Toutes les représentations des listes hiérarchiques sont désormais automatiquement redessinées. Lorsqu'elle est appelée, cette commande ne fait rien.

Nouvelles commandes

Plusieurs nouvelles commandes de gestion des listes hiérarchiques (*RéfListe*) ont été ajoutées dans le thème “Listes hiérarchiques”.

FIXER POLICE ELEMENT

FIXER POLICE ELEMENT({*; }liste; réfElément | *; police)

Paramètres	Type		Description
*	*	→	Si spécifié, liste est un nom d’objet (chaîne) Si omis, liste est un numéro de référence de liste
liste	RefListe	Chaîne	→	Nom d’objet de type liste (si * spécifié) Numéro de référence de liste (si * omis)
réfElément	*	Entier long	*	→	Numéro de référence d’élément ou 0 pour le dernier élément ajouté à la liste ou * pour l’élément courant de la liste
police	Chaîne	Num	→	Nom ou numéro de police

La commande FIXER POLICE ELEMENT modifie la police de caractères de l’élément désigné par le paramètre *réfElément* de la liste dont vous avez passé le numéro de référence ou le nom d’objet dans *liste*.

Si vous passez le premier paramètre optionnel *, vous indiquez que le paramètre *liste* est un nom d’objet (chaîne) correspondant à une représentation de liste dans le formulaire. Si vous ne passez pas ce paramètre, vous indiquez que le paramètre *liste* est une référence de liste hiérarchique (*RefListe*).
Si vous utilisez une seule représentation de liste ou travaillez avec les éléments structurels (le second * est omis), vous pouvez utiliser indifféremment l’une ou l’autre syntaxe.

En revanche, si vous utilisez plusieurs représentations d’une même liste et travaillez avec l’élément courant (le second * est passé), la syntaxe basée sur le nom d’objet est requise car chaque représentation peut disposer de son propre élément courant.

Vous pouvez passer un numéro de référence dans *réfElément*. Si ce numéro ne correspond à aucun élément de la liste, la commande ne fait rien.
Vous pouvez également passer 0 dans *réfElément* afin de demander la modification du dernier élément ajouté à la liste (à l’aide de AJOUTER A LISTE).

Vous pouvez enfin passer * dans *réfElément* : dans ce cas, la commande s'appliquera à l'élément courant de la liste. Si plusieurs éléments sont sélectionnés manuellement, l'élément courant est celui qui a été sélectionné en dernier. Si aucun élément n'est sélectionné, la commande ne fait rien.

Passez dans le paramètre *police* le nom ou le numéro de la police à utiliser. Pour réappliquer la police par défaut de la liste hiérarchique, passez une chaîne vide dans *police*.

▼ Appliquer la police Times à l'élément courant de la liste :

```
FIXER POLICE ELEMENT(*;"Maliste";*;"Times")
```

Référence : Lire police element

Lire police element

Lire police element({*; }liste; réfElément | *) → Chaîne

Paramètres	Type		Description
*	*	→	Si spécifié, liste est un nom d'objet (chaîne) Si omis, liste est un numéro de référence de liste
liste	RefListe | Chaîne	→	Nom d'objet de type liste (si * spécifié) Numéro de référence de liste (si * omis)
réfElément | *	Entier long | *	→	Numéro de référence d'élément ou 0 pour le dernier élément ajouté à la liste ou * pour l'élément courant de la liste
Résultat	Chaîne	←	Nom de police

La commande Lire police element retourne le nom de la police de caractères courante de l'élément désigné par le paramètre *réfElément* de la liste dont vous avez passé le numéro de référence ou le nom d'objet dans *liste*.

Si vous passez le premier paramètre optionnel *, vous indiquez que le paramètre *liste* est un nom d'objet (chaîne) correspondant à une représentation de liste dans le formulaire. Si vous ne passez pas ce paramètre, vous indiquez que le paramètre *liste* est une référence de liste hiérarchique (*RefListe*).
Si vous utilisez une seule représentation de liste ou travaillez avec les éléments structurels (le second * est omis), vous pouvez utiliser indifféremment l'une ou l'autre syntaxe.

En revanche, si vous utilisez plusieurs représentations d'une même liste et travaillez avec l'élément courant (le second * est passé), la syntaxe basée sur le nom d'objet est requise car chaque représentation peut disposer de son propre élément courant.

Vous pouvez passer un numéro de référence dans *réfElément*. Si ce numéro ne correspond à aucun élément de la liste, la commande ne fait rien.
Vous pouvez également passer 0 dans *réfElément* afin de demander la modification du dernier élément ajouté à la liste (à l'aide de AJOUTER A LISTE).
Vous pouvez enfin passer * dans *réfElément* : dans ce cas, la commande s'appliquera à l'élément courant de la liste. Si plusieurs éléments sont sélectionnés manuellement, l'élément courant est celui qui a été sélectionné en dernier. Si aucun élément n'est sélectionné, la commande ne fait rien.

Référence : FIXER POLICE ELEMENT

Chercher dans liste

Chercher dans liste({*; }liste; valeur; portée{; tabEléments}{; *}) → Entier long

Paramètres	Type		Description
*	*	→	Si spécifié, liste est un nom d'objet (chaîne) Si omis, liste est un numéro de référence de liste
liste	RefListe | Chaîne	→	Nom d'objet de type liste (si * spécifié) Numéro de référence de liste (si * omis)
valeur	Chaîne	→	Valeur à rechercher
portée	Entier	→	0=Liste principale, 1=Sous-listes
tabEléments	Tab Entier long	←	• Si 2e * omis : tableau des positions des éléments trouvés • Si 2e * passé : tableau des numéros de référence des éléments trouvés
*	*	→	• Si omis : utiliser la position des éléments • Si passé : utiliser le numéro de référence des éléments

Paramètres	Type		Description
Résultat	Entier long	←	• Si 2e * omis : position de l'élément trouvé • Si 2e * passé : numéro de référence de l'élément trouvé

La commande Chercher dans liste retourne la position ou la référence du premier élément de *liste* qui équivaut à la chaîne passée dans *valeur*. Si plusieurs éléments sont trouvés, la fonction peut également remplir le tableau *tabEléments* avec la position ou la référence de chaque élément.

Si vous passez le premier paramètre optionnel *****, vous indiquez que le paramètre *liste* est un nom d'objet (chaîne) correspondant à une représentation de liste dans le formulaire. Si vous ne passez pas ce paramètre, vous indiquez que le paramètre *liste* est une référence de liste hiérarchique (*RefListe*).
Si vous utilisez une seule représentation de liste ou travaillez avec les références (le second ***** est omis), vous pouvez utiliser indifféremment l'une ou l'autre syntaxe.
En revanche, si vous utilisez plusieurs représentations d'une même liste et travaillez avec les positions (le second ***** est passé), la syntaxe basée sur le nom d'objet est requise car chaque représentation peut disposer de sa propre configuration d'éléments déployés/contractés.

Le second paramètre ***** vous permet d'indiquer si vous souhaitez travailler avec des positions courantes d'éléments (dans ce cas, ce paramètre doit être omis) ou des références absolues d'éléments (dans ce cas, il doit être passé).

Passez dans *valeur* la chaîne de caractères à rechercher. La recherche sera du type "est exactement", c'est-à-dire que la recherche de "bois" ne trouvera pas "boissons". Toutefois, vous pouvez utiliser le caractère @ pour définir des recherches du type "commence par", "se termine par" ou "contient".

Le paramètre *portée* vous permet de définir si la recherche doit porter uniquement sur le premier niveau de la *liste* ou si elle doit inclure toutes ses sous-listes. Passez 0 pour concentrer la recherche sur le premier niveau de la liste et 1 pour l'étendre à toutes les sous-listes.

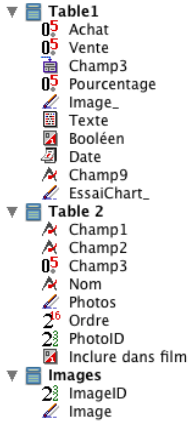
Si vous souhaitez connaître la position ou le numéro de tous les éléments correspondant à *valeur*, passez un tableau d'entiers longs dans

le paramètre facultatif *tabEléments*. Si nécessaire, le tableau sera créé et redimensionné par la commande. La commande remplira le tableau avec les positions (si le second * est omis) ou les numéros de référence (si le second * est passé) des éléments trouvés.

Les positions sont exprimées relativement à l'élément supérieur de la liste principale, en tenant compte de l'état courant déployé/contracté de la liste et de ses sous-listes.

Si aucun élément ne correspond à la *valeur* recherchée, la fonction retourne 0 et le tableau *tabEléments* est retourné vide.

▼ Soit la liste hiérarchique suivante :




```
$vlItemPos:=Chercher dans liste(hList;"P@";1; $arrPos)
   `$vlItemPos vaut 5
   `$arrPos{1} vaut 5, $arrPos{2} vaut 17 et $arrPos{3} vaut 19

$vlItemRef:=Chercher dans liste(hList;"P@";1;$arrRefs;*)
   `$vlItemRef vaut 7
   `$arrRefs{1} vaut 7, $arrRefs{2} vaut 18 et $arrRefs{3} vaut 23

$vlItemPos:=Chercher dans liste(hList;"Date";1;$arrPos)
   `$vlItemPos vaut 9
   `$arrPos{1} vaut 9

$vlItemRef:Chercher dans liste(hList;"Date";1;$arrRefs;*)
   `$vlItemRef vaut 11
   `$arrRefs{1} vaut 11

$vlItemPos:=Chercher dans liste(hList;"Date";0;*)
   `$vlItemPos vaut 0
```

FIXER ICONE ELEMENT

FIXER ICONE ELEMENT({*; }liste; réfElément | *; icône)

Paramètres	Type		Description
*	*	→	Si spécifié, liste est un nom d'objet (chaîne) Si omis, liste est un numéro de référence de liste
liste	RefListe | Chaîne	→	Nom d'objet de type liste (si * spécifié) Numéro de référence de liste (si * omis)
réfElément | *	Entier long | *	→	Numéro de référence d'élément ou 0 pour le dernier élément ajouté à la liste ou * pour l'élément courant de la liste
icône	Image	→	Icône à associer à l'élément

La commande FIXER ICONE ELEMENT permet de modifier l'icône associée à l'élément désigné par le paramètre *réfElément* de la liste dont vous avez passé le numéro de référence ou le nom d'objet dans *liste*.

Note Il est déjà possible de modifier l'icône associée à un élément à l'aide de la commande CHANGER PROPRIETES ELEMENT. Toutefois, à la différence de cette nouvelle commande, CHANGER PROPRIETES ELEMENT accepte uniquement des références d'images statiques (références de ressources ou images de la bibliothèque).

Si vous passez le premier paramètre optionnel *, vous indiquez que le paramètre *liste* est un nom d'objet (chaîne) correspondant à une représentation de liste dans le formulaire. Si vous ne passez pas ce paramètre, vous indiquez que le paramètre *liste* est une référence de liste hiérarchique (*RefListe*).
Si vous utilisez une seule représentation de liste ou travaillez avec les éléments structurels (le second * est omis), vous pouvez utiliser indifféremment l'une ou l'autre syntaxe.
En revanche, si vous utilisez plusieurs représentations d'une même liste et travaillez avec l'élément courant (le second * est passé), la syntaxe basée sur le nom d'objet est requise car chaque représentation peut disposer de son propre élément courant.

Vous pouvez passer un numéro de référence dans *réfElément*. Si ce numéro ne correspond à aucun élément de la liste, la commande ne fait rien. Vous pouvez également passer 0 pour désigner le dernier élément ajouté à la liste (à l'aide de AJOUTER A LISTE).
Vous pouvez enfin passer * dans *réfElément* pour désigner l'élément

courant de la liste. Si plusieurs éléments sont sélectionnés manuellement, l'élément courant est celui qui a été sélectionné en dernier. Si aucun élément n'est sélectionné, la commande ne fait rien.

Passez dans le paramètre *icône* une expression image 4D valide (champ, variable, pointeur, etc.). L'image sera placée à gauche de l'élément.

L'emploi de pointeurs est particulièrement recommandé car les listes hiérarchiques sont optimisées dans ce cas : une seule instance de l'image sera créée en mémoire si la même icône est utilisée pour plusieurs éléments de la liste.

Note A l'inverse, l'emploi direct de variables générées par les commandes LIRE RESSOURCE ICONE ou LIRE RESSOURCE IMAGE est déconseillé car l'icône sera dupliquée en mémoire pour chaque élément de la liste.

▼ Ce code est optimisé grâce à l'emploi d'un pointeur :

```
vIcon:=->[Params]Icone
FIXER ICONE ELEMENT(maliste;ref1;vIcon->)
FIXER ICONE ELEMENT(maliste;ref2;vIcon->)
```

Référence : LIRE ICONE ELEMENT

LIRE ICONE ELEMENT

LIRE ICONE ELEMENT({*; }liste; réfElément | *; icône)

Paramètres	Type		Description
*	*	→	Si spécifié, liste est un nom d'objet (chaîne) Si omis, liste est un numéro de référence de liste
liste	RefListe | Chaîne	→	Nom d'objet de type liste (si * spécifié) Numéro de référence de liste (si * omis)
réfElément | *	Entier long | *	→	Numéro de référence d'élément ou 0 pour le dernier élément ajouté à la liste ou * pour l'élément courant de la liste
icône	Var image	←	Icône associée à l'élément

La commande LIRE ICONE ELEMENT retourne dans *icône* l'icône associée à l'élément dont vous avez passé le numéro de référence dans *réfElément* dans la liste dont vous avez passé le numéro de référence ou le nom d'objet dans *liste*.

Si vous passez le premier paramètre optionnel *****, vous indiquez que le paramètre *liste* est un nom d'objet (chaîne) correspondant à une représentation de liste dans le formulaire. Si vous ne passez pas ce paramètre, vous indiquez que le paramètre *liste* est une référence de liste hiérarchique (*RefListe*).
Si vous utilisez une seule représentation de liste ou travaillez avec les éléments structurels (le second ***** est omis), vous pouvez utiliser indifféremment l'une ou l'autre syntaxe.
En revanche, si vous utilisez plusieurs représentations d'une même liste et travaillez avec l'élément courant (le second ***** est passé), la syntaxe basée sur le nom d'objet est requise car chaque représentation peut disposer de son propre élément courant.

Vous pouvez passer un numéro de référence dans *réfElément*. Si ce numéro ne correspond à aucun élément de la liste, la commande ne fait rien. Vous pouvez également passer 0 pour désigner le dernier élément ajouté à la liste (à l'aide de AJOUTER A LISTE). Vous pouvez enfin passer ***** dans *réfElément* pour désigner l'élément courant de la liste. Si plusieurs éléments sont sélectionnés manuellement, l'élément courant est celui qui a été sélectionné en dernier. Si aucun élément n'est sélectionné, la commande ne fait rien.

Passez dans *icône* une variable image. A l'issue de l'exécution de la commande, elle contiendra l'icône associée à l'élément, quelle que soit la source de l'icône (image statique, ressource ou expression image).

Si aucune icône n'est associée à l'élément, la variable *icône* est retournée vide.

Référence : FIXER ICONE ELEMENT

FIXER PARAMETRE ELEMENT

FIXER PARAMETRE ELEMENT({*; }liste; réfElément | *; sélecteur; valeur)

Paramètres	Type		Description
*	*	→	Si spécifié, liste est un nom d'objet (chaîne) Si omis, liste est un numéro de référence de liste
liste	RefListe	Chaîne	→	Nom d'objet de type liste (si * spécifié) Numéro de référence de liste (si * omis)
réfElément	*	Entier long	*	→	Numéro de référence d'élément ou 0 pour le dernier élément ajouté à la liste ou * pour l'élément courant de la liste
sélecteur	Chaîne	→	Constante de paramètre
valeur	Chaîne	Booléen	Numérique	→	Valeur du paramètre

La commande FIXER PARAMETRE ELEMENT permet de modifier le paramètre *sélecteur* pour l'élément *réfElément* de la liste hiérarchique dont vous avez passé la référence ou le nom d'objet dans le paramètre *liste*.

Si vous passez le premier paramètre optionnel *****, vous indiquez que le paramètre *liste* est un nom d'objet (chaîne) correspondant à une représentation de liste dans le formulaire. Si vous ne passez pas ce paramètre, vous indiquez que le paramètre *liste* est une référence de liste hiérarchique (*RefListe*).
Si vous utilisez une seule représentation de liste ou travaillez avec les références, vous pouvez utiliser indifféremment l'une ou l'autre syntaxe.
En revanche, si vous utilisez plusieurs représentations d'une même liste et passez le deuxième *****, la syntaxe basée sur le nom d'objet est requise car chaque représentation peut disposer de son propre élément courant.

Vous pouvez passer dans *sélecteur* la constante Texte supplémentaire (placée dans le thème "Listes hiérarchiques") ou toute valeur personnalisée :

Constante	Type	Valeur
Texte supplémentaire	Alpha	4D_additional_text

- Texte supplémentaire : cette constante permet d'ajouter un texte à droite de l'élément *réfElément*. Ce libellé supplémentaire reste toujours affiché dans la partie droite de la liste, même si l'utilisateur déplace le curseur de défilement horizontal.
  Lorsque vous utilisez cette constante, passez dans *valeur* le texte à afficher.

- Sélecteur personnalisé : vous pouvez passer dans *sélecteur* tout texte personnalisé et lui associer une *valeur* de type texte, numérique ou booléen. Cette *valeur* sera stockée avec l'élément et pourra être récupérée via la commande LIRE PARAMETRE ELEMENT.
  Ce principe permet de mettre en place tout type d'interface associée aux listes hiérarchiques. Par exemple, dans une liste stockant des noms de personnes, vous pouvez stocker l'âge de chaque personne et ne l'afficher que lorsque l'élément correspondant est sélectionné.

Référence : LIRE PARAMETRE ELEMENT

LIRE PARAMETRE ELEMENT

LIRE PARAMETRE ELEMENT({*; }liste; réfElément | *; sélecteur; valeur)

Paramètres	Type		Description
*	*	→	Si spécifié, liste est un nom d'objet (chaîne) Si omis, liste est un numéro de référence de liste
liste	RefListe | Chaîne	→	Nom d'objet de type liste (si * spécifié) Numéro de référence de liste (si * omis)
réfElément | *	Entier long | *	→	Numéro de référence d'élément ou 0 pour le dernier élément ajouté à la liste ou * pour l'élément courant de la liste
sélecteur	Chaîne	→	Constante de paramètre
valeur	Chaîne | Booléen | Numérique	←	Valeur courante du paramètre

La commande LIRE PARAMETRE ELEMENT permet de connaître la *valeur* courante du paramètre *sélecteur* pour l'élément *réfElément* de la liste hiérarchique dont vous avez passé la référence ou le nom d'objet dans le paramètre *liste*.

Si vous passez le premier paramètre optionnel *, vous indiquez que le paramètre *liste* est un nom d'objet (chaîne) correspondant à une représentation de liste dans le formulaire. Si vous ne passez pas ce paramètre, vous indiquez que le paramètre *liste* est une référence de liste hiérarchique (*RefListe*).
Si vous utilisez une seule représentation de liste ou travaillez avec les références, vous pouvez utiliser indifféremment l'une ou l'autre syntaxe.
En revanche, si vous utilisez plusieurs représentations d'une même liste et passez le deuxième paramètre *, la syntaxe basée sur le nom d'objet est requise car chaque représentation peut disposer de son propre élément courant.

Vous pouvez passer dans *sélecteur* la constante Texte supplémentaire (placée dans le thème "Listes hiérarchiques") ou toute valeur personnalisée.

Pour plus d'informations sur les paramètres *sélecteur* et *valeur*, reportez-vous à la description de la commande FIXER PARAMETRE ELEMENT, page 334.

Référence : FIXER PARAMETRE ELEMENT

LISTE ENUMERATIONS

LISTE ENUMERATIONS(tabNums;tabNoms)

Paramètres	Type		Description
tabNums	Tab Entier long	←	Numéros des énumérations
tabNoms	Tab Texte	←	Noms des énumérations

La commande LISTE ENUMERATIONS retourne dans les tableaux synchronisés *tabNums* et *tabNoms* les numéros et les noms des énumérations définies dans l'éditeur d'énumérations en mode Développement.

Les numéros des énumérations correspondent à leur ordre de création. Dans l'éditeur d'énumérations, les énumérations sont affichées par ordre alphabétique.

List box

Dans 4D v11, il est possible d'associer des champs et des expressions calculées aux colonnes de list box. Cette nouveauté est décrite dans le paragraphe "Associer des champs ou des expressions aux list box", page 148.

En conséquence, plusieurs commandes du thème "List Box" ont été modifiées et deux nouvelles commandes ont été ajoutées.

A noter que d'autres commandes en relation avec les list box ont été ajoutées ou modifiées dans 4D v11 :

- la nouvelle commande Choisir du thème "Outils", très utile pour l'insertion d'expressions,
- la commande Numero de ligne affichee fonctionne désormais avec les list box,
- la fonction Objet focus prend en charge les colonnes calculées.

Commandes modifiées

INSERER COLONNE LISTBOX

INSERER COLONNE LISTBOX({*; }objet; positionCol; nomCol; variableCol; nomEntête; variableEntête)

Paramètres	Type		Description
*		→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est une variable
objet	Objet de formulaire	→	Nom d'objet (si * est spécifié) ou Variable (si * est omis)
positionCol	Numérique	→	Emplacement de la colonne à insérer
nomCol	Alpha	→	Nom d'objet de la colonne
variableCol	Tableau | *Champ* | *Variable*	→	Nom de la variable tableau de la colonne *ou champ ou variable*
nomEntête	Alpha	→	Nom d'objet de l'en-tête de la colonne
variableEntête	Var Entier	→	Variable d'en-tête de la colonne

Il est désormais possible de passer un champ ou une variable dans le paramètre *variableCol*. Dans ce cas, le contenu de la colonne sera la valeur du champ ou de la variable, évaluée pour chaque

enregistrement de la sélection associée à la list box. Ce type de contenu ne peut être utilisé que lorsque la propriété “Source de données” de la list box est **Sélection courante** ou **Sélection temporaire**.

Note Il n’est pas possible de combiner dans une même list box des colonnes de type tableau (source de données tableaux) et des colonnes de type champ ou variable (source de données sélection).

Vous pouvez utiliser des champs ou des variables de type Alpha, Texte, numérique, Date, Heure, Image et Booléen.

Dans le contexte de list box basées sur des sélections, INSERER COLONNE LISTBOX permet d’insérer des éléments simples (champs ou variables). Si vous souhaitez manipuler des expressions plus complexes (telles que des formules ou des méthodes), vous devez utiliser la nouvelle commande INSERER COLONNE FORMULE LISTBOX, page 340.

▼ Nous souhaitons ajouter une colonne à la droite de la list box et lui associer les valeurs du champ [Envois]Frais :

```
$der:=Lire nombre colonnes listbox(*;"ListBox1")+1
INSERER COLONNE LISTBOX(*;"ListBox1";$der;"ColChamp";
                                    [Envois]Frais;"NomEntete";VarEntete)
```

INSERER LIGNE LISTBOX

INSERER LIGNE LISTBOX({*; }objet; position)

Cette commande fonctionne uniquement avec les list box basées sur des tableaux. Lorsque cette commande est utilisée avec une list box basée sur des sélections, elle ne fait rien et la variable système OK retourne 0.

SUPPRIMER LIGNE LISTBOX

SUPPRIMER LIGNE LISTBOX({*; }objet; position)

Cette commande fonctionne uniquement avec les list box basées sur des tableaux. Lorsque cette commande est utilisée avec une list box basée sur des sélections, elle ne fait rien et la variable système OK retourne 0.

LIRE TABLEAUX LISTBOX

LIRE TABLEAUX LISTBOX({*; }objet; tabNomsCols; tabNomsEntêtes; *tabVarCols*; tabVariablesEntêtes; tabVisibles; *tabStyles*)

Paramètres	Type		Description
*		→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est une variable
objet	Objet de formulaire	→	Nom d'objet (si * est spécifié) ou Variable (si * est omis)
tabNomsCols	Tab alpha	←	Noms d'objet des colonnes
tabNomsEntêtes	Tab alpha	←	Noms d'objet des en-têtes
tabVarCols	Tab pointeur	←	Pointeurs vers les variables des colonnes *ou Pointeurs vers les champs des colonnes ou NIL*
tabVariablesEntêtes	Tab pointeur	←	Pointeurs vers les variables des entêtes
tabVisibles	Tab booléen	←	Visibilité de chaque colonne
tabStyles	Tab pointeur	←	Pointeurs vers les tableaux *ou les variables* de styles et de couleurs *ou Nil*

Lorsque la source des données de la list box est "Sélections", le contenu du tableau *tabVarCols* dépend du type de source associée à chaque colonne :

- pour une colonne associée à un champ, *tabVarCols* contient un pointeur vers le champ associé,
- pour une colonne associée à une variable, *tabVarCols* contient un pointeur vers la variable,
- pour une colonne associée à une expression, *tabVarCols* contient un pointeur Nil.

Dans le cas des list box basées sur une sélection, le paramètre *tabStyles* retourne trois pointeurs dont la nature dépend du type de source utilisée pour le paramétrage du style :

- pour chaque paramétrage défini via une variable, *tabStyles* contient un pointeur vers la variable,
- pour chaque paramétrage défini via une expression, *tabStyles* contient un pointeur Nil.

Nouvelles commandes

INSERER COLONNE FORMULE LISTBOX

INSERER COLONNE FORMULE LISTBOX({*; }objet; positionCol; nomCol; formule; typeDonnées; nomEntête; variableEntête)

Paramètres	Type		Description
*		→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est une variable
objet	Objet de formulaire	→	Nom d'objet (si * est spécifié) ou Variable (si * est omis)
positionCol	Numérique	→	Emplacement de la colonne à insérer
nomCol	Chaîne	→	Nom d'objet de la colonne
formule	Chaîne	→	Formule 4D associée à la colonne
typeDonnées	Entier long	→	Type du résultat de la formule
nomEntête	Chaîne	→	Nom d'objet de l'en-tête de la colonne
variableEntête	Var Entier	→	Variable d'en-tête de la colonne

La nouvelle commande INSERER COLONNE FORMULE LISTBOX est semblable à la commande INSERER COLONNE LISTBOX, à la différence près qu'elle permet la saisie d'une formule comme contenu de la colonne.
Ce type de contenu ne peut être utilisé que lorsque la propriété "Source de données" de la list box est **Sélection courante** ou **Sélection temporaire**.

Le paramètre *formule* peut contenir toute expression valide, soit :

- une instruction,
- une formule générée à l'aide de l'éditeur de formules,
- un appel à une commande 4D,
- un appel à une méthode projet.

Au moment de l'appel de la commande, la *formule* est analysée puis exécutée.

Note Utilisez la commande Nom commande afin de définir des formules indépendantes de la langue de l'application (lorsqu'elles font appel à des commandes 4D).

Le paramètre *typeDonnées* permet de désigner le type des données issues de l'exécution de la *formule*. Vous devez passer dans ce paramètre une des constantes du thème "Types champs et variables" suivantes :

Constante	Type	Valeur
Est un numérique	Entier long	1
Est un texte	Entier long	2
Est une image	Entier long	3
Est une date	Entier long	4
Est un booléen	Entier long	6
Est une heure	Entier long	11

Si le résultat de la *formule* ne correspond pas au type de données attendu, une erreur est générée.

Les autres paramètres sont identiques à ceux de la commande INSERER COLONNE LISTBOX.

▼ Nous souhaitons ajouter une nouvelle colonne à la droite de la list box qui contiendra une formule calculant l'âge de l'employé :

```
vAge:="Date du jour-[Employés]DateNaissance)\365"
$der:=Lire nombre colonnes listbox(*;"ListBox1")+1
INSERER COLONNE FORMULE LISTBOX(*;"ListBox1";$der;
                    "ColFormule";Est un numérique;vAge;"Age";VarEntete)
```

Référence : INSERER COLONNE LISTBOX

FIXER TABLE SOURCE LISTBOX

FIXER TABLE SOURCE LISTBOX({*; }objet; numTable | tempo)

Paramètres	Type		Description
*		→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est une variable
objet	Objet de formulaire	→	Nom d'objet (si * est spécifié) ou Variable (si * est omis)
numTable | tempo	E. long | Chaîne	→	Numéro de la table de laquelle utiliser la sélection courante ou Nom de la sélection temporaire à utiliser

La nouvelle commande FIXER TABLE SOURCE LISTBOX vous permet de modifier la source des données affichées dans la listbox désignée par les paramètres * et *objet*.

Si vous passez un numéro de table comme paramètre *numTable*, la listbox sera remplie avec les données des enregistrements de la sélection courante de la table.

Si vous passez un nom de sélection temporaire comme paramètre *tempo*, la listbox sera remplie avec les données des enregistrements appartenant à la sélection temporaire.

Cette commande ne fait rien si vous l'utilisez avec une listbox associée à un tableau.

Si la listbox contenait déjà des colonnes, leur contenu est mis à jour à l'issue de l'exécution de la commande.

Référence : LIRE TABLE SOURCE LISTBOX

LIRE TABLE SOURCE LISTBOX

LIRE TABLE SOURCE LISTBOX({*; }objet; numTable{; tempo})

Paramètres	Type		Description
*		→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est une variable
objet	Objet de formulaire	→	Nom d'objet (si * est spécifié) ou Variable (si * est omis)
numTable	Entier long	←	Numéro de la table de la sélection
tempo	Chaîne	←	Nom de la sélection temporaire ou "" pour la sélection courante

La nouvelle commande LIRE TABLE SOURCE LISTBOX vous permet de connaître la source courante des données affichées dans la listbox désignée par les paramètres * et *objet*.

La commande retourne dans le paramètre *numTable* le numéro de la table principale associée à la list box et dans le paramètre facultatif *tempo* le nom de la sélection temporaire éventuellement utilisée.
Si les lignes de la list box sont liées à la sélection courante de la table, le paramètre *tempo*, s'il est passé, retourne une chaîne vide.
Si les lignes de la list box sont liées à une sélection temporaire, le paramètre *tempo* retourne le nom de cette sélection temporaire.

Si la list box est associée à des tableaux, *numTable* retourne -1 et *tempo*, s'il est passé, retourne une chaîne vide.

Référence : FIXER TABLE SOURCE LISTBOX

Fenêtres

La commande Creer fenetre accepte une nouvelle constante permettant de générer des fenêtres d'aspect métal sous Mac OS.

Creer fenetre

Dans 4D v11 , il est possible d'appliquer l'apparence "métal" aux fenêtres Mac OS (cf. paragraphe "Aspect "métal"", page 168).

Pour cela, une nouvelle constante de type de fenêtre est disponible dans le thème de constantes "Creer fenetre" :
Aspect métal (valeur = 2048).

Cette constante permet d'appliquer l'apparence "métal" à une fenêtre générée par Creer fenetre. Elle doit être ajoutée au type de fenêtre défini dans le paramètre *type*. Par exemple :

```
$fen:=Creer fenetre(10;80;-1;-1;Fenêtre standard+Aspect métal;"")
```

Note Sous Windows, cette constante est sans effet.

Cette apparence peut être associée à la plupart des types de fenêtres sous Mac OS.

Thème : Fenêtres.

Impressions

Le thème "Impressions" comporte les nouvelles commandes OUVRIR TACHE IMPRESSION et FERMER TACHE IMPRESSION, vous permettant de contrôler les tâches d'impression. Plusieurs commandes existantes ont également été modifiées.

Nouvelles commandes

OUVRIR TACHE IMPRESSION

OUVRIR TACHE IMPRESSION

Paramètres	Type	Description
Cette commande ne requiert pas de paramètre		

La commande OUVRIR TACHE IMPRESSION ouvre une tâche d'impression (*print job)* et y empile tous les ordres d'impression exécutés par la suite, tant que la commande FERMER TACHE

IMPRESSION n'est pas appelée. Cette commande vous permet de contrôler les tâches d'impression, et notamment de vous assurer qu'aucune tâche d'impression "parasite" ne puisse s'intercaler dans une séquence d'impressions.

La commande OUVRIR TACHE IMPRESSION peut être utilisée avec toutes les commandes d'impression de 4D, les commandes de l'éditeur d'états rapides ainsi que les commandes d'impression des plug-ins 4D Write et 4D View.

En revanche, cette commande n'est pas compatible avec les plug-ins 4D Chart et 4D Draw, ainsi que la plupart des plug-ins tiers. Lorsqu'une tâche est ouverte avec cette commande, l'imprimante est placée en mode "occupé" jusqu'à ce que l'impression soit effectivement lancée. Si un plug-in non compatible lance une impression dans ce contexte, l'erreur "imprimante occupée" est retournée.

Vous devez appeler la commande FERMER TACHE IMPRESSION pour terminer la tâche et envoyer le document d'impression à l'imprimante. Si vous omettez cette commande, le document d'impression restera dans la pile et l'imprimante sera inaccessible jusqu'à ce que vous quittiez l'application 4D.

La tâche d'impression est locale au process. Il est possible d'ouvrir autant de tâches d'impressions que de process. Bien entendu, dans ce cas il est nécessaire de disposer de plusieurs imprimantes car chaque imprimante est occupée jusqu'à la la fin de la tâche.

OUVRIR TACHE IMPRESSION utilise les paramètres d'impression courants (paramètres par défaut ou définis via les commandes UTILISER PARAMETRES IMPRESSION et/ou FIXER OPTION IMPRESSION). Les commandes modifiant les paramètres d'impression doivent être appelées avant OUVRIR TACHE IMPRESSION. Dans le cas contraire, une erreur est générée.

Référence : FERMER TACHE IMPRESSION

FERMER TACHE IMPRESSION

FERMER TACHE IMPRESSION

Paramètres	Type	Description
Cette commande ne requiert pas de paramètre		

La commande FERMER TACHE IMPRESSION permet de refermer la tâche d'impression préalablement ouverte par la commande OUVRIR TACHE IMPRESSION et d'envoyer à l'imprimante courante le document d'impression éventuellement construit.

Une fois cette commande exécutée, l'imprimante redevient disponible pour d'autres impressions.

Référence : OUVRIR TACHE IMPRESSION

Commandes modifiées

PARAMETRES IMPRESSION

PARAMETRES IMPRESSION*{(typeDial)}*

Paramètres	Type		Description
typeDial	Entier long	→	Boîte(s) de dialogue à afficher : 0 (ou paramètre omis) = toutes 1 = Format d'impression 2 = Impression

La commande PARAMETRES IMPRESSION admet désormais le paramètre facultatif *typeDial* permettant de définir l'affichage des boîtes de dialogue d'impression.

- Si vous passez 0 dans *typeDial* ou omettez ce paramètre, les deux boîtes de dialogue d'impression sont affichées.
- Si vous passez 1 dans *typeDial*, seule la boîte de dialogue de paramétrage de l'impression est affichée. Les options d'impression courantes seront utilisées.
- Si vous passez 2 dans *typeDial*, seule la boîte de dialogue d'impression est affichée. Les paramètres d'impression courants seront utilisés.

Cette commande doit être appelée avant une série de commandes Imprimer ligne ou la commande OUVRIR TACHE IMPRESSION.

Référence : OUVRIR TACHE IMPRESSION

CUMULER SUR, NIVEAUX DE RUPTURE

Dans 4D v11, vous devez exécuter les commandes NIVEAUX DE RUPTURE et CUMULER SUR avant la génération de chaque état dans lequel vous voulez utiliser des ruptures, **quel que soit le mode d'exécution** (compilé ou interprété).

Dans les versions précédentes, ces commandes étaient obligatoires pour le mode compilé uniquement.

Environnement 4D

Ce thème s'est enrichi de plusieurs nouvelles commandes dans 4D v11, permettant d'effectuer par programmation des opérations de vérification, de réparation ou et de compactage des fichiers de données de 4D. Ces opérations sont également disponibles dans le Centre de Sécurité et de Maintenance.

Note Pour les utilisateurs des versions précédentes de 4D, ces fonctions étaient auparavant présentes dans l'utilitaire 4D Tools. Elles ont été intégrées dans 4D en version 11.

Ce thème comporte également les nouvelles commandes Lire langue courante base, permettant de connaître la langue utilisée par l'application, et LISTE COMPOSANTS, permettant d'obtenir la liste des composants installés.

Enfin, plusieurs commandes existantes ont été modifiées.

VERIFIER FICHIER DONNEES

VERIFIER FICHIER DONNEES(cheminStructure; cheminDonnées; objets; options; méthode{; tabTables; tabChamps})

Paramètres	Type		Description
cheminStructure	Texte	→	Chemin d'accès du fichier de structure de la base à vérifier
cheminDonnées	Texte	→	Chemin d'accès du fichier de données 4D à vérifier
objets	Numérique	→	Objets à vérifier
options	Numérique	→	Options de vérification
méthode	Texte	→	Nom de la méthode 4D de rétro-appel
tabTables	Tab num	→	Numéros des tables à vérifier
tabChamps	Tab num 2D	→	Numéros des index à vérifier

La commande VERIFIER FICHIER DONNEES effectue une vérification structurelle des *objets* contenus dans le fichier de données 4D désigné par *cheminStructure* et *cheminDonnées*.

Note Pour plus d'informations sur la vérification des données, reportez-vous au paragraphe "Page Vérification", page 193.

- *cheminStructure* désigne le fichier de structure (compilé ou non) associé au fichier de données à vérifier. Il peut s'agir du fichier de structure ouvert ou de tout autre fichier de structure. Vous devez passer un chemin d'accès complet, exprimé avec la syntaxe du système d'exploitation. Vous pouvez également passer une chaîne vide, dans ce cas une boîte de dialogue standard d'ouverture de fichiers apparaît, permettant à l'utilisateur de désigner le fichier de structure à utiliser.
- *cheminDonnées* désigne un fichier de données 4D (.4DD). Il doit correspondre au fichier de structure défini par le paramètre *cheminStructure*. Attention, vous pouvez désigner le fichier de structure courant mais le fichier de données ne doit pas être le fichier courant (ouvert). Pour vérifier le fichier de données ouvert, utilisez la commande VERIFIER FICHIER DONNEES OUVERT.
  Si vous tentez de vérifier le fichier de données courant avec la commande VERIFIER FICHIER DONNEES, une erreur est générée.

  Le fichier de données désigné est ouvert en lecture seulement. Vous devez veiller à ce qu'aucune application n'accède à ce fichier en écriture, sinon les résultats de la vérification pourront être faussés.

  Vous pouvez passer dans le paramètre *cheminDonnées* une chaîne vide, un nom de fichier ou un chemin d'accès complet, exprimé dans la syntaxe du système d'exploitation. Si vous passez une chaîne vide, la boîte de dialogue standard d'ouverture de fichier apparaît, permettant à l'utilisateur de désigner le fichier à vérifier (à noter dans ce cas qu'il n'est pas possible de sélectionner le fichier de données courant). Si vous passez uniquement un nom de fichier de données, 4D le recherchera au même niveau que le fichier de structure défini.

- Le paramètre *objets* permet de désigner les types d'objets à vérifier. Deux types d'objets peuvent être vérifiés : les enregistrements et les index. Utilisez les constantes suivantes, placés dans le thème "Maintenance fichier de données" :
  - Vérifier enregistrements (4)
  - Vérifier index (8)
  - Tout vérifier sans rétroappel (16)

  Pour vérifier les enregistrements et les index, passez le cumul Vérifier enregistrements+Vérifier index. La valeur 0 (zéro) permet également d'obtenir le même résultat.
  L'option Tout vérifier sans rétroappel est un cas particulier. Elle effectue la vérification interne la plus complète, mais, pour des raisons internes, ne permet pas le rétro-appel d'une méthode. Cette vérification est néanmoins compatible avec la création d'un historique.

- Le paramètre *options* permet de définir les options de vérification. Une seule option est actuellement disponible, accessible via une constante du thème "Maintenance fichier de données" :
  - Ne pas créer d'historique (16384)
    En principe, la commande VERIFIER FICHIER DONNEES crée un fichier d'historique au format xml (reportez-vous à la fin de la description de cette commande). Vous pouvez annuler ce fonctionnement en passant cette option.

  Pour créer l'historique, passez 0 dans *options*.

- Le paramètre *méthode* permet de définir une méthode de rétro-appel qui sera régulièrement appelée durant la vérification. Si vous passez une chaîne vide, aucune méthode n'est appelée. Si la méthode passée n'existe pas, la vérification n'est pas effectuée, une erreur est générée et la variable OK prend la valeur 0.
  Lorsqu'elle est appelée, cette méthode reçoit jusqu'à 5 paramètres en fonction du type d'événement à l'origine de l'appel (cf. tableau des appels). Vous devez impérativement déclarer ces paramètres dans la méthode :

$1	Entier long	Type de message (cf. tableau)
$2	Entier long	Type d'objet
$3	Texte	Message
$4	Entier long	Numéro de table
$5	Entier long	*Réservé*

Le tableau suivant décrit le contenu des paramètres en fonction du type d'événement :

Evénement	$1 (E. long)	$2 (E. long)	$3 (Texte)	$4 (E. long)	$5 (E. long)
Message	1	0	Message de progression	Pourcentage réalisé (0-100)	*Réservé*
Vérification OK	2	Type d'objet	Texte du message OK	Numéro de table ou d'index	*Réservé*
Erreur	3	Type d'objet	Texte du message d'erreur	Numéro de table ou d'index	*Réservé*
Fin d'exécution	4	0	DONE	0	*Réservé*
Warning	5	Type d'objet	Texte du message d'erreur	Numéro de table ou d'index	*Réservé*

Type d'objet : Lorsqu'un objet a été vérifié, un message OK ($1=2), erreur ($1=3) ou warning ($1=5) peut être envoyé. Le type d'objet retourné dans $2 peut être l'un des suivants :

- 0 = indéterminé
- 4 = enregistrement
- 8 = index
- 16 = objet structure (contrôle préliminaire du fichier de données).

Cas particulier : lorsque $4 = 0 pour $1 = 2, 3 ou 5, le message ne concerne pas une table ou un index mais le fichier de données dans son ensemble.

La méthode de rétro-appel doit également retourner une valeur dans $0 (Entier long), permettant de contrôler l'exécution de l'opération :

- si $0 = 0, l'opération continue normalement
- si $0 = -128, l'opération est stoppée sans erreur générée
- si $0 = autre valeur, l'opération est stoppée et la valeur passée dans $0 est retournée en tant que numéro d'erreur. Cette erreur peut être interceptée par une méthode d'appel sur erreur.

Si la méthode de rétro-appel n'existe pas dans la base, une erreur est générée et la variable système OK prend la valeur 0.

Deux tableaux facultatifs peuvent également être utilisés par la commande :

- Le tableau *tabTables* contient les numéros des tables dont les enregistrements doivent être vérifiés. Il permet de limiter la vérification à certaines tables. Si ce paramètre n'est pas passé ou si le tableau est vide et que le paramètre *objets* contient Vérifier enregistrements, toutes les tables sont vérifiées.
- Le tableau *tabChamps* contient les numéros des champs indexés dont les index doivent être vérifiés.
  Si ce paramètre n'est pas passé ou si le tableau est vide et que le paramètre *objets* contient Vérifier index, tous les index sont vérifiés.
  La commande ignore les champs non indexés.
  Si un champ contient plusieurs index, tous les index sont vérifiés. Si un champ fait partie d'un index composite, la totalité de l'index est vérifiée.
  Vous devez passer un tableau 2D dans *tabChamps*. Pour chaque ligne du tableau :
  - l'élément {0} contient le numéro de la table,
  - les autres éléments {1...n} contiennent les numéros des champs.

  Par défaut, la commande VERIFIER FICHIER DONNEES crée un fichier d'historique au format xml (si vous n'avez pas passé l'option Ne pas créer d'historique, cf. paramètre *options*). Son nom est basé sur celui du fichier de données et il est placé à côté de ce fichier. Par exemple, pour un fichier de données nommé "data.4dd", le fichier d'historique sera nommé "data_verify_log.xml".

▼ Vérification simple des données et des index :

VERIFIER FICHIER DONNEES($NomStruct;$NomData;Vérifier index+
Vérifier enregistrements; Ne pas créer d'historique;"")

▼ Vérification complète avec historique :

VERIFIER FICHIER DONNEES($NomStruct;$NomData;
Tout vérifier sans rétroappel;0;"")

▼ Vérification des enregistrements uniquement :

VERIFIER FICHIER DONNEES($NomStruct;$NomData;
Vérifier enregistrements;0;"")

- Vérification des enregistrements des tables 3 et 7 uniquement :

```
TABLEAU ENTIER LONG($tnumTables;2)
TABLEAU ENTIER LONG($tindex;0) `non utilisé mais obligatoire
$tnumTables{1}:=3
$tnumTables{2}:=7
VERIFIER FICHIER DONNEES($NomStruct;$NomData;
                Vérifier enregistrements;0;"FollowScan";$tnumTables;$tindex)
```

- Vérification d'index spécifiques (index du champ 1 de la table 4 et index des champs 2 et 3 de la table 5) :

```
TABLEAU ENTIER LONG($tnumTables;0) `non utilisé mais obligatoire
TABLEAU ENTIER LONG($tindex;2;0) `2 lignes (colonnes ajoutées ensuite)

$tindex{1}{0}:=4 ` numéro de table dans l'élément 0
AJOUTER A TABLEAU($tindex{1};1) ` numéro du 1er champ à vérifier

$tindex{2}{0}:=5 ` numéro de table dans l'élément 0
AJOUTER A TABLEAU($tindex{2};2) ` numéro du 1er champ à vérifier
AJOUTER A TABLEAU($tindex{2};3) ` numéro du 2e champ à vérifier

VERIFIER FICHIER DONNEES($NomStruct;$NomData;Vérifier index;0;
                "FollowScan";$tnumTables;$tindex)
```

Référence : Compacter fichier donnees, VERIFIER FICHIER DONNEES OUVERT

VERIFIER FICHIER DONNEES OUVERT

VERIFIER FICHIER DONNEES OUVERT {(objets; options; méthode; {tabTables; tabChamps})}

Paramètres	Type		Description
objets	Numérique	→	Objets à vérifier
options	Numérique	→	Options de vérification
méthode	Texte	→	Nom de la méthode 4D de rétro-appel
tabTables	Tab num	→	Numéros des tables à vérifier
tabChamps	Tab num 2D	→	Numéros des index à vérifier

La commande VERIFIER FICHIER DONNEES OUVERT effectue une vérification structurelle des *objets* contenus dans le fichier de données actuellement ouvert par 4D.

Cette commande a un fonctionnement identique à celui de la commande VERIFIER FICHIER DONNEES, à la différence près qu'elle

s'applique uniquement au fichier de données courant de la base de données ouverte. Elle ne nécessite donc pas de paramètres désignant la structure et les données.
Reportez-vous à la commande VERIFIER FICHIER DONNEES pour une description des paramètres.

Si vous passez la commande VERIFIER FICHIER DONNEES OUVERT sans aucun paramètre, la vérification est effectuée avec les valeurs par défaut des paramètres :

- *objets* = Vérifier enregistrements+Vérifier index (= valeur 0)
- *options* = 0 (l'historique est créé)
- *méthode* = ""
- *tabTables* et *tabChamps* sont omis.

Lorsque cette commande est exécutée, le cache de données est écrit sur le disque et toutes les opérations accédant aux données sont bloquées durant la vérification.

Note Cette commande ne doit pas être utilisée lorsqu'un process est en cours de création ou de mise à jour d'index car dans ce cas, les résultats de la vérification seront invalides.

Référence : VERIFIER FICHIER DONNEES

Compacter fichier donnees

Compacter fichier donnees (cheminStructure; cheminDonnées; {dossierArchive{; options{; méthode}}}) → Texte

Paramètres	Type		Description
cheminStructure	Texte	→	Chemin d'accès du fichier de structure
cheminDonnées	Texte	→	Chemin d'accès du fichier de données à compacter
dossierArchive	Texte	→	Chemin d'accès du dossier dans lequel placer le fichier de données original
options	Num	→	Options de compactage
méthode	Texte	→	Nom de la méthode 4D de rétro-appel
Résultat	Texte	←	Chemin d'accès complet du dossier contenant le fichier de données original

La commande Compacter fichier donnees effectue un compactage du fichier de données désigné par le paramètre *cheminDonnées* associé au fichier de structure *cheminStructure*. Pour plus d'informations sur le

compactage, reportez-vous au paragraphe “Pourquoi compacter ?”, page 197.
Pour assurer la continuité du fonctionnement de la base, le nouveau fichier de données compacté remplace automatiquement le fichier original. Par sécurité, le fichier original n’est pas modifié et est déplacé dans un dossier spécial nommé “Replaced files (compacting) *AAAA-MM-JJ HH-MM-SS*” où *AAAA-MM-JJ HH-MM-SS* représente la date et l’heure de la sauvegarde. Par exemple : “Replaced files (compacting) 2007-09-27 15-20-35”.

La commande retourne le chemin d’accès complet du dossier effectivement créé pour stocker le fichier de données original.

Cette commande peut être exécutée depuis 4D monoposte ou 4D Server uniquement (procédure stockée).

Le fichier de données à compacter doit correspondre au fichier de structure désigné par *cheminStructure*. En outre, il ne doit PAS être ouvert au moment de l’exécution de la commande, sinon une erreur est générée.
Si une erreur se produit durant le processus de compactage, les fichiers originaux sont conservés à leur emplacement initial.

Si un fichier d’index (fichier *.4DIndx*) est associé au fichier de données, il est également compacté. Comme pour le fichier de données, le fichier original est sauvegardé et la nouvelle version compactée remplace la précédente.

- Passez dans *cheminStructure* le chemin d’accès complet du fichier de structure associé au fichier de données que vous souhaitez compacter Cette information est nécessaire à la procédure de compactage. Le chemin d’accès doit être exprimé dans la syntaxe du système d’exploitation.
Vous pouvez également passer une chaîne vide, dans ce cas une boîte de dialogue standard d’ouverture de fichiers apparaît, permettant à l’utilisateur de désigner le fichier de structure à utiliser.
- Vous pouvez passer dans *cheminDonnées* une chaîne vide, un nom de fichier ou un chemin d’accès complet, exprimé dans la syntaxe du système d’exploitation. Si vous passez une chaîne vide, la boîte de dialogue standard d’ouverture de fichier apparaît, permettant à l’utilisateur de désigner le fichier de données à compacter. Ce fichier doit correspondre au fichier de structure défini dans le paramètre

cheminStructure. Si vous passez uniquement un nom de fichier de données, 4D le recherchera au même niveau que le fichier de structure.

- Le paramètre facultatif *dossierArchive* permet de désigner l'emplacement du dossier "Replaced files (compacting) *Dateheure*" destiné à recueillir les versions originales des fichiers de données ainsi que des éventuels fichiers d'index.
  La commande retourne le chemin d'accès complet du dossier effectivement créé.
  - Si vous omettez ce paramètre, les fichiers d'origine sont automatiquement déplacés dans un dossier "Replaced files (compacting) *Dateheure*" créé à côté du fichier de données.
  - Si vous passez une chaîne vide, une boîte de dialogue standard d'ouverture de dossier apparaît, permettant à l'utilisateur de désigner l'emplacement du dossier à créer.
  - Si vous passez un chemin d'accès (exprimé dans la syntaxe du système d'exploitation), la commande créera le dossier "Replaced files (compacting) *Dateheure*" à cet emplacement.
- Le paramètre facultatif *options* permet de définir diverses options liées au compactage. Pour cela, utilisez les constantes suivantes, placées dans le thème "Maintenance fichier de données". Vous pouvez passer les deux options en les cumulant :
  - Ne pas créer d'historique (16384)
    En principe, la commande Compacter fichier donnees crée un fichier d'historique au format xml (reportez-vous à la fin de la description de cette commande). Vous pouvez annuler ce fonctionnement en passant cette option.
  - Créer process (32768)
    Lorsque cette option est passée :
    - le compactage sera asynchrone et vous devrez gérer les résultats à l'aide de la méthode de rétro-appel (voir ci-dessous),
    - 4D n'affichera pas de barre de progression (il est possible de le faire via la méthode de rétro-appel),
    - la variable système OK prendra la valeur 1 si le process a été correctement lancé et 0 dans les autres cas.
    Lorsque cette option n'est pas passée, la variable OK prendra la valeur 1 si le compactage s'est correctement déroulé et sinon 0.
- Le paramètre *méthode* permet de désigner une méthode de rétro-appel qui sera régulièrement appelée durant le compactage si l'option Créer process a été passée. Dans le cas contraire, la méthode de rétro-appel

n'est jamais appelée. Pour plus d'informations sur cette méthode, reportez-vous à la description de la commande VERIFIER FICHIER DONNEES, page 346.
Si la méthode de rétro-appel n'existe pas dans la base, une erreur est générée et la variable système OK prend la valeur 0.

- Si l'opération de compactage s'est déroulée correctement, la variable système OK prend la valeur 1, sinon elle prend la valeur 0.

Par défaut, la commande Compacter fichier donnees crée un fichier d'historique au format xml (si vous n'avez pas passé l'option Ne pas créer d'historique, cf. paramètre *options*). Son nom est basé sur celui du fichier de données et il est placé à côté de ce fichier. Par exemple, pour un fichier de données nommé "data.4dd", le fichier d'historique sera nommé "data_compact_log.xml".

- L'exemple suivant (Windows) effectue le compactage d'un fichier de données. Un fichier d'historique sera créé.

```
$ficStruc:=Fichier structure
$ficDonnées:="C:\Bases\Factures\Janvier\Factures.4dd"
$ficOrig:="C:\Bases\Factures\Archives\Janvier\"
$dossierArch:=Compacter fichier donnees($ficStruc;$ficDonnées;
                                                    $ficOrig)
```

Référence : VERIFIER FICHIER DONNEES

OUVRIR CENTRE DE SECURITE

OUVRIR CENTRE DE SECURITE

Paramètres	Type	Description
Cette commande ne requiert pas de paramètre		

La commande OUVRIR CENTRE DE SECURITE provoque l'affichage de la fenêtre du Centre de sécurité et de maintenance (CSM).

En fonction des privilèges d'accès de l'utilisateur courant, certaines fonctions proposées dans la fenêtre pourront être désactivées.

Pour plus d'informations sur le CSM, reportez-vous au chapitre "Centre de Sécurité et de Maintenance", page 185.

Lire langue courante base

Lire langue courante base → Chaîne

Paramètres	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Chaîne	←	Langue courante de la base de données

La commande Lire langue courante base retourne la langue courante utilisée par la base de données, exprimée dans la norme définie par la RFC 3066. Typiquement, la commande retourne “fr” pour le français “es” pour l’espagnol, etc. Pour plus d’informations sur cette norme et sur les valeurs retournées par cette commande, reportez-vous au paragraphe “Nom de dossier .lproj”, page 105.

La langue courante de la base permet de définir le dossier *.lproj* dans lequel le programme va chercher les éléments localisés de la base de données. 4D détermine automatiquement la langue courante au démarrage de la base en fonction du contenu du dossier *Resources* et de l’environnement système. Le principe est que 4D charge le premier dossier *.lproj* de la base correspondant à la langue de référence, dans l’ordre de priorité suivant :

1. Langue du système (sous Mac OS, plusieurs langues peuvent être définies avec un ordre de préférence, 4D utilise ce paramétrage).
2. Langue de l’application 4D.
3. Anglais
4. Première langue trouvée dans le dossier *Resources*.

Note Si la base ne contient aucun dossier *.lproj*, 4D applique l’ordre de priorité suivant : 1. Langue du système, 2. Anglais (si la langue du système n’a pas pu être identifiée).

LISTE COMPOSANTS

LISTE COMPOSANTS(tabComposants)

Paramètres	Type		Description
tabComposants	Tab chaîne	←	Noms des composants

A l’ouverture d’une base, 4D charge les composants valides situés dans le dossier Components situé à côté du fichier de structure. La commande LISTE COMPOSANTS dimensionne et remplit le tableau *tabComposants* avec les noms des composants chargés par l’application 4D pour la base hôte courante.

Cette commande peut être appelée depuis la base hôte ou depuis un composant. Si la base n'utilise pas de composant, le tableau *tabComposants* est retourné vide.

Les noms des composants sont les noms des fichiers de structure des bases matrices (.4db, .4dc ou .4dbase).

Cette commande permet de mettre en place des architectures et des interfaces modulaires proposant des fonctionnalités supplémentaires en fonction de la présence des composants.

Note Pour plus d'informations sur les composants dans 4D v11, reportez-vous au paragraphe "Nouvelle architecture des composants", page 50.

Commandes modifiées

Les capacités de plusieurs commandes du thème "Environnement 4D" ont été étendues dans 4D v11.

Fichier structure

Fichier structure *{(*)}* → Alpha

Paramètres	Type		Description
*		→	*Retourner le fichier de structure de la base hôte*
Résultat	Alpha	←	Nom long du fichier de structure de la base

La commande Fichier structure accepte désormais le paramètre facultatif *****. Ce paramètre est utile dans le cadre de l'utilisation de composants.

- Lorsque la commande est appelée depuis un composant :
  - si le paramètre ***** est passé, la commande retourne le nom long du fichier de structure de la base hôte,
  - si le paramètre ***** n'est pas passé, la commande retourne le nom long du fichier de structure du composant.

Le fichier de structure d'un composant correspond au fichier .4db ou .4dc du composant situé dans le dossier "Components" de la base. Cependant, un composant peut également être installé sous la forme d'un alias/raccourci ou d'un dossier/package .4dbase :

- dans le cas d'un composant installé sous forme d'alias/raccourci, la commande retourne le chemin d'accès du fichier .4db ou .4dc original (l'alias ou le raccourci est résolu).

- dans le cas d'un composant installé sous forme de dossier/package .4dbase, la commande retourne le chemin d'accès du fichier .4db ou .4dc à l'intérieur de ce dossier/package.

- Lorsque la commande est appelée depuis une méthode d'une base hôte, elle retourne toujours le nom long du fichier de structure de la base hôte, que le paramètre * soit passé ou non.

▼ L'exemple suivant permet de savoir si la méthode est appelée depuis un composant :

```
C_BOOLEEN($0)
$0:=(Fichier structure#Fichier structure(*))
    ` $0=Vrai si la méthode est appelée depuis un composant
```

Dossier 4D

Dossier 4D ({dossier*{;*}*}) → Alpha

Paramètres	Type		Description
dossier	Entier long	→	Type de dossier
*		→	*Retourner le dossier de la base hôte*
Résultat	Alpha	←	Chemin d'accès du dossier désigné

La commande Dossier 4D accepte désormais un paramètre facultatif * ainsi que la nouvelle constante Dossier Resources courant :

Constante	Type	Valeur
Dossier Resources courant	Entier long	6

Ces nouveautés sont utiles dans le cadre de l'utilisation de composants.

- Lorsque la commande est appelée depuis un composant :
  - si le paramètre * est passé, la commande retourne pour les valeurs 4, 5 et 6 du paramètre *dossier* le chemin d'accès du dossier de la base hôte,
  - si le paramètre * n'est pas passé, la commande retourne pour les valeurs 4, 5 et 6 du paramètre *dossier* le chemin d'accès du dossier du composant.
    Le dossier de la base (valeurs 4 et 5) retourné diffère en fonction du type d'architecture du composant :
    - dans le cas d'un dossier/package .4dbase, la commande retourne le chemin d'accès du dossier/package .4dbase,
    - dans le cas d'un fichier .4db ou .4dc, la commande retourne le chemin d'accès du dossier "Components",

- dans le cas d'un alias ou raccourci, la commande retourne le chemin d'accès du dossier contenant la base matrice originale. Le résultat diffère en fonction du format de cette base (dossier/package .4dbase ou fichier .4db/.4dc), comme décrit ci-dessus.

- Lorsque la commande est appelée depuis la base hôte, elle retourne toujours pour les valeurs 4, 5 et 6 du paramètre *dossier* le chemin d'accès du dossier de la base hôte, que le paramètre * soit passé ou non.

  La nouvelle constante Dossier Resources courant permet de connaître le chemin d'accès du dossier *Resources* de la base hôte ou du composant.

Mode compile

Mode compile*{(*)}* → Booleen

Paramètres	Type		Description
*		→	*Retourner l'information de la base hôte*
Résultat	Booléen	←	Mode compilé (Vrai), mode interprété (Faux)

Note Dans les versions précédentes de 4D, cette commande était appelée Application compilee. Elle a été renommée pour plus de clarté dans 4D v11.

La commande Mode compile accepte désormais le paramètre facultatif *. Ce paramètre est utile dans le cadre de l'utilisation de composants.

- Lorsque la commande est appelée depuis un composant :
  - si le paramètre * est passé, la commande retourne Vrai ou Faux en fonction du mode d'exécution de la base hôte,
  - si le paramètre * n'est pas passé, la commande retourne Vrai ou Faux en fonction du mode d'exécution du composant.
- Lorsque la commande est appelée depuis une méthode d'une base hôte, elle retourne toujours Vrai ou Faux en fonction du mode d'exécution de la base hôte.

FIXER PARAMETRE BASE, Lire parametre base

FIXER PARAMETRE BASE({table; }sélecteur; valeur)
Lire parametre base({table; }sélecteur) → Entier long

Note Ces commandes étaient auparavant placées dans le thème "Définition structure".

Nouveaux sélecteurs

Sélecteur = 41 (Mode Unicode)

- Valeurs : 0 (mode compatibilité) ou 1 (mode Unicode)
- Description : Mode d'exécution courant de la base, relatif au jeu de caractères.
  4D v11 prend en charge le jeu de caractères Unicode mais peut fonctionner en mode "compatibilité" (basé sur le jeu de caractères ASCII Mac).
  Par défaut, les bases de données converties sont exécutées en mode compatibilité (0) et les bases créées en version 11 ou suivantes sont exécutées en mode Unicode. Le mode d'exécution est contrôlé via une option des préférences (cf. paragraphe "Compatibilité de l'Unicode avec les bases 4D", page 68). Il peut également être lu ou (à des fins de test) modifié via ce sélecteur.
  La modification de cette option nécessite le redémarrage de la base pour être prise en compte. A noter que, dans le cadre d'un composant, il n'est pas possible de modifier cette valeur, mais uniquement de la lire.

▼ Ce code permet de permuter le mode courant et de redémarrer la base :

```
Mode_unicode_courant:=Lire parametre base(Mode Unicode)
FIXER PARAMETRE BASE(Mode Unicode;1-Mode_unicode_courant)
OUVRIR FICHIER DONNEES(Fichier donnees)
```

Sélecteur = 42 (Taille mémoire temporaire)

- Valeurs : Entier long positif > 50 000 000
- Description : Taille de la mémoire temporaire exprimée en octets. Par défaut, cette taille est de 50 000 000 (50 Mo).
  4D v11 utilise une mémoire temporaire spéciale dédiée aux opérations d'indexation et de tri. Cette mémoire a pour but de préserver la mémoire cache "classique" lors d'opérations massives. Lorsque la taille maximale de la mémoire temporaire est atteinte (cas critique), 4D interrompt automatiquement la dernière opération demandée afin que cela ne pénalise pas les autres traitements.
  Dans la plupart des cas, la configuration par défaut de la mémoire temporaire sera parfaitement suffisante. Toutefois, dans certaines applications spécifiques effectuant des tris ou des indexations de façon très intensive, l'augmentation de la taille de cette mémoire pourra contribuer à améliorer sensiblement les performances. La valeur idéale sera à déterminer de façon empirique en fonction des spécificités de l'application et résultera généralement de tests en volumétrie réelle.

Sélecteur = 43 (SQL Autocommit)

- Valeurs : 0 (désactivation) ou 1 (activation)
- Description : Activation ou désactivation du mode SQL auto-commit. Par défaut, la valeur est 0 (mode désactivé).
  Le mode auto-commit permet de renforcer l'intégrité référentielle de la base. Lorsque ce mode est actif, les requêtes *SELECT*, *INSERT*, *UPDATE* et *DELETE* (SIUD) sont automatiquement incluses dans des transactions si ce n'est pas déjà le cas. Ce mode peut également être défini dans les préférences de la base. Pour plus d'informations, reportez-vous au paragraphe "Auto commit", page 219.

Sélecteurs modifiés

Sélecteur = 17 (Jeu de caractères)
Lorsque l'application est exécutée en mode Unicode, les valeurs à passer pour ce paramètre sont désormais des identifiants de jeux de caractères (*MIBEnum*, identifiants définis par l'IANA, cf. http://www.iana.org/assignments/character-sets). Voici la liste des identifiants correspondant aux jeux de caractères pris en charge par le serveur Web de 4D :

- 4 = ISO-8859-1
- 12 = ISO-8859-9
- 13 = ISO-8859-10
- 17 = Shift_JIS
- 2026 = Big5
- 38 = euc-kr
- 106 = UTF-8
- 2250 = Windows-1250
- 2251 = Windows-1251
- 2253 = Windows-1253
- 2255 = Windows-1255
- 2256 = Windows-1256

A noter également que, dans le cadre du sélecteur Jeu de caractères, la commande Lire parametre base retourne le nom IANA du jeu de caractères dans le paramètre facultatif *valeurAlpha*.

Sélecteur = 29 (Enreg requêtes Web)
Sélecteur = 30 (Client Enreg requêtes Web)

Dans 4D v11, de nouveaux formats de fichiers d'historique des requêtes sont disponibles (cf. paragraphe "Paramétrage de l'historique

des requêtes (logweb.txt)", page 247). Par conséquent, de nouvelles valeurs peuvent être associées aux sélecteurs 29 et 30. Ces nouvelles valeurs (2, 3 et 4) représentent ces formats.

- Valeurs possibles : 0, 1, 2, 3 ou 4
  - 0 = Ne pas enregistrer (défaut)
  - 1 = Enregistrer au format CLF
  - *2 = Enregistrer au format DLF*
  - *3 = Enregistrer au format ELF*
  - *4 = Enregistrer au format WLF*

Pour plus d'informations sur ces formats, reportez-vous au paragraphe "Format du fichier", page 247.

ATTENTION : *Les formats 3 et 4 sont des formats personnalisés, dont le contenu doit être défini au préalable dans les Préférences de l'application, page "*Web/Format du journal*". Si vous utilisez l'un de ces formats sans que ses champs n'aient été sélectionnés, le fichier d'historique n'est pas généré.*

Sélecteurs inactivés

Les sélecteurs suivants ne sont plus utilisés dans 4D v11 car ils correspondent à des optimisations ou des réglages désormais sans objet :

Sélecteur = 1 (Ratio de tri séq)
Sélecteur = 2 (Optimisation accès séq)
Sélecteur = 3 (Ratio valeurs distinctes séq)
Sélecteur = 4 (Compression index)
Sélecteur = 5 (Ratio chercher dans sélec séq)
Sélecteur = 26 (Mode écriture cache)

Par conséquent, ces sélecteurs sont ignorés par FIXER PARAMETRE BASE et Lire parametre base retourne 0 lorsqu'ils sont passés.

OUVRIR PREFERENCES 4D

OUVRIR PREFERENCES 4D(sélecteur)

Paramètres	Type		Description
sélecteur	Chaîne	→	Clé désignant un thème ou une page ou un groupe de paramètres des Préférences

Afin de refléter les modifications apportées à la boîte de dialogue des Préférences de 4D, la liste des clés acceptées par le paramètre *sélecteur* a été modifiée.

Clés modifiées :

• /Application/Compatibility/Design Compatibility *au lieu de /Application/Compatibility/Structure Compatibility*
• /Design Mode/Method Editor/Syntax Styles *au lieu de /Design Mode/Method Editor/Styles for Syntax Elements*
• /Database/International *au lieu de /Database/Script Manager*
• /Client-Server/Publishing/Allow-Deny Configuration Table *au lieu de /Client-Server/Publishing/Allow-Deny Table Configuration*

Nouvelles clés :

• /Application/Access/General Settings
• /Design Mode/Structure/Automatic Form Creation
• /Moving
• /Moving/Default Actions during the Copy if Dependent Objects
• /Moving/Moving Dialog
• /Database/International/Right-to-left Languages
• /Database/International/Numeric Display Format
• /Web/Options/Options
• /Web/Log Format
• /Web/Log Format/Web Log Type
• /Web/Log Format/Web Log Token Selection
• /Web/Log Scheduler
• /Web/Log Scheduler/Backup Frequency for Web Log File
• /SQL
• /SQL/Configuration
• /SQL/Configuration/SQL Server Access

LISTE SEGMENTS DE DONNEES, AJOUTER SEGMENT DE DONNEES

Le fichier de données ne peut plus être divisé en segments dans 4D v11. Les éventuels segments de données sont fusionnés dans les bases de données converties (cf. paragraphe “Conversion de bases en multi-segments”, page 26). Par conséquent, ces commandes sont désormais sans objet.

Environnement système

Selectionner couleur RVB

Selectionner couleur RVB({coulDéfaut{; message}}) → Entier long

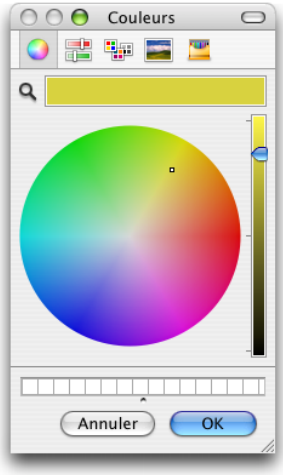
Paramètres	Type		Description
coulDéfaut	Entier long	→	Couleur RVB présélectionnée
message	Alpha 255	→	Titre de la fenêtre de sélection
Résultat	Entier long	←	Couleur RVB

La nouvelle commande Selectionner couleur RVB affiche la fenêtre système de sélection de couleur et retourne la valeur RVB de la couleur sélectionnée par l'utilisateur.

La fenêtre système de sélection de couleur a l'apparence suivante :

Windows

Mac OS



Le paramètre facultatif *coulDéfaut* vous permet de pré-sélectionner une couleur dans la fenêtre. Ce paramètre vous permet par exemple de restituer par défaut la dernière couleur définie par l'utilisateur.
Passez dans ce paramètre une valeur de couleur au format RVB (pour plus d'informations, reportez-vous à la description de la commande FIXER COULEURS RVB). Vous pouvez utiliser l'une des constantes du thème "FIXER COULEURS RVB".
Si le paramètre *coulDéfaut* est omis ou si vous passez 0, la couleur noir est sélectionnée à l'ouverture de la boîte de dialogue.

Le paramètre facultatif *message* vous permet de personnaliser le titre de la fenêtre système. Par défaut, si ce paramètre est omis, le libellé "Couleurs" est affiché.

Si l'utilisateur valide la boîte de dialogue, la commande retourne la valeur de couleur sélectionnée au format RVB et la variable système OK prend la valeur 1. Si l'utilisateur annule la boîte de dialogue, la commande retourne -1 et la variable système OK prend la valeur 0.

Note Cette commande ne doit pas être exécutée sur le poste serveur ni dans le cadre d'un process Web.

LIRE FORMATAGE SYSTEME

LIRE FORMATAGE SYSTEME(formatage; valeur)

Paramètres	Type		Description
formatage	Entier long	→	Formatage système à récupérer
valeur	Chaîne	←	Valeur de formatage définie dans le système

La commande LIRE FORMATAGE SYSTEME retourne la valeur courante de plusieurs paramètres régionaux définis dans le système d'exploitation. Cette commande permet de construire des formats personnalisés "automatiques" basés sur les préférences système.

Passez dans le paramètre *formatage* le type de paramètre dont vous souhaitez connaître la valeur. Le résultat est retourné directement par le système dans le paramètre *valeur* sous forme de chaîne de caractères.

Vous devez passer dans *formatage* une des constantes du thème "Formatages système". Voici un descriptif de ces constantes :

Constante (valeur)	Valeur(s) retournée(s)
Séparateur décimal (0)	Séparateur décimal (ex : ",")
Séparateur de milliers (1)	Séparateur de milliers (ex : " ")
Symbole monétaire (2)	Symbole monétaire (ex : "$")
Motif heure court (3)	Format d'affichage d'heure correspondant sous la forme "HH:mm:ss"
Motif heure abrégé (4)	
Motif heure long (5)	
Motif date court (6)	Format d'affichage de date correspondant sous la forme "dddd d MMMM yyyy"
Motif date abrégé (7)	
Motif date long (8)	

Constante (valeur)	Valeur(s) retournée(s)
Séparateur date (13)	Séparateur utilisé dans les formats de dates (ex : "/")
Séparateur heure (14)	Séparateur utilisé dans les formats d'heures (ex : ":")
Position jour date courte (15)	Position du jour, du mois et de l'année dans le format de date court : "1" = à gauche "2" = au milieu "3" = à droite
Position mois date courte (16)	
Position année date courte (17)	
Libellé AM heure système (18)	Libellé additionnel des heures d'avant midi dans les formats sur 12 heures (ex : "Matin")
Libellé PM heure système (19)	Libellé additionnel des heures après midi dans les formats sur 12 heures (ex : "Après-Midi")

- Sur un chèque rempli mécaniquement, les sommes inscrites sont généralement précédées de "*" afin d'empêcher les fraudes. Si le format d'affichage système standard pour la monnaie est "$ 5,422.33", le format pour les chèques devrait, lui, être du type "$***5432.33" : pas de virgule entre les milliers et pas d'espace entre le symbole $ et le premier chiffre.
  Le format à utiliser avec la fonction Chaine doit être "$*******.**". Pour le construire par programmation il est nécessaire de connaître le symbole monétaire et le séparateur décimal :

```
$MonFormat:="###"+LIRE FORMATAGE SYSTEME(Symbole monétaire) +
              "*******"+LIRE FORMATAGE SYSTEME(Séparateur décimal)+"**"
$Résultat:=Chaine(montant;$MonFormat)
```

Ecrans multiples sous Windows

Dans la version 11 de 4D, les commandes Nombre ecrans, COORDONNEES ECRAN, PROFONDEUR ECRAN et FIXER PROFONDEUR ECRAN prennent en charge les écrans multiples sous Windows.

PROPRIETES PLATE FORME

PROPRIETES PLATE FORME(plateForme{; système{; machine{; langue}}})

Paramètres	Type		Description
plateForme	Numérique	←	*2=Mac OS, 3=Windows*
système	Numérique	←	Dépend de la version (*inchangé*)
machine	Numérique	←	*Famille de processeur*
langue	Numérique	←	Dépend de la version (*inchangé*)

Les informations retournées par la commande PROPRIETES PLATE FORME ont été simplifiées afin de mieux correspondre aux besoins des développeurs et à l'évolution du matériel.
Les constantes associées à cette commande ont également été renommées.

- *plateForme*
Désormais, le paramètre *plateForme* indique strictement le système d'exploitation utilisé. Deux valeurs peuvent être renvoyées, disponibles sous forme de constantes :

Constante	Type	Valeur
Mac OS	Entier long	2
Windows	Entier long	3

- *machine*
Le paramètre *machine* retourne désormais la "famille" du microprocesseur de la machine. Deux valeurs peuvent être renvoyées, disponibles sous forme de constantes :

Constante	Type	Valeur
Power PC	Entier long	406
Compatible Intel	Entier long	586

La combinaison de ces deux paramètres permet par exemple de savoir sans ambiguïté si la machine utilisée est de type "MacIntel" (*plateForme*=Mac OS et *machine*=Compatible Intel).

Constantes

Certaines constantes du thème "Propriétés plate-forme" ont été modifiées afin de refléter l'évolution de la commande. Les constantes actives (listées ci-dessus) ont été renommées et les constantes désormais inutilisées ont été préfixées _O_ .

Constante 4D v11	Valeur	Nom versions précédentes
Compatible Intel	586	*Pentium*

Mac OS	2	*Power Macintosh*
Windows	3	Windows *(inchangé)*
Power PC	406	*Autres G3 et supérieurs*
_O_INTEL 386	386	*INTEL 386*
_O_INTEL 486	468	*INTEL 486*
_O_Macintosh 68K	1	*Macintosh 68K*
_O_PowerPC 601	601	*PowerPC 601*
_O_PowerPC 603	603	*PowerPC 603*
_O_PowerPC 604	604	*PowerPC 604*
_O_PowerPC G3	510	*PowerPC G3*

Interface utilisateur

La nouvelle commande Hauteur barre outils a été ajoutée et le fonctionnement de la commande Objet focus a été adapté à l'utilisation d'une list box associée à une expression.

Ce thème accueille également diverses commandes auparavant situées dans d'autres thèmes (cf. paragraphe "Changements de thèmes", page 429).

Hauteur barre outils

Hauteur barre outils → Entier long

Paramètres	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Entier long	←	Hauteur (exprimée en pixels) de la barre d'outils (retourne zéro si la barre d'outils n'est pas affichée)

La commande Hauteur barre outils retourne la hauteur de la barre d'outils, exprimée en pixels. Si la barre d'outils n'est pas affichée, la commande retourne 0.

Note Cette commande remplace avantageusement la commande AP Toolbar installed proposée dans les versions précédentes de 4D Pack.

Référence : Hauteur barre menus, CACHER BARRE OUTILS, AFFICHER BARRE OUTILS

Objet focus

Lorsqu'elle est utilisée dans le contexte d'une list box, cette fonction retourne :

- pour une colonne associée à un champ, un pointeur vers le champ associé,
- pour une colonne associée à une variable, un pointeur vers la variable,
- pour une colonne associée à une expression, un pointeur vers la variable de la list box.

Pour plus d'informations sur les nouveautés relatives aux list box en version 11, reportez-vous au paragraphe "List box", page 337.

Principes d'accès à la structure virtuelle

La *structure virtuelle* désigne les noms de tables et de champs qui ont été modifiés via les commandes FIXER TITRES TABLES et FIXER TITRES CHAMPS. Ces noms sont utilisés dans les éditeurs de 4D (recherche, états rapides, etc.) et les libellés dynamiques afin de contrôler la structure proposée à l'utilisateur.

Le principe d'accès à la structure virtuelle dans les éditeurs de 4D a été modifié dans 4D v11. Il dépend désormais du contexte d'appel des noms de tables et de champs : menus du mode Développement ou langage (mode Application). Le principe est que la structure virtuelle n'est PAS affichée lorsque l'éditeur est appelé directement depuis le mode Développement, et elle est affichée lorsque l'éditeur est appelé depuis une commande de langage (CHERCHER, TRIER...). Rappelons que les plug-ins accèdent toujours à la structure virtuelle et que la propriété "Invisible" est prioritaire.

Le tableau suivant récapitule ces principes :

Visibilité des éléments	**Structure virtuelle**	**Structure réelle**	**Tables / champs invisibles**
Mode Développement (éditeurs)		X	X
Plug-ins	X		
Mode Application	X		

Documents système

La commande Selectionner dossier admet un second paramètre et le paramètre *type* des commandes Ouvrir document, Creer document et Ajouter a document a été modifié.

Selectionner dossier

Selectionner dossier {(message*{;cheminDéfaut}*)} → Chaîne

Paramètres	Type		Description
message	Chaîne	→	Titre de la fenêtre de sélection
cheminDéfaut	*Chaîne* | *Entier long*	→	• *Chemin d'accès par défaut ou* • *Chaîne vide pour afficher le dossier d'utilisateur par défaut ("Mes documents" sous Windows, "Documents" sous MacOS), ou* • *Numéro de chemin d'accès mémorisé*
Résultat	Chaîne	←	Chemin d'accès du dossier sélectionné

Le nouveau paramètre *cheminDéfaut* vous permet de désigner un emplacement de dossier qui sera affiché initialement dans la boîte de dialogue de sélection de dossier.

Vous pouvez passer dans ce paramètre trois types de valeurs :

- un chemin d'accès de dossier valide utilisant la syntaxe de la plate-forme courante.
- une chaîne vide ("") pour afficher le dossier d'utilisateur par défaut du système ("Mes documents" sous Windows, "Documents" sous Mac OS).
- un numéro de chemin d'accès mémorisé (de 1 à 32000) pour afficher le dossier associé.
  Avec ce principe, vous pouvez conserver en mémoire le chemin d'accès du dossier ouvert au moment où l'utilisateur a cliqué sur le bouton de sélection, c'est-à-dire le dossier choisi par l'utilisateur. Lors du premier appel d'un numéro arbitraire (par exemple 5), la commande affiche le dossier d'utilisateur par défaut du système (équivaut à passer une chaîne vide). L'utilisateur peut alors éventuellement naviguer parmi les dossiers de son disque dur. Au moment où il clique sur le bouton de sélection, le chemin d'accès est mémorisé et associé au numéro 5. Lors des appels suivants du numéro 5, le chemin d'accès mémorisé sera utilisé par défaut. En cas de sélection d'un nouvel emplacement, le chemin numéro 5 sera mis à jour, et ainsi de suite.

Ce mécanisme vous permet de mémoriser jusqu'à 32000 chemins d'accès. Sous Windows, chaque chemin est conservé durant la session uniquement. Sous Mac OS, les chemins restent mémorisés d'une session à l'autre.

Note Ce mécanisme est identique à celui utilisé par la commande Selectionner document. Les numéros de chemins d'accès mémorisés sont partagés entre les deux commandes.

Si le chemin d'accès est incorrect, le paramètre *cheminDéfaut* est ignoré.

Ouvrir document, Ajouter a document, Creer document

Ouvrir document (document{; type{; mode}}) → DocRef
Ajouter a document (document{; type}) → DocRef
Creer document (document{; type}) → DocRef

Le paramètre *type* a été modifié pour ces commandes. Il est désormais possible de passer dans ce paramètre une liste de types de documents, séparés par des points-virgules (;). Ce mécanisme permet de restreindre les types de fichiers proposés dans les boîtes de dialogue d'ouverture ou de création de fichier (lorsque vous passez une chaîne vide dans *document*).

Images

La gestion des images a été modifiée dans 4D v11 afin de privilégier la prise en charge native des différents formats. Pour plus d'informations sur ces modifications, reportez-vous au paragraphe "Optimisation des variables et champs image", page 160. En outre, de nouvelles commandes permettent de manipuler les images natives.

Par ailleurs, ces modifications entraînent l'incompatibilité ou l'obsolescence de certaines commandes de gestion des images proposées dans les versions précédentes de 4D_Pack. Les commandes 4D_Pack désormais inutilisables sont décrites dans cette section.

LISTE CODECS IMAGES

LISTE CODECS IMAGES(tabCodecs{; tabNoms})

Paramètres	Type		Description
tabCodecs	Tab chaîne	←	Identifiants des codecs d'images disponibles
tabNoms	Tab chaîne	←	Noms des codecs d'images

La commande LISTE CODECS IMAGES remplit le tableau *tabCodecs* avec la liste des identifiants des codecs d'images disponibles sur la machine où elle est exécutée. Cette liste comporte à la fois les codecs des formats d'images gérés en natif par 4D v11 (cf. ci-dessous) ainsi que les identifiants des codecs QuickTime supplémentaires éventuellement installés sur le machine.

Ces identifiants peuvent être utilisés dans le paramètre *format* des commandes d'exportation d'images ECRIRE FICHIER IMAGE et IMAGE VERS BLOB.

Les identifiants des codecs peuvent être retournés dans le tableau *tabCodecs* sous trois formes :

- une extension (par exemple ".gif")
- un type *Mime* (par exemple "image/jpg")
- un code QuickTime sur 4 caractères (par exemple "PNTG")

La forme renvoyée par la commande dépend du mode de déclaration du codec au niveau du système d'exploitation.

Le tableau facultatif *tabNoms* permet de récupérer le nom de chaque codec. Ces noms sont plus explicites que les identifiants. Ce tableau permet par exemple de construire et d'afficher un menu listant les codecs disponibles.

Formats d'image natifs

4D v11 intègre la gestion native de plusieurs formats d'images. Ces formats sont disponibles dans tous les cas et seront toujours retournés par la commande LISTE CODECS IMAGES, quel que soit le système d'exploitation et la configuration de la machine. Ces formats sont :

- Jpeg
- Png
- Bmp
- Gif
- Tif

- Emf (sous Windows uniquement)
- Pict
- Pdf (sous Mac OS uniquement)

Note de compatibilité L'ancienne commande LISTE TYPES IMAGES est conservée pour des raisons de compatibilité. Toutefois, elle nécessite la présence de QuickTime et ne donne pas accès aux formats gérés en natif par 4D. Son intérêt est limité et elle est avantageusement remplacée par LISTE CODECS IMAGES.

Référence : CONVERTIR IMAGE

TRANSFORMER IMAGE

TRANSFORMER IMAGE(image; opérateur{; param1{; param2{; param3{; param4}}}})

Paramètres	Type		Description
image	Image	→	Image source à transformer
		←	Image résultant de la transformation
opérateur	Entier long	→	Type de transformation à effectuer
param1...4	Numérique	→	Paramètre(s) de la transformation

La nouvelle commande TRANSFORMER IMAGE permet d'appliquer une transformation de type *opérateur* à l'image passée dans le paramètre *image*.

Note Cette commande reprend et étend les fonctionnalités proposées par les opérateurs "classiques" de transformation d'images (+/, etc.). Ces opérateurs restent parfaitement utilisables dans 4D v11.

L'*image* source est modifiée directement à l'issue de l'exécution de la commande. A noter cependant qu'à l'exception de "Redimensionnement" et "Passage en niveaux de gris", les opérations ne sont pas destructives et permettent un retour en arrière via l'opération inverse ou l'opération "Réinitialisation". Par exemple, une image réduite à 1 % retrouvera sa taille originale sans altération si elle est agrandie 100 fois par la suite. Les transformations ne modifient pas le type d'origine de l'image : par exemple, une image vectorielle restera vectorielle à l'issue de la transformation.

Passez dans *opérateur* le numéro de l'opération à effectuer et dans *param* le ou les paramètres nécessaire(s) à cette opération (le nombre de paramètres dépend de l'opération). Vous pouvez utiliser dans *opérateur*

l'une des constantes du nouveau thème "Transformation des images". Ces opérateurs et leurs paramètres sont décrits dans le tableau suivant :

opérateur (valeur)	param1	param2	param3	param4	Valeurs
Réinitialisation (0)	-	-	-	-	
Redimensionnement (1)	Largeur	Hauteur	-	-	Facteurs
Translation (2)	Axe X	Axe Y	-	-	Pixels
Miroir horizontal (3)	-	-	-	-	
Miroir vertical (4)	-	-	-	-	
Recadrage (100)	Orig. X	Orig. Y	Largeur	Hauteur	Pixels
Passage en niveaux de gris (101)	-	-	-	-	

Des explications supplémentaires sont fournies ci-dessous :

- Réinitialisation : toutes les opérations matricielles effectuées sur l'image (redimensionnement, miroir...) sont annulées.
- Redimensionnement : l'image est redimensionnée horizontalement et verticalement en fonction des valeurs passées respectivement dans *param1* et *param2*. Ces valeurs sont des facteurs : par exemple, pour agrandir la largeur de 50 %, passez 1,5 dans *param1* et pour réduire la hauteur de 50 %, passez 0,5 dans *param2*.
- Translation : l'image est déplacée de *param1* pixels horizontalement et de *param2* pixels verticalement. Passez une valeur positive pour un déplacement vers la droite ou vers le bas et une valeur négative pour un déplacement vers la gauche ou vers le haut.
- Miroir horizontal et Miroir vertical : l'effet miroir est appliqué à l'image d'origine. Tout déplacement éventuel effectué auparavant ne sera pas pris en compte.
- Recadrage : l'image est recadrée à partir du point de coordonnées *param1* et *param2* (exprimé en pixels). La largeur et la hauteur de la nouvelle image sont déterminées par les paramètres *param3* et *param4*. Cette transformation ne peut pas être annulée.
- Passage en niveaux de gris : l'image est passée en niveaux de gris (aucun paramètre n'est requis). Cette transformation ne peut pas être annulée.

▼ Voici un exemple de recadrage (l'image est affichée dans le formulaire avec le format "Image tronquée (non centrée") :

TRANSFORMER IMAGE($vpRouages;Recadrage;50;50;100;100)

Image d'origine

Image résultante

Référence : COMBINER IMAGES

COMBINER IMAGES

COMBINER IMAGES(imageRésultat; image1; opérateur; image2{; décalHoriz; décalVert})

Paramètres	Type		Description
imageRésultat	Image	←	Image résultant de la combinaison
image1	Image	→	Première image à combiner
opérateur	Entier long	→	Type de combinaison à effectuer
image2	Image	→	Seconde image à combiner
décalHoriz	Entier long	→	Décalage horizontal pour la superposition
décalVert	Entier long	→	Décalage vertical pour la superposition

La nouvelle commande COMBINER IMAGES permet de combiner les images *image1* et *image2* en mode *opérateur* pour en produire une troisième, *imageRésultat*. L'image résultat est de type composé et conserve toutes les caractéristiques des images sources.

Note Cette commande reprend et étend les fonctionnalités proposées par les opérateurs "classiques" de combinaison d'images (+/, etc.). Ces opérateurs restent utilisables dans 4D v11.

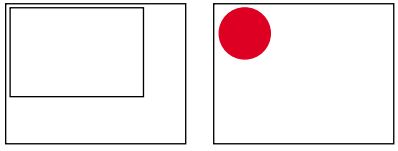
Passez dans *opérateur* le type de combinaison à appliquer.

Trois types de combinaisons sont proposés, accessibles via des constantes placées dans le thème "Transformation des images" :

- Concaténation horizontale (1) : *image2* est accolée à *image1*, le coin supérieur gauche de *image2* coïncidant avec le coin supérieur droit de *image1*.
- Concaténation verticale (2) : *image2* est accolée à *image1*, le coin supérieur gauche de *image2* coïncidant avec le coin inférieur gauche de *image1*.
- Superposition (3) : *image2* est placée par-dessus *image1*, le coin supérieur gauche de *image2* coïncidant avec le coin supérieur gauche de *image1*. Si les paramètres facultatifs *décalHoriz* et *décalVert* sont utilisés, une translation est appliquée à *image2* avant la superposition. Les valeurs passées dans *décalHoriz* et *décalVert* doivent correspondre à des pixels. Passez des valeurs positives pour un décalage vers la droite ou vers le bas et une valeur négative pour un décalage vers la gauche ou vers le haut.

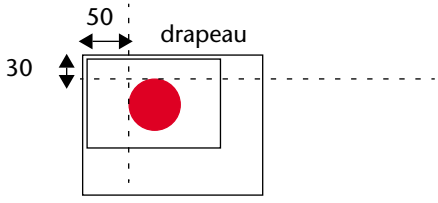
Note La superposition effectuée par la commande COMBINER IMAGES diffère de la superposition proposée par les opérateurs “classiques” & et | (superposition exclusive et superposition inclusive). Tandis que la commande COMBINER IMAGES conserve les caractéristiques de chaque image source dans l’image résultante, les opérateurs & et | traitent chaque pixel et génèrent une image bitmap dans tous les cas. Ces opérateurs, conçus à l’origine pour les images monochromes, sont désormais obsolètes.

▼ Soient les images suivantes :



COMBINER IMAGES(drapeau;monfond;Superposition;monrond;50;30)

Résultat :



Référence : TRANSFORMER IMAGE

CONVERTIR IMAGE

CONVERTIR IMAGE(image; codec)

Paramètres	Type		Description
image	Image	→	Image à convertir
		←	Image convertie
codec	Chaîne	→	Identifiant de codec d'image

La commande CONVERTIR IMAGE convertit *image* dans un nouveau type.

Le paramètre *codec* indique le type d'image à générer. Un *codec* peut être une extension (par exemple ".gif"), un type *Mime* (par exemple "image/jpg") ou un code QuickTime sur 4 caractères (par exemple "PNTG". Vous pouvez obtenir la liste des codecs disponibles via la nouvelle commande LISTE CODECS IMAGES.

Si le champ ou la variable *image* est de type composé (si par exemple elle est issue d'un copier-coller), seules les informations correspondant au type *codec* sont conservées dans l'image résultante.

▼ Conversion de l'image *vpPhoto* au format jpeg :

CONVERTIR IMAGE(vpPhoto;".jpg")

Référence : LISTE CODECS IMAGES

Commandes 4D Pack obsolètes

La plupart des commandes de gestion des images proposées dans 4D Pack sont désormais obsolètes. Lors de la conversion de bases de données existantes, ces commandes doivent être remplacées. Pour plus d'informations, reportez-vous à l'annexe C, "4D Pack v11", page 445

Graphes

4D v11 SQL comporte un moteur de rendu SVG intégré. SVG (*Scalable Vector Graphics*) est un format de fichier graphique vectoriel (extension .svg). Pour plus d'informations sur ce format, reportez-vous à l'adresse http://www.w3.org/Graphics/SVG/.

Pour vous permettre de tirer parti de ce moteur, les commandes GRAPHE et PARAMETRES DU GRAPHE acceptent désormais une variable image comme paramètre de zone. Dans ce cas, la représentation graphique est prise en charge par le moteur SVG.

Note La commande DOM EXPORTER VERS IMAGE, page 416 permet également de tirer parti du moteur de rendu SVG de 4D.

GRAPHE

GRAPHE(zoneGraphe; numGraphe; xCatégories; zValeurs{; zValeurs2;...; zValeursN})

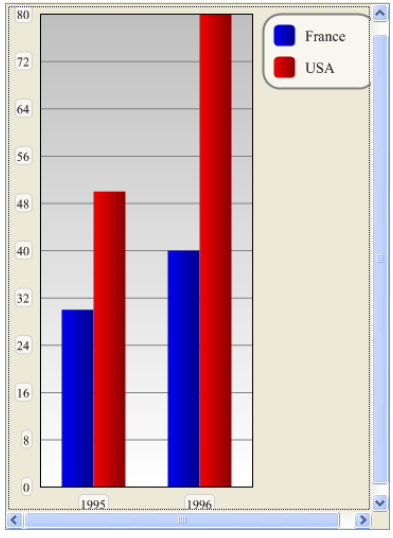
Paramètres	Type		Description
zoneGraphe	Variable graphe | *Variable image*	→	Zone de graphe dans le formulaire
numGraphe	Num	→	Numéro de type de graphe
xCatégories	Tableau	→	Catégories sur l'axe des x
zValeurs	Tableau	→	Valeurs à représenter graphiquement

La commande GRAPHE admet désormais une variable image dans le paramètre *grapheZone*. Dans ce cas, le graphe est généré via le moteur de rendu SVG intégré de 4D et non le moteur de graphe standard.

Les avantages de ce nouveau rendu sont nombreux : aspect plus moderne, possibilités d'ajout de barres de défilement, accès au menu contextuel en mode Application (cf paragraphe "Menu contextuel", page 163), etc.

Le reste de la syntaxe de la commande est inchangé.

▼ Voici un exemple de graphe réalisé via le moteur SVG ainsi que le code l'ayant généré :




```
C_IMAGE(vImGraphe)  ` Initialisation de la variable image
TABLEAU ALPHA(4;X;2)  ` Création d'un tableau pour l'axe des X
X{1}:="1995"  ` Premier libellé X
X{2}:="1996"  ` 2e libellé X
```

```
TABLEAU REEL(A;2)  ` Création d'un tableau pour l'axe des Z
A{1}:=30
A{2}:=40
TABLEAU REEL(B;2) ` Création d'un 2e tableau pour l'axe des Z
B{1}:=50
B{2}:=80
GRAPHE(vImGraphe;1;X;A;B) ` Dessiner le graphe (type 1)
PARAMETRES DU GRAPHE(vImGraphe;0;0;0;0;Faux;Faux;Vrai;"France";
                                    "USA")    ` Définition des légendes du graphe
```

PARAMETRES DU GRAPHE

PARAMETRES DU GRAPHE(zoneGraphe; xmin; xmax; ymin; ymax; xprop; grilleX; grilleY; titre{; titre2;...;titreN})

Paramètres	Type		Description
zoneGraphe	Variable graphe | *Variable image*	→	Zone de graphe dans le formulaire
xmin	Num, Date ou Heure	→	Valeur minimale de l'échelle des X pour graphe proportionnel
xmax	Num, Date ou Heure	→	Valeur maximale de l'échelle des X pour graphe proportionnel
ymin	Num	→	Valeur minimale de l'échelle des Y
ymax	Num	→	Valeur maximale de l'échelle des Y
xprop	Booléen	→	Vrai=Echelle des X proportionnelle Faux=Echelle des X normale
grilleX	Booléen	→	Vrai=Grille sur l'axe des X Faux=Pas de grille sur l'axe des X
grilleY	Booléen	→	Vrai=Grille sur l'axe des Y Faux=Pas de grille sur l'axe des Y
titre	Chaîne	→	Titre(s) des légende(s)

La commande PARAMETRES DU GRAPHE admet désormais une variable image dans le paramètre *grapheZone*. Ce principe permet de paramétrer les graphes générés par la commande GRAPHE via le moteur de rendu SVG intégré de 4D. Pour plus d'informations, reportez-vous à la description de la commande GRAPHE.

Le reste de la syntaxe de la commande est inchangé.

Chaînes de caractères

Dans le cadre de la prise en charge des conversions de chaînes en mode Unicode, deux nouvelles commandes ont été ajoutées (CONVERTIR DEPUIS TEXTE, Convertir vers texte) et la commande Code ascii a été renommée Code de caractere.

La nouvelle commande Lire traduction chaine permet de tirer parti de l'architecture XLIFF et la nouvelle commande Trouver regex manipule les "expressions régulières".

La commande Num accepte un nouveau paramètre désignant le séparateur décimal et la commande Chaine de nouveaux types d'expression. Les commandes Majusc, Minusc et Position acceptent des paramètres supplémentaires.

CONVERTIR DEPUIS TEXTE

CONVERTIR DEPUIS TEXTE(texte4D; jeuCaractères; blobConverti)

Paramètres	Type		Description
texte4D	Chaîne	→	Texte exprimé dans le jeu de caractères courant de 4D
jeuCaractères	Chaîne | Entier long	→	Nom ou Numéro de jeu de caractères
blobConverti	BLOB	←	BLOB contenant le texte converti

La commande CONVERTIR DEPUIS TEXTE permet de convertir un texte exprimé dans le jeu de caractères courant de 4D en un texte exprimé dans un autre jeu de caractères.

Passez dans le paramètre *texte4D* le texte devant être converti. Ce texte est exprimé dans le jeu de caractères de 4D. En version 11, 4D utilise le jeu de caractères Unicode.

Passez dans *jeuCaractères* le jeu de caractères à utiliser pour la conversion. Vous pouvez passer une chaîne contenant le nom standard du jeu (par exemple "ISO-8859-1" ou "UTF-8") ou son identifiant *MIBEnum*. Voici une liste (non exhaustive) de ces identifiants :

Identifiant	Nom IANA
1	UTF-16BE
2	UTF-16LE
7	UTF-8
8	UTF-7

9	US-ASCII
10	ebcdic-cp-us
100	x-mac-roman
101	windows-1252
102	x-mac-ce
103	windows-1250
104	x-mac-cyrillic
105	windows-1251
106	x-mac-greek
107	windows-1253
108	x-mac-turkish
109	windows-1254
110	x-mac-arabic
111	windows-1256
112	x-mac-hebrew
113	windows-1255
114	x-mac-ce
115	windows-1257
1000	Shift_JIS
1001	ISO-2022-JP
1002	Big5
1003	EUC-KR
1004	KOI8-R
1005	ISO-8859-1
1006	ISO-8859-2
1007	ISO-8859-3
1008	ISO-8859-4
1009	ISO-8859-5
1010	ISO-8859-6
1011	ISO-8859-7
1012	ISO-8859-8
1013	ISO-8859-9

Les noms standard des jeux de caractètres sont définis par l'IANA. La commande accepte le nom principal du jeu de caractères ainsi que tous ses alias référencés (par exemple, "IO-8859-1" peut être nommé "CP819", "csISOLatin1", "latin1" ou encore "l1"). Pour plus

d'informations sur les noms des jeux de caractères, reportez-vous à l'adresse http://www.iana.org/assignments/character-sets

Après l'exécution de la commande, le texte converti sera retourné dans le BLOB *blobConverti*. Ce BLOB pourra être relu par la commande Convertir vers texte.

Note Cette commande fonctionne uniquement lorsque 4D est exécuté en mode Unicode (l'option Unicode doit être cochée dans les préférences de 4D, cf. paragraphe "Compatibilité de l'Unicode avec les bases 4D", page 68). Si elle est utilisée dans le mode de compatibilité (non Unicode), *blobConverti* est retourné vide et la variable OK prend la valeur 0.
Pour plus d'informations, reportez-vous au paragraphe "Modifications liées à l'Unicode", page 258.

Si la commande a été correctement exécutée, la variable OK prend la valeur 1. Sinon, elle prend la valeur 0.

Référence : Convertir vers texte

Convertir vers texte

Convertir vers texte (blob; jeuCaractères) → Texte

Paramètres	Type		Description
blob	BLOB	→	BLOB contenant un texte exprimé dans un jeu de caractères spécifique
jeuCaractères	Chaîne | Entier long	→	Nom ou Numéro du jeu de caractères de blob
Résultat	Texte	←	Contenu de blob exprimé dans le jeu de caractères 4D

La commande Convertir vers texte convertit le texte contenu dans le paramètre *blob* et le retourne en texte exprimé dans le jeu de caractères de 4D. En version 11, 4D utilise le jeu de caractères Unicode.

Passez dans *jeuCaractères* le jeu de caractères dans lequel est exprimé le texte contenu dans *blob*, et qui sera utilisé pour la conversion. Vous pouvez passer une chaîne fournissant le nom standard du jeu (par exemple "ISO-8859-1" ou "UTF-8") ou son identifiant (entier long). Pour plus d'informations, reportez-vous à la description de la commande CONVERTIR DEPUIS TEXTE, page 380.

Note Cette commande fonctionne uniquement lorsque 4D est exécuté en mode Unicode (l'option Unicode doit être cochée dans les préférences de 4D, cf. paragraphe "Compatibilité de l'Unicode avec les bases 4D", page 68). Si elle est utilisée dans le mode de compatibilité (non Unicode), elle retourne une chaîne vide et la variable OK prend la valeur 0.
Pour plus d'informations, reportez-vous au paragraphe "Modifications liées à l'Unicode", page 258.

Si la commande a été correctement exécutée, la variable OK prend la valeur 1. Sinon, elle prend la valeur 0.

Référence : CONVERTIR DEPUIS TEXTE

Code de caractere

Code de caractere(caractère) → Numérique

Paramètres	Type		Description
caractère	Alpha	→	Caractère dont vous voulez obtenir le code
Résultat	Numérique	←	Code de caractère

Note Dans les versions précédentes de 4D, cette commande était appelée Code ascii. Elle a été renommée du fait de la prise en charge de l'Unicode.

La commande Code de caractere retourne le code du caractère passé en paramètre. Suivant le mode d'exécution de la base (mode Unicode ou mode compatibilité ASCII), la commande retourne le code Unicode ou le code ASCII du caractère.

Lire traduction chaine

Lire traduction chaine (resName) → Chaîne

Paramètres	Type		Description
resName	Chaîne	→	Nom d'attribut resname
Résultat	Chaîne	←	Valeur de la chaîne désignée par resName dans le langage courant

Cette commande fonctionne uniquement dans le cadre d'une architecture XLIFF. Pour plus d'informations sur ce point, reportez-vous au paragraphe "Prise en charge du standard XLIFF", page 101.

La commande Lire traduction chaine retourne la valeur de la chaîne désignée par l'attribut *resName* pour la langue courante.

Note La nouvelle commande Lire langue courante base permet de connaître la langue utilisée par l'application.

Passez dans *resName* le nom de ressource de la chaîne dont vous voulez obtenir la traduction dans la langue cible courante (*target*).

Si la commande a été exécutée correctement, la variable OK prend la valeur 1. Si *resName* n'est pas trouvé, la commande retourne une chaîne vide et la variable OK prend la valeur 0.

▼ Voici un extrait de fichier .xlf :

```
<file source-language="en-US" target-language="fr-FR">
  <trans-unit resname="Show on disk">
    <source>Show on disk</source>
    <target>Montrer sur disque</target>
  </trans-unit>
```

Après exécution de l'instruction suivante :

```
$valeurFR:=Lire traduction chaine("Show on disk")
```

... si la langue courante est le français, *$valeurFR* contient "Montrer sur disque".

Trouver regex

Trouver regex(motif; laChaîne{, début; pos_trouvée; long_trouvée; *}) → Booléen

Paramètres	Type		Description
motif	Texte	→	Expression régulière
laChaîne	Texte	→	Chaîne dans laquelle s'effectue la recherche
début	Numérique	→	Position dans laChaîne où doit débuter la recherche
pos_trouvée	Var Entier long | Tab Entier long	←	Position de l'occurence
long_trouvée	Var Entier long | Tab Entier long	←	Longueur de l'occurence
*	*	→	Si passé : rechercher uniquement à la position indiquée
Résultat	Booléen	←	Vrai = la recherche a trouvé une occurrence, Faux sinon

La nouvelle commande Trouver regex permet de tester la conformité d'une chaîne de caractères par rapport à un ensemble de règles synthétisé au moyen d'un méta-langage appelé "expression régulière" ou "expression rationnelle". L'abrévation *regex* est communément employée pour désigner ces familles de notations.

Passez dans *motif* l'expression régulière à rechercher. Il s'agit d'une suite de caractères chargée de décrire une chaîne de caractères, à l'aide de caractères spéciaux.

Passez dans *laChaîne* la chaîne dans laquelle rechercher l'expression régulière.

Passez dans *début* la position dans *laChaîne* où doit débuter la recherche.

Si *pos_trouvée* et *long_trouvée* sont des variables, la commande retourne la position et la longueur de l'occurrence dans ces variables. Si vous passez des tableaux, la commande retourne la position et la longueur de l'occurrence dans l'élément zéro des tableaux et les positions et longueurs des groupes capturés par l'expression régulière dans les éléments suivants.

Le paramètre facultatif * indique, lorsqu'il est passé, que la recherche doit s'effectuer à la position définie par *début* sans chercher plus loin en cas d'échec.

La commande retourne Vrai si la recherche a trouvé une occurrence.

Cette commande peut être utilisée de plusieurs manières :

- Recherche d'égalité complète :
vtrouvé:=Trouver regex(motif;montexte)

  Exemple :

  CHERCHER PAR FORMULE([Employés];**Trouver regex**(".*smith.*";[Employés]nom))

- Recherche dans le texte par position :
vtrouvé:=Trouver regex(motif;montexte; début; pos_trouvée; long_trouvée)

  Exemple pour afficher tous les tags de $1 :

  début:=1
  Repeter

```
  vtrouvé:=Trouver regex("<.*>";$1;début;pos_trouvée;long_trouvée)
  Si(vtrouvé)
    ALERTE(Sous chaine($1;pos_trouvée;long_trouvée))
    début:=pos_trouvée+long_trouvée
  Fin de si
Jusque(Non(vtrouvé))
```

- Recherche avec prise en charge des “groupes capturés” :
vtrouvé:=Trouver regex(motif;montexte; début; tab_pos_trouvée; tab_long_trouvée)

  Exemple :

```
TABLEAU ENTIER LONG(tab_pos_trouvée;0)
TABLEAU ENTIER LONG(tab_long_trouvée;0)
vtrouvé:=Trouver regex("(.*)truc(.*)";$1;1;tab_pos_trouvée;
tab_long_trouvée)
Si (vtrouvé)
  $group1:=Sous chaine($1;tab_pos_trouvée{1};tab_long_trouvée{1})
  $group2:=Sous chaine($1;tab_pos_trouvée{2};tab_long_trouvée{2})
Fin de si
```

- Recherche en limitant la comparaison de motif à la position indiquée :
Rajouter une étoile à la fin d’une des deux syntaxes précédentes.

  Exemple :

```
vtrouvé:=Trouver regex("a.b";"---a-b---";1;$pos_trouvée;$long_trouvée)
  `retourne Vrai

vtrouvé:=Trouver regex("a.b";"---a-b---";1;$pos_trouvée;$long_trouvée;*)
  `retourne Faux

vtrouvé:=Trouver regex("a.b";"---a-b---";4;$pos_trouvée;$long_trouvée;*)
  `retourne Vrai
```

Note Les positions et longueurs retournées n’ont de sens qu’en mode Unicode ou si le texte manipulé est de type ASCII 7 bits.

En cas d’erreur, la commande génère une erreur que vous pouvez intercepter via une méthode installée par la commande APPELER SUR ERREUR.

- Pour plus d’informations sur les *regex*, reportez-vous par exemple à l’adresse suivante :
http://fr.wikipedia.org/wiki/Expression_rationnelle

- Pour plus d'informations sur la syntaxe de l'expression régulière passée dans le paramètre *motif*, reportez-vous à l'adresse suivante : http://www.icu-project.org/userguide/regexp.html

Num

Num (expression{; *séparateur*}) → Numérique

Paramètres	Type		Description
expression	Alpha | Booléen | *Num*	→	Chaîne à convertir en numérique ou Booléen à convertir en 0 ou 1 *ou Expression numérique*
séparateur	*Chaîne*	→	*Séparateur décimal*
Résultat	Chaîne	←	Valeur numérique de l'expression

- La commande Num accepte une *expression* de type numérique. Dans ce cas, la commande retourne la valeur passée dans le paramètre *expression*. Ce fonctionnement est utile notamment dans la perspective d'une programmation générique utilisant des pointeurs.
- En outre, la commande Num accepte désormais un nouveau paramètre, permettant de désigner un séparateur décimal pour l'évaluation de *expression*.
Par défaut, la commande utilise le séparateur décimal défini dans le système d'exploitation. Lorsque la chaîne à évaluer est exprimée avec un séparateur décimal différent du séparateur système, la commande retourne un résultat incorrect.
Le nouveau paramètre *séparateur* permet dans ce cas d'obtenir une évaluation correcte. Lorsque ce paramètre est passé, la commande ne tient pas compte du séparateur décimal système. Vous pouvez passer un ou plusieurs caractères.

Note La nouvelle commande LIRE FORMATAGE SYSTEME permet de connaître le séparateur décimal courant ainsi que plusieurs autres paramètres système régionaux.

▼ Cet exemple compare les résultats obtenus en fonction du séparateur "courant" :

```
$lachaine:="33,333.33"
$lenum:=Num($lachaine)
    ` par défaut, $lenum vaut 33,33333 sur un système français
$lenum:=Num($lachaine;".")
    ` $lenum vaut bien 33 333,33 quel que soit le système
```

Chaine

Chaine (expression{; format}) → Alpha

Paramètres	Type		Description
expression		→	Expression à convertir en chaîne
format	Alpha I Num	→	Format d'affichage
Résultat	Alpha	←	Valeur alphanumérique de l'expression

La commande Chaine accepte deux types d'expressions supplémentaires : Alpha et booléen

- *expression* de type alpha : dans ce cas, la commande retourne la même valeur que celle passée en paramètre. Ce fonctionnement est utile notamment dans la perspective d'une programmation générique utilisant des pointeurs.
- *expression* de type booléen : dans ce cas, la commande retourne la chaîne "Vrai" ou "Faux" dans la langue de l'application ("True" ou "False" dans une version anglaise de 4D).

Dans les deux cas, le paramètre *format*, s'il est passé, est ignoré.

Majusc

Majusc (chaîne*{; *}*) → Alpha

Paramètres	Type		Description
chaîne	Alpha	→	Chaîne à convertir en majuscules
*	*	→	Si passé : conserver les accents
Résultat	Alpha	←	chaîne en majuscules

La commande Majusc admet désormais le paramètre facultatif * indiquant que les éventuels caractères accentués présents dans *chaîne* doivent être retournés sous forme de majuscules accentuées.
Par défaut, lorsque ce paramètre est omis, les caractères accentués "perdent" leur accent une fois la conversion effectuée.

▼ Cet exemple compare les résultats obtenus suivant que le paramètre * a été passé ou non :

```
$lachaine:=Majusc("hélène")  ` $lachaine vaut « HELENE »
$lachaine:=Majusc("hélène";*)  ` $lachaine vaut « HÉLÈNE »
```

Référence : Minusc

Minusc

Minusc (chaîne*{; *}*) → Alpha

Paramètres	Type		Description
chaîne	Alpha	→	Chaîne à convertir en minuscules
*	*	→	Si passé : conserver les accents
Résultat	Alpha	←	chaîne en minuscules

Le fonctionnement de la commande Minusc a été aligné sur celui de la commande Majusc en ce qui concerne les caractères accentués. En outre, elle accepte désormais le paramètre facultatif *.

- Par défaut, lorsque le paramètre * est omis, la commande Minusc retourne en minuscules et sans accents tous les caractères accentués présents dans *chaîne*.

Note de compatibilité Dans les versions précédentes de 4D, les caractères majuscules accentués éventuellement présents dans *chaîne* étaient retournés tels quels par la commande.

- Lorsque le paramètre facultatif * est passé, la commande conserve les accents.

▼ Cet exemple compare les résultats obtenus suivant que le paramètre * a été passé ou non :

```
$lachaine:=Minusc("DÉJÀ VU")  ` $lachaine vaut « deja vu »
$lachaine:=Minusc("DÉJÀ VU";*)  ` $lachaine vaut « déjà vu »
```

Référence : Majusc

Position

Position (àChercher; laChaîne*{; départ{; longTrouvée}}{; *}*) → Numérique

Paramètres	Type		Description
àChercher	Alpha	→	Chaîne à rechercher
laChaîne	Alpha	→	Chaîne dans laquelle effectuer la recherche
départ	*Numérique*	→	*Position dans laChaîne où débuter la recherche*
longTrouvée	*Entier long*	←	*Longueur de la chaîne trouvée*
*	*	→	*Si passé : recherche diacritique*
Résultat	Numérique	←	Position de la première occurrence de àChercher

La commande Position admet trois nouveaux paramètres facultatifs.

- Le paramètre *départ* permet de préciser le caractère de début de la recherche parmi *laChaîne*.
- Le paramètre *longTrouvée* retourne la longueur de la chaîne effectivement trouvée par la recherche. Ce paramètre est nécessaire pour pouvoir gérer correctement les lettres pouvant comporter un ou plusieurs caractères (ex : æ et ae, ß et ss...).
  A noter que si le paramètre * est passé (cf. ci-dessous), ces lettres ne sont pas considérées comme équivalentes (æ # ae).
- Le paramètre *, s'il est passé, indique que la recherche doit être diacritique, c'est-à-dire qu'elle doit tenir compte de la casse des caractères et des caractères accentués (a#A, a#à...).

▼ Ces exemples illustrent le fonctionnement des nouveaux paramètres :

```
$posTrouvée:=Position("day";"Today is the first day";1) `retourne 3
$posTrouvée:=Position("day";"Today is the first day";4) `retourne 20
$posTrouvée:=Position("DAY";"Today is the first day";1;*) `retourne 0

$posTrouvée:=Position("oe";"Nœud";1;$longchaine)
        `$posTrouvée retourne 2 et $longchaine retourne 1
```

▼ Dans l'exemple suivant, le paramètre *longTrouvée* permet de rechercher toutes les occurrences de "fluss" dans un texte, quelle que soit son orthographe ("fluss" ou "fluß") :

```
$départ:=1
Repeter
  $pos:=Position("fluss";$letexte;$départ;$longchaine)
  $départ:=$départ+$longchaine
Jusque($pos=0)
```

Note de compatibilité sur les caractères de contrôle

Dans 4D version 11, la norme Unicode interdit l'usage de caractères de contrôle dans les chaînes (par exemple Caractere(1)). Dans les versions précédentes de 4D, la commande Position pouvait fonctionner avec ces caractères. Par conséquent, dans certains cas il pourra être nécessaire d'adapter votre code. Soit par exemple le code suivant :

```
$s1:=Caractere(1)
$s2:="a"+Caractere(1)+"b"
$pos:=Posisition($s1;$s2;1)
```

- Dans 4D 200x, *$pos* vaut 2.
- Dans 4D v11, *$pos* vaut 1 car Caractere(1) étant ignoré, la recherche porte sur un contenu vide.

Pour connaître la liste des caractères interdits en Unicode, reportez-vous à l'adresse suivante :
http://www.unicode.org/charts/collation/chart_Null.html

Astuce : L'usage du paramètre * permet de conserver en v11 le fonctionnement précédent. Pour reprendre l'exemple ci-dessus, si vous passez $pos:=**Position**($s1;$s2;1;*) en v11, *$pos* retournera 2.

BLOB

Plusieurs commandes et constantes de ce thème ont été modifiées afin de prendre en compte le fonctionnement de l'application en mode Unicode (cf. paragraphe "Modifications liées à l'Unicode", page 258).

TEXTE VERS BLOB

TEXTE VERS BLOB(texte; blob; formatTexte{; offset | *})

Paramètres	Type		Description
texte	Chaîne	→	Texte à écrire dans blob
blob	BLOB	←	BLOB devant recevoir le texte
formatTexte	Numérique	→	Format *et jeu de caractères du texte*
offset | *	Variable | *	→	Offset (en octets) dans le BLOB ou * pour ajouter la valeur à la fin du BLOB
		←	Nouvel offset après l'écriture si * omis

4D v11 utilisant le jeu de caractères Unicode (UTF8), les BLOBs générés par cette commande seront différents de ceux générés dans les versions précédentes du programme (jeu de caractères ASCII Mac).

Par compatibilité, cette commande permet de "forcer" la conversion en utilisant le jeu de caractères Mac Roman. Le choix du jeu de caractères s'effectue via le paramètre *formatTexte*. Pour cela, les constantes existantes du thème "BLOB" ont été renommées et de nouvelles constantes ont été ajoutées :

Constante 4D v11 thème "BLOB"	Valeur	Nom versions précédentes
Mac Chaîne en C	0	*Chaîne en C*
Mac Chaîne pascal	1	*Chaîne pascal*
Mac Texte avec longueur	2	*Texte avec longueur*
Mac Texte sans longueur	3	*Texte sans longueur*
UTF8 Chaîne en C	4	-
UTF8 Texte avec longueur	5	-

Constante 4D v11 thème "BLOB"	Valeur	Nom versions précédentes
UTF8 Texte sans longueur	6	-

Notes
- Les constantes "UTF8" sont utilisables uniquement lorsque l'application fonctionne en mode Unicode.
- La constante Mac Texte avec longueur ne permet pas de traiter des textes de plus de 32 ko.

Le tableau suivant décrit chacun de ces formats :

Format Texte	Description	Exemples Mac	Exemples UTF8
Chaîne en C	Le texte se termine par un caractère NULL	"" → $00 "Café" → $43 61 66 8E 00	"" → $00 "Café" → $43 61 66 C3 A9 00
Chaîne pascal	Le texte est précédé d'un octet de longueur	"" → $00 "Café" → $04 43 61 66 8E	-
Texte avec longueur	Le texte est précédé de 2 octets (Mac) ou de 3 octets (UTF8) de longueur	"" → $00 00 "Café" → $00 04 43 61 66 8E	"" → $00 00 00 00 "Café" → $00 00 00 05 43 61 66 C3 A9
Texte sans longueur	Le texte est composé seulement de ses caractères	"" → Pas de valeur "Café" → $43 61 66 8E	"" → Pas de valeur "Café" → $43 61 66 C3 A9

Si vous souhaitez manipuler des jeux de caractères autres que UTF8, utilisez la nouvelle commande CONVERTIR DEPUIS TEXTE.

BLOB vers texte

BLOB vers texte (blob; formatTexte{; offset{; longueurTexte}}) → Texte

Paramètres	Type		Description
blob	BLOB	→	BLOB duquel extraire le texte
formatTexte	Numérique	→	Format *et jeu de caractères du texte*
offset	Variable	→	Offset (en octets) dans le BLOB
		←	Nouvel offset après la lecture
longueurTexte	Numérique	→	Nombre de caractères à lire
Résultat	Chaîne	←	Texte

4D v11 utilisant le jeu de caractères Unicode (UTF8), les textes manipulés par cette commande seront différents de ceux des versions précédentes du programme (jeu de caractères ASCII Mac).

Par compatibilité, cette commande permet de “forcer” l’utilisation du jeu de caractères Mac Roman. Le choix du jeu de caractères s’effectue via le paramètre *formatTexte*. Pour cela, les constantes existantes du thème “BLOB” ont été renommées et de nouvelles constantes ont été ajoutées :

Constante 4D v11	Valeur	Nom versions précédentes
Mac Chaîne en C	0	*Chaîne en C*
Mac Chaîne pascal	1	*Chaîne pascal*
Mac Texte avec longueur	2	*Texte avec longueur*
Mac Texte sans longueur	3	*Texte sans longueur*
UTF8 Chaîne en C	4	-
UTF8 Texte avec longueur	5	-
UTF8 Texte sans longueur	6	-

Notes • Les constantes “UTF8” sont utilisables uniquement lorsque l’application fonctionne en mode Unicode.
• La constante Mac Texte avec longueur ne permet pas de traiter des textes de plus de 32 ko.
Pour plus d’informations sur ces formats, reportez-vous à la description de la commande TEXTE VERS BLOB, page 391.

Si vous souhaitez manipuler des jeux de caractères autres que UTF8, utilisez la nouvelle commande Convertir vers texte.

Langage

La commande EXECUTER METHODE a été ajoutée dans le thème “Langage” dans 4D v11. Pour plus de clarté, la commande existante EXECUTER a été renommée EXECUTER FORMULE et a été déplacée dans le thème “Formules”.

EXECUTER METHODE

EXECUTER METHODE(nomMéthode; résultat | * {; param1{; param2;...;paramN}})

Paramètres	Type		Description
nomMéthode	Chaîne	→	Nom de méthode projet à exécuter
résultat	*	Variable	*	←	Variable recevant le résultat de la méthode ou * pour une méthode ne retournant pas de résultat
param1...paramN	Expression	→	Paramètre(s) de la méthode

La nouvelle commande EXECUTER METHODE provoque l'exécution de la méthode projet *nomMéthode* en lui passant éventuellement les paramètres *param1...paramN*. Vous pouvez passer tout nom de méthode appelable depuis la base ou le composant exécutant la commande.

Passez dans *résultat* une variable devant recevoir le résultat de l'exécution de *nomMéthode* (valeur placée dans $0 à l'intérieur de *nomMéthode*). Si la méthode ne retourne pas de résultat, passez * comme deuxième paramètre.

Le contexte d'exécution est préservé dans la méthode appelée, ce qui signifie que le formulaire courant et l'éventuel événement formulaire courant restent définis.

Si vous appelez cette commande depuis un composant et passez dans *nomMéthode* un nom de méthode appartenant à la base hôte (ou inversement), la méthode doit avoir été partagée (option "Partager entre composants et base hôte", cf. paragraphe "Partage des méthodes projet", page 61).

Si cette commande est exécutée correctement, la variable système OK prend la valeur 1, sinon elle prend la valeur 0.

EXECUTER FORMULE

La commande EXECUTER a été renommée EXECUTER FORMULE dans 4D v11 et a été déplacée dans le thème "Formules". Son fonctionnement est inchangé.

Cette modification a pour but de clarifier le fonctionnement de la commande et d'éviter toute confusion avec la nouvelle commande EXECUTER METHODE.

Self

La fonction Self peut désormais être utilisée dans le contexte d'une méthode projet appelée directement ou indirectement par une méthode objet.

Par ailleurs, lorsqu'elle est utilisée dans le contexte d'une list box, la fonction Self retourne :

- pour une colonne associée à un champ, un pointeur vers le champ associé,
- pour une colonne associée à une variable, un pointeur vers la variable,
- pour une colonne associée à une expression, un pointeur Nil.

Pour plus d'informations sur les nouveautés relatives aux list box en version 11, reportez-vous au paragraphe "List box", page 337.

Pointeurs sur variables process

Le fonctionnement des pointeurs sur les variables process a été modifié dans 4D v11. Dans les versions précédentes de 4D, il existait une différence entre le mode interprété et le mode compilé en matière de "dépointage" du pointeur dans des process différents.

Par exemple, dans le process 1 :

```
C_TEXTE(vVar)
vVar:="hello"
<>pointeur_vVar:=->vVar
```

Dans le process 2 :

```
ALERTE(<>pointeur_vVar->)
```

En mode compilé, la boîte de dialogue d'alerte affiche "hello". En mode interprété, dans les versions précédentes de 4D, la boîte de dialogue affichait dans ce cas une chaîne vide (valeur de *vVar* dans le process 2).

Désormais, le fonctionnement du mode interprété est aligné sur celui du mode compilé : dans les deux cas, la valeur "hello" est bien retrouvée.

Ressources

Deux commandes du thème “Ressources” ont été modifiées en version 11 afin de pouvoir être utilisées dans le contexte d’une architecture XLIFF : Lire chaine dans liste et LISTE DE CHAINES VERS TABLEAU.

Note Pour plus d’informations sur le XLIFF, reportez-vous au paragraphe “Prise en charge du standard XLIFF”, page 101.

Lire chaine dans liste

Lire chaine dans liste (resNum; strNum{; resFichier}) → Chaîne

La commande Lire chaine dans liste est compatible avec l’architecture XLIFF dans 4D v11 : la commande recherche dans un premier temps les valeurs correspondant à *resNum* et *strNum* dans tous les fichiers XLIFF ouverts (si le paramètre *resFichier* est omis). Dans ce cas, *resNum* désigne l’attribut **id** de l’élément **group** et *strNum* désigne l’attribut **id** de l’élément **trans-unit**.

Si la valeur n’est pas trouvée, la commande poursuit la recherche dans les fichiers de ressources ouverts.

LISTE DE CHAINES VERS TABLEAU

LISTE DE CHAINES VERS TABLEAU (resNum; chaînes{; resFichier})

La commande LISTE DE CHAINES VERS TABLEAU est compatible avec l’architecture XLIFF dans 4D v11 : la commande recherche dans un premier temps la valeur correspondant à *resNum* dans tous les fichiers XLIFF ouverts (si le paramètre *resFichier* est omis) et remplit le tableau *chaînes* avec les valeurs correspondantes. Dans ce cas, *resNum* désigne l’attribut **id** de l’élément **group** et le tableau *chaînes* contient toutes les chaînes de l’élément.

Si la valeur n’est pas trouvée, la commande poursuit la recherche dans les fichiers de ressources ouverts.

Communications

Les capacités des commandes ENVOYER PAQUET et RECEVOIR PAQUET ont été étendues, et la commande UTILISER FILTRE fonctionne différement en mode Unicode.

ENVOYER PAQUET

ENVOYER PAQUET({docRef; }paquet)

Paramètres	Type		Description
docRef	docRef	→	Référence de document ou canal courant
paquet	Chaîne | *BLOB*	→	Texte *ou BLOB* à envoyer

Cette commande permet désormais d'envoyer des paquets sous forme de BLOBs. Ce principe permet notamment de s'affranchir des contraintes liées à l'encodage des caractères envoyés en mode texte.

Note Lorsque vous passez un paramètre de type BLOB à cette commande, elle ne tient pas compte du jeu de caractères éventuellement défini par UTILISER FILTRE. Le BLOB est envoyé sans aucune modification.

▼ Cet exemple illustre l'envoi et la récupération de caractères étendus via un BLOB dans un document :

```
C_BLOB($blob_envoi)
C_BLOB($blob_reception)

TEXTE VERS BLOB("âzértÿ";$blob_envoi;UTF8 Texte sans longueur)
FIXER TAILLE BLOB($blob_envoi;16;255)
$blob_envoi{6}:=0
$blob_envoi{7}:=1
$blob_envoi{8}:=2
$blob_envoi{9}:=3
$blob_envoi{10}:=0

$vlDocRef:=Creer document("blob.test")
Si (OK=1)
  ENVOYER PAQUET($vlDocRef;$blob_envoi)
  FERMER DOCUMENT($vlDocRef)
Fin de si

$vlDocRef:=Ouvrir document(document)
Si (OK=1)
  RECEVOIR PAQUET($vlDocRef;$blob_reception;65536)
  FERMER DOCUMENT($vlDocRef)
Fin de si
```

Référence : RECEVOIR PAQUET

RECEVOIR PAQUET

RECEVOIR PAQUET({docRef; }réceptVar; stopCar | nbCar)

Paramètres	Type		Description
docRef	docRef	→	Référence de document ou canal courant
réceptVar	Variable chaîne | *Variable BLOB*	→	Variable devant revevoir les paquets
stopCar | nbCar	Alpha | Num	→	Caractère de stop ou nombre de caractères à recevoir

Cette commande peut désormais recevoir des paquets envoyés sous forme de BLOBs par la commande ENVOYER PAQUET. Dans ce cas, la variable *réceptVar* doit être de type BLOB.

▼ Reportez-vous à l'exemple de la commande ENVOYER PAQUET.

Note Lorsque cette commande reçoit un paquet de type BLOB, elle ne tient pas compte du jeu de caractères éventuellement défini par UTILISER FILTRE. Le BLOB est retourné tel quel, sans aucune modification.

Référence : ENVOYER PAQUET

UTILISER FILTRE

UTILISER FILTRE(filtre | *{; typeFiltre})

Lorsque l'application 4D est exécutée en mode Unicode, le paramètre *filtre*, s'il est passé, doit correspondre au nom "IANA" du jeu de caractères à utiliser, ou l'un de ses alias. Par exemple, les noms "iso-8859-1" ou "utf-8" sont des noms valides, ainsi que les alias "latin1" ou "l1". Pour plus d'informations sur ces noms, reportez-vous à l'adresse http://www.iana.org/assignments/character-sets.

Lorsque le paramètre * est passé, le jeu de caractères par défaut est rétabli. Dans 4D v11, ce jeu de caractères est UTF-8.

Note Lorsque l'application 4D est exécutée en mode non Unicode (mode compatibilité), le fonctionnement de cette commande est identique à celui des versions précédentes.

Définition structure

Plusieurs commandes du thème “Définition structure” ont été modifiées afin de prendre en compte la suppression des tables et des champs.
En outre, les commandes Champ, Nom du champ et FIXER INDEX ont été modifiées et les fonctions des commandes FIXER PARAMETRE BASE et Lire parametre base ont été enrichies.

Compter les tables et les champs

Dans 4D v11, il est désormais possible de supprimer des tables et des champs (cf. paragraphe “Suppression des tables et des champs”, page 34).

Cette nouvelle possibilité nécessite la modification des algorithmes utilisés pour dénombrer les tables et les champs. En effet, les commandes existantes Nombre de tables et Nombre de champs ne tiennent pas compte des éventuelles suppressions et donc des ruptures dans la numérotation de ces objets.

Par exemple, dans le cas d’une base comportant 4 tables numérotées 1, 2, 3 et 5 (la table 4 a été supprimée), Nombre de tables retournera 5, c’est-à-dire le numéro de table le plus élevé dans la base.

C’est la raison pour laquelle les commandes Nombre de tables et Nombres de champs ont été renommées respectivement Lire numero derniere table et Lire numero dernier champ. En outre, les commandes Est un numero de table valide et Est un numero de champ valide ont été ajoutées afin de permettre un décompte itératif fiable des tables et des champs d’une base. Voici un exemple de ce type d’algorithme :

```
Boucle($latable;1;Lire numero derniere table)
  Si (Est un numero de table valide($latable))
    Boucle($lechamp;1;Lire numero dernier champ($latable))
      Si(Est un numero de champ valide($latable;$lechamp))
        ... `Le champ existe et est valide
      Fin de si
    Fin de boucle
  Fin de si
Fin de boucle
```

Lire numero derniere table

Lire numero derniere table → Entier long

Paramètres	Type		Description
Résultat	Entier long	←	Numéro de table le plus élevé dans la base

Note Cette commande a été renommée, dans les versions précédentes de 4D elle était intitulée Nombre de tables.

La commande Lire numero derniere table retourne le numéro de table le plus élevé parmi les tables de la base.

Les tables sont numérotées dans l'ordre dans lequel elles ont été créées. Si aucune table n'a été supprimée dans la base, cette commande retourne donc le nombre de tables présentes dans la base. Dans le cadre de boucles itératives sur les numéros de tables de la base, vous devez utiliser la commande Est un numero de table valide afin de vérifier que la table n'a pas été supprimée.

▼ L'exemple suivant initialise les éléments du tableau *tabTables*. Ce tableau peut être utilisé comme liste déroulante (ou onglets, zone de défilement, etc.) pour afficher dans un formulaire la liste des tables de la base :

```
TABLEAU ALPHA (31; tabTables; Lire numero derniere table)
Boucle ($vlTables;Taille tableau(tabTables);1;-1)
  Si(Est un numero de table valide($vlTables))
    tabTables{$vlTables}:= Nom de la table ($vlTables)
  Sinon
    SUPPRIMER DANS TABLEAU(tabTables; $vlTables)
  Fin de si
Fin de boucle
```

Référence : Lire numero dernier champ, Est un numero de table valide

Lire numero dernier champ

Lire numero dernier champ (numTable | ptrTable) → Entier long

Paramètres	Type		Description
numTable | ptrTable	Num | Pointeur	→	Numéro de table ou Pointeur vers une table
Résultat	Entier long	←	Numéro de champ le plus élevé dans la table

Note Cette commande a été renommée, dans les versions précédentes de 4D, elle était intitulée Nombre de champs.

La commande Lire numero dernier champ retourne le numéro de champ le plus élevé parmi les champs de la table dont le numéro ou le pointeur est passé dans le paramètre *numTable* ou *ptrTable*.

Les champs sont numérotés dans l'ordre dans lequel ils ont été créés. Si aucun champ n'a été supprimé dans la table, cette commande retourne donc le nombre de champs que contient la table. Dans le cadre de boucles itératives sur les numéros de champs de la table, vous devez utiliser la commande Est un numero de champ valide afin de vérifier que le champ n'a pas été supprimé.

▼ La méthode projet suivante crée le tableau *taChamps* avec les noms des champs de la table dont le pointeur est reçu en paramètre :

```
$vlTable:=Table($1)
TABLEAU ALPHA(31;taChamps;Lire numero dernier champ($vlTable))
Boucle ($vlChamp;Taille tableau(taChamps);1;-1)
  Si(Est un numero de champ valide($vlTable;$vlChamp))
    taChamps{$vlChamp}:=Nom du champ($vlTable;$vlChamp)
  Sinon
    SUPPRIMER DANS TABLEAU(taChamps; $vlChamp)
  Fin de si
Fin de boucle
```

Référence : Lire numero derniere table, Est un numero de champ valide

Est un numero de table valide

Est un numero de table valide (numTable) → Booléen

Paramètres	Type		Description
numTable	Entier long	→	Numéro de table
Résultat	Booléen	←	Vrai = la table existe dans la base Faux = la table n'existe pas dans la base

La nouvelle commande Est un numero de table valide retourne Vrai si la table dont le numéro est passé dans *numTable* existe dans la base et Faux sinon. A noter que la commande retourne Faux si la table se trouve dans la Corbeille de l'Explorateur.

Cette commande permet de détecter d'éventuelles suppressions de tables, ce qui crée des ruptures dans la séquence des numéros de tables.

Référence : Lire numero derniere table, Est un numero de champ valide

Est un numero de champ valide

Est un numero de champ valide (numTable | ptrTable; numChamp) → Booléen

Paramètres	Type		Description
numTable | ptrTable	Num | Pointeur	→	Numéro de table ou Pointeur vers une table
numChamp	Entier long	→	Numéro de champ
Résultat	Booléen	←	Vrai = le champ existe dans la table Faux = le champ n'existe pas dans la table

La nouvelle commande Est un numero de champ valide retourne Vrai si le champ dont le numéro est passé dans *numChamp* existe dans la table dont le numéro ou le pointeur est passé dans le paramètre *numTable* ou *ptrTable*. Si le champ n'existe pas, la commande retourne Faux. A noter que la commande retourne Faux si la table du champ se trouve dans la Corbeille de l'Explorateur.
Cette commande permet de détecter d'éventuelles suppressions de champs, ce qui crée des ruptures dans la séquence des numéros de champs.

Référence : Lire numero dernier champ, Est un numero de table valide

FIXER INDEX

FIXER INDEX(champ; index{; mode}{; *})

Paramètres	Type		Description
champ	Champ	→	Champ duquel créer ou supprimer l'index
index	Booléen | *Entier*	→	Créer l'index (Vrai) ou le supprimer (Faux) *OU BIEN* *Créer un index : -1 = type Mots-clés, 0 = type par défaut, 1 = type B-Tree standard, 3 = type B-Tree cluster*
mode	Entier long	→	Utiliser mode d'indexation rapide (pourcentage)
*		→	Si passé = opération asynchrone

La commande FIXER INDEX a été modifiée afin de permettre la création des nouveaux types d'index.

Le paramètre *index* accepte désormais une valeur entière. Dans ce cas, la commande crée un index du type spécifié.

Vous pouvez passer une des constantes suivantes, placées dans le thème "Type index" :

- Index de mots clés (-1) : index de mots-clés, permettant l'indexation mot à mot du contenu du champ. Ce type d'index n'est utilisable qu'avec les champs de type Texte ou Alpha.
- Type index par défaut (0) : dans ce cas, 4D définit le type d'index (hors index de mots-clés) le plus optimisé en fonction du contenu du champ.
- Index BTree standard (1) : index de type B-Tree classique. Ce type d'index polyvalent est utilisé dans les versions précédentes de 4D.
- Index BTree cluster (3) : index de type B-Tree utilisant des clusters. Ce type d'index est optimisé lorsque l'index contient peu de clés, c'est-à-dire lorsque les mêmes valeurs reviennent souvent dans les données.

Pour plus d'information sur les nouveaux types d'index, reportez-vous au paragraphe "Gestion des index", page 39.

Note Les index créés par cette commande ne portent pas de nom. Ils ne pourront pas être supprimés par la commande SUPPRIMER INDEX via la syntaxe basée sur le nom.

Lorsque vous passez un entier dans *index*, le paramètre *mode* est inutile.

La commande accepte toujours un booléen dans le paramètre *index* (= syntaxe précédente). Si vous passez Vrai, la commande crée un index du type par défaut. Si vous passez Faux, la commande supprimera tous les index non composites associés au champ.

Notes • Cette commande ne permet pas de créer ou de supprimer des index composites.
• Cette commande ne permet pas de supprimer un index de mots-clés créé par la commande CREER INDEX.

▼ Cet exemple illustre les deux syntaxes possibles pour la commande :

```
  `Création d'un index de mots-clés :
FIXER INDEX([Livres]Résumé; -1)
  `Suppression de tous les index non composites sur le champ Résumé :
FIXER INDEX ([Livres]Résumé; Faux)
```

CREER INDEX

CREER INDEX(table; tabChamps; typeIndex; nomIndex{; *})

Paramètres	Type		Description
table	Table	→	Table pour laquelle créer un index
tabChamps	Tableau pointeur	→	Pointeur(s) vers le(s) champ(s) à indexer
typeIndex	Entier	→	Type d'index à créer : -1 = Mots-clés, 0 = par défaut, 1 = B-Tree standard, 3 = B-Tree cluster
nomIndex	Texte	→	Nom de l'index à créer
*		→	Si passé = indexation asynchrone

La nouvelle commande CREER INDEX permet de créer :

- un index standard sur un ou plusieurs champs (index composite) ou
- un index de mots-clés sur un champ.

L'index est créé pour la *table* en utilisant le ou les champ(s) désigné(s) par le tableau de pointeurs *tabChamps*. Ce tableau contient une seule ligne si vous souhaitez créer un index simple et deux ou plusieurs lignes si vous souhaitez créer un index composite (sauf index de mots-clés). Dans le cas d'index composites, l'ordre des champs dans le tableau est important lors de la construction de l'index.

Le paramètre *typeIndex* vous permet de définir le type d'index à créer. Vous pouvez passer une des constantes suivantes, placées dans le thème "Type index" :

- Index de mots clés (-1) : index de mots-clés. Attention, les index de mots-clés ne peuvent pas être composites : vous ne devez passer qu'un seul champ dans le tableau *tabChamps*.
- Type index par défaut (0) : dans ce cas, 4D définit le type d'index (hors index de mots-clés) le plus optimisé en fonction du contenu du champ.
- Index BTree standard (1) : index de type B-Tree classique. Ce type d'index polyvalent est utilisé dans les versions précédentes de 4D.
- Index BTree cluster (3) : index de type B-Tree utilisant des clusters. Ce type d'index est optimisé lorsque l'index contient peu de clés, c'est-à-dire lorsque les mêmes valeurs reviennent souvent dans les données.

Pour plus d'information sur les nouveaux types d'index, reportez-vous au paragraphe "Gestion des index", page 39.

Passez dans *nomIndex* le nom de l'index à créer. Ce nom est obligatoire, si vous passez une chaîne vide, une erreur est générée. Si l'index *nomIndex* existe déjà, la commande ne fait rien.

Le paramètre facultatif *****, lorsqu'il est passé, permet d'effectuer l'indexation en mode asynchrone. Dans ce mode, la méthode d'origine poursuit son exécution après l'appel de la commande, que l'indexation soit terminée ou non.

Si la commande CREER INDEX rencontre des enregistrements verrouillés, elle ne les indexe pas et attend qu'ils soient libérés.

Si une erreur se produit durant l'exécution de la commande (champ non indexable, tentative de création d'index de mots-clés sur plusieurs champs, etc.), une erreur est générée. Cette erreur peut être interceptée à l'aide d'une méthode d'appel sur erreur.

▼ Création de deux index standard sur les champs "Nom" et "Téléphone" de la table [Clients] :

```
TABLEAU POINTEUR(tabPtrChp;1)
tabPtrChp{1}:=->[Clients]Nom
CREER INDEX([Clients];tabPtrChp;Index BTree standard;"IdxCltNom")
tabPtrChp{1}:=->[Clients]Téléphone
CREER INDEX([Clients];tabPtrChp;Index BTree standard;"IdxCltTel")
```

- Création d'un index de mots-clés sur le champ "Observations" de la table [Clients] :

```
TABLEAU POINTEUR(tabPtrChp;1)
tabPtrChp{1}:=->[Clients]Observations
CREER INDEX([Clients];tabPtrChp;Index de mots clés;"IdxCltObs")
```

- Création d'un index composite sur les champs "CodePostal" et "Ville" de la table [Clients] :

```
TABLEAU POINTEUR(tabPtrChp;2)
tabPtrChp{1}:=->[Clients]CodePostal
tabPtrChp{2}:=->[Clients]Ville
CREER INDEX([Clients];tabPtrChp;Index BTree standard;"CPVille")
```

Référence : SUPPRIMER INDEX, FIXER INDEX

SUPPRIMER INDEX

SUPPRIMER INDEX(ptrChp | nomIndex{; *})

Paramètres	Type		Description
ptrChp | nomIndex	Pointeur | Texte	→	Pointeur vers le champ duquel supprimer les index OU Nom de l'index à supprimer
*		→	Si passé = opération asynchrone

La commande SUPPRIMER INDEX permet de supprimer un ou plusieurs index existant dans la base.

Vous pouvez passer en paramètre soit un pointeur vers un champ, soit un nom d'index :

- Si vous passez un pointeur vers un champ (*ptrChp*), tous les index associés au champ seront supprimés. Il peut s'agir d'index de mots-clés ou d'index standard. Si le champ est inclus dans un ou plusieurs index composite(s), ils sont également supprimés.
- Si vous passez nom d'index (*nomIndex*), seul l'index désigné sera supprimé. Il peut s'agir d'index de mots-clés ou d'index standard. Si le champ comporte d'autres index ou appartient à d'autres index composites, ils ne sont pas supprimés.

Le paramètre facultatif *, lorsqu'il est passé, permet d'effectuer la désindexation en mode asynchrone. Dans ce mode, la méthode d'origine poursuit son exécution après l'appel de la commande, que la suppression d'index soit terminée ou non.

S'il n'existe pas d'index correspondant à *ptrChp* ou à *nomIndex*, la commande ne fait rien.

▼ Cet exemple illustre les deux syntaxes de la commande :

```
  `Suppression de tous les index liés au champ Nom
SUPPRIMER INDEX(->[Clients]Nom)
  `Suppression de l'index nommé "CPVille"
SUPPRIMER INDEX("CPVille")
```

Référence : CREER INDEX, FIXER INDEX

Champ, Nom du champ

Lorsque vous passez un pointeur sur un sous-champ, ces commandes retournent désormais le numéro ou le nom du sous-champ et non celui du champ source.

Serveur Web

Le thème de commandes "Serveur Web" contient la nouvelle commande Valider mot de passe digest Web, utilisée dans le cadre du mode d'authentification Web *Digest*. La commande FIXER RACINE HTML a été modifiée.

Valider mot de passe digest Web

Valider mot de passe digest Web(nomUtilisateur; motDePasse) → Booléen

Paramètres	Type		Description
nomUtilisateur	Texte	→	Nom de l'utilisateur
motDePasse	Texte	→	Mot de passe de l'utilisateur
Résultat	Booléen	←	Vrai=Authentification correcte Faux=Echec de l'authentification

La commande Valider mot de passe digest Web permet de vérifier la validité des identifiants (nom et mot de passe) fournis par un utilisateur se connectant au serveur Web. Cette commande doit être utilisée dans la Méthode base Sur Authentification Web dans le cadre d'une authentification Web en mode Digest (cf. paragraphe "Authentification en mode Digest", page 243).

Passez dans les paramètres *nomUtilisateur* et *motDePasse* les identifiants de l'utilisateur conservés en local. La commande utilise ces identifiants pour générer une valeur qu'elle compare aux informations envoyées par le navigateur Web.

Si les valeurs sont identiques, la commande retourne Vrai. Sinon, elle retourne Faux.

Ce mécanisme vous permet de gérer et de maintenir par programmation votre propre système sécurisé d'accès au serveur Web. A noter que la validation Digest ne peut pas être utilisée conjointement avec les mots de passe 4D.

Note Si le navigateur ne prend pas en charge l'authentification Digest, une erreur est retournée (erreur d'authentification).

▼ Exemple de méthode base Sur authentification Web en mode Digest

```
  ` Méthode base Sur authentification Web
C_TEXTE($1;$2;$5;$6;$3;$4)
C_TEXTE($utilisateur)
C_BOOLEEN($0)

$0:=Faux
$utilisateur:=$5

  `Pour des raisons de sécurité, refuser les noms qui contiennent @
Si (AvecJoker($utilisateur))
  $0:=Faux
    `La méthode AvecJoker est décrite dans la documentation 4D
Sinon
  CHERCHER([WebUsers];[WebUsers]User=$utilisateur)
  Si (OK=1)
    $0:=Valider mot de passe digest Web($utilisateur;[WebUsers]Mdp)
  Sinon
    $0:=Faux `Utilisateur inexistant
  Fin de si
Fin de si
```

FIXER RACINE HTML

FIXER RACINE HTML(*dossierRacine*)

Paramètres	Type		Description
dossierRacine	Alpha	→	Chemin d'accès au dossier racine du serveur Web

Le fonctionnement de la commande FIXER RACINE HTML a été étendu dans 4D v11 :

■ Cette commande fonctionne désormais sans restriction en mode sans contexte. Elle modifie le dossier racine pour tous les process Web

pendant la session courante. Le chemin du dossier racine HTML défini dans les Préférences n'est pas modifié.

- Il est désormais possible d'exprimer l'emplacement du dossier racine sous forme de chemin d'accès absolu ("nom long") avec la syntaxe de la plate-forme courante. Le paramètre *dossierRacine* accepte donc désormais une chaîne du type :
  - (sous Mac OS) Disque:Applications:monserv:dossier
  - (sous Windows) C:\Applications\monserv\dossier

  Bien entendu, la syntaxe précédente (type "url" relatif) reste utilisable.

Note de compatibilité Dans les versions précédentes (mode contextuel), cette commande tenait compte du chemin éventuellement saisi dans les Préférences et concaténait les valeurs. Désormais, la commande définit intégralement le dossier racine et ne tient pas compte des préférences.

Outils

Le thème "Outils" contient la nouvelle commande Choisir ainsi que les nouvelles commandes utilitaires LIRE PARAMETRE MACRO et FIXER PARAMETRE MACRO, permettant d'utiliser les macros de l'éditeur de méthodes dans un environnement multi-composants.

En outre, les commandes existantes LANCER PROCESS EXTERNE et FIXER VARIABLE ENVIRONNEMENT ont été modifiées.

Note Le thème "Outils" dans 4D v11 accueille également des commandes déplacées d'autres thèmes (cf. paragraphe "Changements de thèmes", page 429).

Choisir

Choisir (critère; valeur1{; valeur2;...; valeurN}) → Valeur

Paramètres	Type		Description
critère	Booléen | Entier	→	Valeur à tester
valeur1...N	Expression	→	Valeurs possibles
Résultat	Expression	←	valeur1 ou valeur2 ou valeurN en fonction de la valeur de critère

La commande Choisir retourne l'une des valeurs passées dans les paramètres *valeur1, valeur2*, etc. en fonction de la valeur du paramètre *critère*.

Vous pouvez passer un paramètre *critère* de type booléen ou numérique :

- Si *critère* est un booléen, Choisir retourne *valeur1* si le booléen vaut Vrai et *valeur2* si le booléen vaut Faux.
  Dans ce cas, la commande attend exactement trois paramètres : *critère*, *valeur1* et *valeur2*.
- Si *critère* est un entier, Choisir retourne la *valeur* dont la position correspond à *critère*. Attention, la numérotation des valeurs débute à 0 (la position de *valeur1* est 0).
  Dans ce cas, la commande attend au minimum deux paramètres : *critère* et *valeur1*.

La commande accepte tous les types de données pour le(s) paramètre(s) *valeur*, hormis les images, pointeurs, BLOBS et tableaux. Veillez cependant à ce que toutes les valeurs passées soient du même type, 4D n'effectue pas de vérification sur ce point.

Si aucune *valeur* ne correspond à *critère*, Choisir retourne une valeur "nulle" en rapport avec le type du paramètre *valeur* (par exemple 0 pour le type numérique, "" pour le type chaîne, etc.).

Cette commande permet de générer du code concis remplaçant des tests du type "Au cas ou" sur plusieurs lignes (cf. exemple 2). Elle est également très utile dans les emplacements où des formules peuvent être exécutées : éditeur de recherches, appliquer une formule, éditeur d'états rapides, colonne calculée de list box, etc.

▼ Voici une utilisation type de la commande avec un critère de type booléen :

```
vTitre:=Choisir([Personne]Masculin;"Mr";"Madame")
```

Ce code est strictement équivalent à :

```
Si([Personne]Masculin)
   vTitre:="Mr"
Sinon
   vTitre:="Madame"
Fin de si
```

▼ Voici une utilisation type de la commande avec un critère de type numérique :

```
vStatut:=Choisir([Personne]Statut;"Célibataire";"Marié";'"Veuf";"Divorcé")
```

Ce code est strictement équivalent à :

```
Au cas ou
   :([Personne]Statut=0)
      vStatut:="Célibataire"
   :([Personne]Statut=1)
      vStatut:="Marié"
   :([Personne]Statut=2)
      vStatut:="Veuf"
   :([Personne]Statut=3)
      vStatut:="Divorcé"
Fin de cas
```

LIRE PARAMETRE MACRO

LIRE PARAMETRE MACRO(sélecteur; paramTexte)

Paramètres	Type		Description
sélecteur	Entier long	→	Sélection à utiliser
paramTexte	Texte	←	Texte récupéré

La commande LIRE PARAMETRE MACRO retourne dans *paramTexte* une partie ou la totalité du texte de la méthode depuis laquelle elle a été appelée.

Le paramètre *sélecteur* permet de définir le type d'information à récupérer. Vous pouvez passer l'une des constantes suivantes, ajoutées dans le thème "Environnement 4D" :

Constante	Type	Valeur
Texte méthode	Entier long	1
Texte méthode surligné	Entier long	2

Si vous passez Texte méthode dans *sélecteur*, la totalité du texte de la méthode sera retourné dans *paramTexte*. Si vous passez Texte méthode surligné dans *sélecteur*, seul le texte sélectionné dans la méthode sera retourné dans *paramTexte*.

Référence : FIXER PARAMETRE MACRO

FIXER PARAMETRE MACRO

FIXER PARAMETRE MACRO(sélecteur; paramTexte)

Paramètres	Type		Description
sélecteur	Entier long	→	Sélection à utiliser
paramTexte	Texte	→	Texte envoyé

La commande FIXER PARAMETRE MACRO insère le texte *paramTexte* dans la méthode depuis laquelle elle a été appelée.

Si du texte était sélectionné dans la méthode, le paramètre *sélecteur* permet de définir si le texte *paramTexte* doit remplacer la totalité de la méthode ou uniquement le texte sélectionné. Vous pouvez passer dans *sélecteur* l'une des constantes suivantes, ajoutées dans le thème "Environnement 4D" :

Constante	Type	Valeur
Texte méthode	Entier long	1
Texte méthode surligné	Entier long	2

Si aucun texte n'était sélectionné, *paramTexte* est inséré dans la méthode.

▼ Exemple :

```
C_TEXTE($texte_entrée)
C_TEXTE($texte_sortie)
LIRE PARAMETRE MACRO(Texte méthode surligné;$texte_entrée)
    `Supposons que le texte sélectionné est une table, i.e. "[Clients]"
    `Cette macro construit un nouveau texte qui sera retourné à la
    `méthode appelante
$texte_sortie:=""
$texte_sortie:=$texte_sortie+Nom commande(47)+"("+$texte_entrée+")"
    ` Tout sélectionner ([Clients])
$texte_sortie:=$texte_sortie+"$i:="+Nom commande(76)+"("+
                                                    $texte_entrée+")"
    ` $i:=Enregistrements trouves([Clients])
$texte_sortie:=$texte_sortie+...etc
FIXER PARAMETRE MACRO(Texte méthode surligné;$texte_sortie)
    `On remplace le texte sélectionné par le nouveau code
```

Référence : LIRE PARAMETRE MACRO

Note Pour que les commandes LIRE PARAMETRE MACRO et FIXER PARAMETRE MACRO fonctionnent correctement, le nouvel attribut "version" doit être déclaré dans la macro elle-même. L'attribut "version" doit être

déclaré ainsi :
<macro name="MaMacro" version="2">
--- Texte de la macro ---
</macro>

LANCER PROCESS EXTERNE

LANCER PROCESS EXTERNE(nomFichier{; fluxEntrée{; fluxSortie{; fluxErreur}}})

Note Dans les versions précédentes de 4D, cette commande était placée dans le thème “Environnement système” (cf. paragraphe “Changements de thèmes”, page 429).

Il est désormais possible d’exécuter un process externe de façon asynchrone via cette commande. Pour cela, il suffit d’utiliser la variable d’environnement _4D_OPTION_BLOCKING_EXTERNAL_PROCESS (cf. commande FIXER VARIABLE ENVIRONNEMENT). Dans les versions précédentes de 4D, seule l’exécution synchrone du process était possible.

A noter que dans ce cas, la commande ne retourne pas de valeur pour les paramètres *fluxSortie* et *fluxErreur*.

Référence : FIXER VARIABLE ENVIRONNEMENT

FIXER VARIABLE ENVIRONNEMENT

FIXER VARIABLE ENVIRONNEMENT(nomVar; valeurVar)

Note Dans les versions précédentes de 4D, cette commande était placée dans le thème “Environnement système” (cf. paragraphe “Changements de thèmes”, page 429).

Une nouvelle variable d’environnement est disponible dans le contexte de l’utilisation conjointe de cette commande avec LANCER PROCESS EXTERNE : _4D_OPTION_BLOCKING_EXTERNAL_PROCESS. Cette variable permet d’exécuter le process externe en mode asynchrone, c’est-à-dire non bloquant pour les autres applications. Vous devez passer “false” dans *valeurVar* pour l’exécution asynchrone.

Référence : LANCER PROCESS EXTERNE

Saisie

DIALOGUE

DIALOGUE({table; }formulaire*{; *}*)

Paramètres	Type		Description
table	Table	→	Table à laquelle appartient le formulaire ou Table par défaut si ce paramètre est omis
formulaire	Formulaire	→	Formulaire à afficher comme dialogue
*	*	→	*Utiliser le même process*

La commande DIALOGUE accepte désormais un astérisque * en dernier paramètre. Lorsque ce paramètre est passé, le *formulaire* est chargé et affiché dans la dernière fenêtre ouverte du process courant et la commande termine son exécution en laissant le formulaire actif à l'écran.
Ce formulaire réagit alors "normalement" aux actions de l'utilisateur et est fermé lorsque du code 4D lié au formulaire (méthode objet ou méthode formulaire) appelle la commande NE PAS VALIDER ou VALIDER. Si le process courant se termine, les formulaires créés de cette façon sont automatiquement fermés en simulant un NE PAS VALIDER.

Ce mode d'ouverture est particulièrement utile pour afficher une palette flottante en rapport avec un document, sans pour autant nécessiter un autre process :

```
  `Affichage palette d'outils
$window_palette:=Creer fenetre formulaire("tools";Form fenêtre palette)
DIALOGUE("tools";*) `Rend la main immédiatement

  `Affichage fenêtre document principal
$window_document:=Creer fenetre formulaire("doc";Form fenêtre standard)
DIALOGUE("doc")
```

Sélections

APPLIQUER A SELECTION

APPLIQUER A SELECTION(*table*; formule)

Le paramètre *table* est désormais obligatoire.

Numero de ligne affichee

Cette commande fonctionne désormais dans le contexte d'une list box.

Pour plus d'informations sur les nouveautés relatives aux list box en version 11, reportez-vous au paragraphe "List box", page 337.

Transactions

4D v11 autorise désormais les transactions imbriquées (cf. paragraphe "Extension des capacités", page 33). Afin de permettre une gestion optimisée de ce mécanisme, la commande Niveau de la transaction a été ajoutée dans le thème "Transactions".

Enregistrements créés en transaction

Le principe de gestion des enregistrements créés durant une transaction a été modifié dans 4D v11. Désormais, les enregistrements créés durant une transactions ne reçoivent plus de numéro temporaire (débutant à 18 000 000) mais un numéro standard, correspondant à la numérotation en cours dans la table. Ce numéro devient définitif en cas de validation de la transaction et est libéré en cas d'annulation (les enregistrements créés sont alors supprimés).

Les commandes ANNULER TRANSACTION et VALIDER TRANSACTION ne modifient plus la sélection courante et les commandes fonctionnant avec les numéros d'enregistrements (SCAN INDEX, JOINTURE...) peuvent désormais être utilisées sans restriction dans les transactions.

Niveau de la transaction

Niveau de la transaction → Entier long

Paramètres	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Entier long	←	Niveau de transaction courant (0 si aucune transaction n'a été démarrée)

La commande Niveau de la transaction retourne le niveau de transaction courant pour le process. Cette commande prend en compte toutes les transactions du process courant, qu'elles aient été démarrées via le langage de 4D ou via le SQL.

XML

Nouvelles commandes

DOM Chercher element XML par ID

DOM Chercher element XML par ID (refElément; id) → Chaîne

Paramètres	Type		Description
refElément	Alpha	→	Référence d'élément XML
id	Alpha	→	Valeur de l'attribut ID de l'élément à chercher
Résultat	Chaîne	←	Référence de l'élément trouvé (le cas échéant)

La commande DOM Chercher element XML par ID vous permet de rechercher un élément XML sur la base de la valeur de son attribut ID. La recherche débute à l'élément désigné par le paramètre *refElément*.

L'attribut ID permet d'associer un identificateur unique à chaque élément. La valeur de l'attribut ID doit être un nom XML valide et doit être unique. Passez dans *id* la valeur de l'attribut à rechercher.

La commande retourne en résultat la référence XML de l'élément trouvé.

DOM EXPORTER VERS IMAGE

DOM EXPORTER VERS IMAGE(refElément; vVarImage{; typeExport})

Paramètres	Type		Description
refElément	Chaîne	→	Référence d'élément XML racine
vVarImage	Image	→	Variable image devant recevoir l'arbre XML (image SVG)
typeExport	Entier long	→	0=Ne pas stocker la source de données 1=Copier la source de données 2 (défaut) = Prendre possession de la source de données

La commande DOM EXPORTER VERS IMAGE permet de sauvegarder dans la variable ou le champ image désigné(e) par le paramètre *vVarImage* une image au format SVG contenue dans un arbre XML.

SVG (*Scalable Vector Graphics*) est un format de fichier permettant de décrire en XML un graphique vectoriel (extension .svg). Ces fichiers peuvent être visualisés dans les navigateurs Web, soit nativement, soit

via des plug-ins. 4D v11 comporte un moteur de rendu SVG vous permettant de visualiser les fichiers SVG dans les champs ou les variables image. L'utilisation la plus courante du SVG est la publication de données statistiques ou cartographiques. Pour plus d'informations sur ce format, reportez-vous à l'adresse http://www.w3.org/Graphics/SVG/.

Passez dans *refElément* la référence de l'élément XML racine contenant l'image SVG.

Passez dans *vVarImage* le nom de la variable image ou du champ image 4D devant contenir l'image SVG. L'image est exportée dans son format natif (description XML) et est dessinée via le moteur de rendu SVG au moment de l'affichage.

Le paramètre facultatif *typeExport* vous permet de définir la manière dont la source de données XML doit être prise en charge par la commande. Vous pouvez passer dans ce paramètre une des constantes suivantes, placées dans le thème "XML" :

- Lire source données XML (0) : 4D lit uniquement la source de données XML, elle n'est pas conservée avec l'image. Ce paramétrage accélère sensiblement l'exécution de la commande, toutefois l'arbre DOM n'étant pas conservé, il n'est pas possible de stocker ni d'exporter l'image.
- Copier source données XML (1) : 4D conserve une copie de l'arbre DOM avec l'image, ce qui permet de la sauvegarder dans un champ image de la base de données et de la réafficher ou de l'exporter à tout moment.
- Posséder source données XML (2) : 4D exporte l'arbre DOM avec l'image. L'image pourra être stockée ou exportée et l'exécution de la commande est rapide. Toutefois, la référence XML *refElément* n'est alors plus utilisable par les autres commandes 4D.
  Ce mode d'exportation est utilisé par défaut si le paramètre *typeExport* est omis.

Note Les commandes 4D du thème "Graphes" ont également été modifiées afin de vous permettre de bénéficier du moteur de rendu SVG. Pour plus d'informations, reportez-vous au paragraphe "Graphes", page 377.

▼ L'exemple suivant permet d'afficher "Hello World" dans une image 4D :

```
C_IMAGE(vImage)
$svg:=DOM Creer ref XML("svg";"http://www.w3.org/2000/svg")
$ref:=DOM Creer element XML($svg;"text";"font-size";26;"fill";"red")
DOM ECRIRE VALEUR ELEMENT XML($ref;"Hello World")
DOM EXPORTER VERS IMAGE($svg;vImage;Copier source données XML)
DOM FERMER XML($svg)
```

Commandes modifiées

DOM Chercher element XML

DOM Chercher element XML (refElément; xChemin*{; tabRefEléments}*) → Chaîne

Paramètres	Type		Description
refElément	Alpha	→	Référence d'élément XML
xChemin	Texte | BLOB	→	Chemin xPath de l'élément à chercher
tabRefEléments	*Tab Alpha 16*	←	*Liste des références d'éléments trouvés (le cas échéant)*
Résultat	Chaîne	←	Référence de l'élément trouvé (le cas échéant)

La commande DOM Chercher element XML accepte désormais un troisième paramètre, *tabRefEléments*, de type tableau alpha. Ce paramètre est facultatif.

Lorsque le tableau *tabRefEléments* est passé, la commande le remplit avec la liste des références XML trouvées. Dans ce cas, la commande retourne en résultat le premier élément du tableau *tabRefEléments*. Ce paramètre est utile lorsque plusieurs éléments de même nom existent à l'emplacement désigné par le paramètre *xChemin*.

▼ Soit la structure XML suivante :

```
<Racine>
  <Elem1>
    <Elem2>aaa</Elem2>
    <Elem2>bbb</Elem2>
    <Elem2>ccc</Elem2>
  </Elem1>
</Racine>
```

Le code suivant permet de récupérer la référence de chaque élément Elem2 dans le tableau *tAtrouvés* :

```
TABLEAU ALPHA(16;tAtrouvés;0)
vTrouvé:=DOM Chercher element XML(vRefElem;"/Racine/Elem1/Elem2";
                                  tAtrouvés)
```

DOM Creer element XML, DOM ECRIRE ATTRIBUT XML

DOM Creer element XML(refElément; xChemin{; nomAttribut;*valeurAttribut*}...{; nomAttributN; *valeurAttributN*}) → Chaîne

DOM ECRIRE ATTRIBUT XML(refElément; nomAttribut;*valeurAttribut*{;...; nomAttributN; *valeurAttributN*}})

Il est désormais possible de passer dans le paramètre *valeurAttribut* une valeur d'un type autre que texte : vous pouvez passez des booléens, entiers, réels, heures ou dates. 4D se charge de la conversion en texte, en fonction des principes suivants :

Type	Exemple de valeur convertie
Booléen	"true" ou "false" (non traduit)
Entier	"123456"
Réel	"12.34" (le séparateur décimal est toujours ".")
Heure	"5233" (nombre de secondes)
Date	"2006-12-04T00:00:00Z" (RFC 3339)

DOM Analyser source XML

DOM Analyser source XML (document{; validation{; dtd | *schéma*}) → Chaîne

Paramètres	Type		Description
document	Chaîne	→	Chemin d'accès du document
validation	Booléen	→	Vrai = *Validation* Faux = *Pas de validation*
dtd | *schéma*	Chaîne	→	Emplacement de la DTD *ou* *Emplacement du schéma XML*
Résultat	Chaîne	←	Référence de l'élément XML

DOM Analyser variable XML

DOM Analyser variable XML (variable{; validation{; dtd | *schéma*}) → Chaîne

Paramètres	Type		Description
variable	BLOB	Texte	→	Nom de la variable
validation	Booléen	→	Vrai = *Validation* Faux = *Pas de validation*
dtd	*schéma*	Chaîne	→	Emplacement de la DTD *ou* *Emplacement du schéma XML*
Résultat	Chaîne	←	Référence de l'élément XML

Les commandes DOM Analyser source XML et DOM Analyser variable XML autorisent désormais la validation par schéma XML, c'est-à-dire via un document XSD (*XML Schema Definition*). Pour cela, il suffit de passer dans le troisième paramètre un fichier ou un URL d'extension "xsd" au lieu de "dtd".

La validation par schéma XML est considérée comme plus souple et plus puissante que la validation par DTD. Le langage des documents XSD est basé sur le langage XML. Les schémas XML prennent notamment en charge des types de données. Pour plus d'informations sur les schémas XML, reportez-vous à l'adresse http://www.w3.org/XML/Schema.

Si vous passez Vrai dans le paramètre *validation* et omettez le troisième paramètre, la commande tentera de valider la structure XML via une référence DTD ou XSD trouvée dans la structure elle-même. La validation peut être indirecte : si la structure contient une référence vers une fichier DTD qui lui-même contient une référence vers un fichier XSD, la commande tentera d'effectuer les deux validations.

SAX AJOUTER VALEUR ELEMENT XML

SAX AJOUTER VALEUR ELEMENT XML(document; données*{; *}*)

Paramètres	Type		Description
document	DocRef	→	Référence du document ouvert
données	Texte	Var	→	Texte ou variable à insére dans le document
*	*	→	• *Si passé : encodage des caractères spéciaux* • *Si omis : pas d'encodage*

La commande SAX AJOUTER VALEUR ELEMENT XML peut désormais encoder automatiquement les caractères spéciaux (< > " '...) contenus dans le paramètre *données*. Pour cela, il suffit de passer le paramètre facultatif *****.

APPLIQUER TRANSFORMATION XSLT

APPLIQUER TRANSFORMATION XSLT(sourceXML; feuilleXSL; résultat*{; compileFeuille}*)

Paramètres	Type		Description
sourceXML	Chaîne | BLOB	→	Nom ou chemin d'accès du document XML source, ou BLOB contenant le XML source
feuilleXSL	Chaîne | BLOB	→	Nom ou chemin d'accès du document contenant la feuille de style XSL, ou BLOB contenant la feuille de style XSL
résultat	Chaîne | BLOB	→	Nom ou chemin d'accès du document recevant le résultat de la transformation XSLT, ou BLOB recevant le résultat de la transformation XSLT
compileFeuille	*Booléen*	→	*Vrai : optimise la transformation XSLT* *Faux ou omis : pas d'optimisation, efface le fichier XSL compilé s'il existe*

La commande APPLIQUER TRANSFORMATION XSLT admet un nouveau paramètre facultatif, *compileFeuille*, permettant d'optimiser la transformation XSLT, notamment en cas d'applications successives de la même feuille XSL.

Lorsque le paramètre *compileFeuille* est passé et vaut Vrai, le fichier XSL *feuilleXSL* est analysé au premier appel de la commande puis est compilé et stocké en mémoire. A chaque appel suivant avec le même fichier XSL, la commande utilise directement le fichier compilé (sauf s'il a été modifié), ce qui permet d'accélérer les traitements.

L'optimisation ne prend pas en compte les éventuelles modifications effectuées dans les fichiers importés (via xsl:import). Si un fichier référencé par le fichier XSL est modifié, il est nécessaire de "forcer" la recompilation du nouveau fichier XSL en rappelant la commande avec le paramètre *compileFeuille* à Faux (ou omis).

Web Services (Client)

La commande APPELER WEB SERVICE a été optimisée en version 11.

APPELER WEB SERVICE

APPELER WEB SERVICE (urlAccès; soapAction; nomMéthode; espaceNommage{; typeComposé}{; *})

Paramètres	Type		Description
urlAccès	Chaîne	→	URL d'accès au Web Service
soapAction	Chaîne	→	Contenu du champ SOAPAction
nomMéthode	Chaîne	→	Nom de la méthode
espaceNommage	Chaîne	→	Espace de nommage (Namespace)
typeComposé	Entier long	→	Configuration de types composés (types simples si omis)
*		→	*Si * passé, ne pas fermer la connexion*

La commande APPELER WEB SERVICE accepte désormais une étoile (*) en dernier paramètre.

Lorsque ce paramètre est passé, la commande ne referme pas la connexion utilisée par le process à l'issue de son exécution. Dans ce cas, l'appel suivant à APPELER WEB SERVICE réutilise cette même connexion si le paramètre * est passé, et ainsi de suite. Pour refermer la connexion, il suffit d'exécuter la commande APPELER WEB SERVICE sans le paramètre *.

Ce mécanisme permet d'accélérer sensiblement les traitements en cas d'appels successifs de plusieurs Web Services sur le même serveur, notamment en configuration *WAN* (via Internet par exemple).

A noter que ce mécanisme s'appuie sur le paramétrage "keep-alive" du serveur Web. Ce paramétrage définit généralement un nombre maximal de requêtes via une même connexion, et peut même les interdire. Si les requêtes APPELER WEB SERVICE enchaînées dans la même connexion atteignent ce nombre maximal ou si les connexions keep-alive ne sont pas autorisées, 4D créera une nouvelle connexion pour chaque requête.

Modifications diverses

Ce paragraphe liste les modifications additionnelles effectuées dans diverses commandes et thèmes de commandes de 4D v11.

Sous-enregistrements

Comme les sous-tables ne sont plus prises en charge dans 4D v11, il n'est plus possible de créer de sous-tables dans cette version.
Dans les bases de données converties, les sous-tables continueront de fonctionner mais elles seront affichées comme des tables standard avec un lien spécial. Pour plus d'informations, reportez-vous au paragraphe "Conversion des sous-tables", page 44.

Dans ce contexte, les commandes du thème "Sous-enregistrements" continueront de fonctionner.

Si le lien spécial est supprimé, la sous-table sera convertie en table standard. Il ne sera plus possible de la rétablir et les commandes du thème "Sous-enregistrements" ne fonctionnent plus avec cette table.

EFFACER VARIABLE

La commande EFFACER VARIABLE fonctionne désormais en mode interprété de la même manière qu'en mode compilé. Cela signifie que la commande réinitialise la variable à la valeur par défaut de son type quel que soit le mode d'exécution.

Dans les versions précédentes, la commande supprimait la variable de la mémoire en mode interprété.

Thème : Variables.

Variable système Document

La variable système *Document* contient désormais dans tous les cas le chemin d'accès complet du document, incluant le disque et l'enchaînement des dossiers.

Dans les versions précédentes de 4D, la variable *Document* pouvait contenir le nom du document seul (par exemple "mondoc.txt") si la commande à l'origine de la mise à jour de la variable passait uniquement ce nom.

Par exemple, soit l'instruction suivante :

```
docRef:=Creer document("mondoc.txt")
```

- Dans les versions précédentes de 4D :
Document = "mondoc.txt"

- A compter de 4D v11 :
Document = "c:\MesDocuments\mondoc.txt" ou
Document = "MacHD:MesDocuments:mondoc.txt

Constantes Format d'affichage

La prise en charge des formats d'affichage système a été renforcée dans 4D v11 (cf. paragraphe "Formats d'affichage des dates et des heures", page 171 et la nouvelle commande LIRE FORMATAGE SYSTEME, page 365).

Dans ce cadre, les constantes des thèmes "Formats d'affichage des dates" et "Formats d'affichage des heures" ont été modifiées et de nouvelles constantes ont été ajoutées. Les tableaux suivants détaillent ces modifications :

Formats d'affichage des dates

Pour plus de clarté, les constantes de formats d'affichage des dates ont été renommées. Les préfixes "Système" et "Interne" indiquent si le format est basé sur un format système ou un format interne à l'application :

Constante 4D v11	Valeur	Nom versions précédentes
Système date court	1	*Format court*
Système date abrégé	2	*Format abrégé*
Système date long	3	*Format long*
Interne date court spécial	4	*Format spécial*
Interne date long	5	*Jour Mois Année*
Interne date abrégé	6	*Abrégé Jour Mois Année*
Interne date court	7	*Spécial forcé*
ISO Date	8	*ISO Date heure*
Vide si date nulle	100	-

Pour plus d'informations sur ces formats, reportez-vous au paragraphe "Formats de date", page 171.

Formats d'affichage des heures

De nouvelles constantes de formats d'affichage des heures sont disponibles et certaines constantes existantes ont été renommées :

Constante 4D v11	Valeur	Nom versions précédentes
h mn s	1	*idem*
h mn	2	*idem*
Heures Minutes Secondes	3	*Heure Minute Seconde*
Heures Minutes	4	*Heure Minute*

Constante 4D v11	Valeur	Nom versions précédentes
h mn Matin Après midi	5	*idem*
mn s	6	-
Minutes secondes	7	-
ISO Heure	8	-
Système heure court	9	-
Système heure long abrégé	10	-
Système heure long	11	-
Vide si heure nulle	100	-

Pour plus d'informations sur ces formats, reportez-vous au paragraphe "Formats d'heure", page 172.

CHOIX FORMATAGE

Deux nouveautés concernent la commande CHOIX FORMATAGE dans 4D v11 :

- CHOIX FORMATAGE permet d'activer dynamiquement le mode "Barber shop" pour les thermomètres. Pour cela, il suffit de passer la valeur 128 dans le paramètre *mode* des jauges (thermomètres et règles). Les autres paramètres (*min*, *max*...) sont ignorés. La valeur 128 ne peut pas être cumulée avec d'autres modes. Pous plus d'informations sur les thermomètres "Barber shop", reportez-vous au paragraphe "Thermomètre "Barber shop"", page 170.

- Il est possible de référencer l'image associée à un bouton 3D via un URL correspondant à une image située dans le dossier **Resources**. Le principe de fonctionnement de ce dossier est décrit dans le paragraphe "Référencement automatique des images", page 49.
  Lorsque l'image est placée dans le dossier **Resources** de la base, vous pouvez passer un URL du type "#{dossier/}monimage" (ou "file:{dossier/}monimage") dans les paramètres *image* ou *imageFond*.

▼ Passage d'un thermomètre en mode "barber shop" :

```
CHOIX FORMATAGE($monthermo;";;;;128")
$monthermo:=1 ` Déclencher l'animation
```

▼ Modification de l'image d'un bouton 3D ; l'image est située dans le sous-dossier "FR" du dossier "Resources" de la base :

```
CHOIX FORMATAGE($monbouton;"Ouvrir;#FR/open.jpg")
```

Thème : Propriétés des objets.

ALLER A CHAMP

ALLER A CHAMP ({*; }objet)

Cette commande peut être utilisée pour supprimer tout focus dans le formulaire courant. Pour cela, il suffit de passer un nom d'objet vide :

ALLER A CHAMP(*;"") ` Plus aucun objet n'a le focus

Thème : Gestion de la saisie.

IMPORTER TEXTE, EXPORTER TEXTE

IMPORTER TEXTE({table; }document)
EXPORTER TEXTE({table; }document)

Note Ces commandes étaient nommées respectivement LECTURE ASCII et ECRITURE ASCII dans les versions précédentes de 4D.

Par défaut en mode Unicode, ces commandes utilisent désormais le jeu de caractères UTF-8. Utilisez la commande UTILISER FILTRE pour modifier ce jeu de caractères.

Note Lorsque l'application 4D est exécutée en mode non Unicode (mode compatibilité ASCII), ces commandes utilisent la table ASCII comme dans les versions précédentes.

Thème : Import-Export.

INFORMATIONS PROCESS

De nouvelles valeurs de type de process peuvent être retournées dans le paramètre *origine*. Ces valeurs sont fournies via les constantes suivantes (thème "Type du process") :

Constante	Type	Valeur
Process du serveur Web	Entier long	-13
Process exécuté sur client	Entier long	-14
Process 4D Server	Entier long	-15
Process sur fermeture	Entier long	-16
Process macro éditeur de méthod	Entier long	-17
Process 4D Server interne	Entier long	-18
Process de sauvegarde	Entier long	-19
Process du fichier d'historique	Entier long	-20
Process de restitution	Entier long	-21
Process CSM	Entier long	-22
Process minuteur	Entier long	-23
Process exécution méthode SQL	Entier long	-24

Process Server Controller	Entier long	-25
Process d'activité	Entier long	-26

A noter également qu'en conformité avec la terminologie de 4D v11, deux constantes existantes ont été renommées :

Nouveau nom	Ancien nom	Valeur
Process développement	Process de structure	-2
Créé par dialogue d'exécution	Créé depuis le mode Utilisation	3

Thème : Process.

Tableaux à deux dimensions

- Si vous passez un tableau 2D comme paramètre à une commande qui attend une variable, 4D v11 génèrera une erreur.
  Dans les versions précédentes de 4D, aucune erreur n'était générée mais la deuxième dimension n'était pas prise en compte.
  Le comportement précédent était :
  monType:=**Type**(monTab2D{1}{1}) était interprété par 4D :
  monType:=**Type**(monTab2D{1})

- Lorsque la fonction **Type** est appliquée à une "ligne" d'un tableau 2D, elle retourne désormais le véritable type du tableau 2D et non plus le type "Est un tableau 2D".
  Soit par exemple le code suivant :

```
TABLEAU TEXTE($monTab2D;50;20)
$letype:=Type($monTab2D{10})
```

  Dans 4D v11, *$letype* vaut 18 (Est un tableau texte). Dans les versions précédentes de 4D, *$letype* valait 13 (Est un tableau 2D).

  Astuce : Dans certains cas, par exemple pour écrire du code générique, il peut être nécessaire de savoir si un tableau est tableau standard indépendant ou une "ligne" d'un tableau 2D. Dans ce cas, il suffit d'utiliser le code suivant :

```
ptrmonTab:=->monTab{6}   ` monTab{6} est-il une ligne d'un tableau 2D ?
RESOUDRE POINTEUR(ptrmonTab;nomVar;numTable;numChamp)
Si(nomVar#"")
   $ptr:=Pointeur vers(nomVar)
   $letype:=Type($ptr->)
      ` Si monTab{6} est une ligne de tableau 2D, $letype vaut 13
Fin de si
```

- Dans les versions précédentes de 4D, il était possible de modifier à la volée la longueur de chaîne d'un tableau alpha 2D en interprété. Par exemple, les instructions suivantes étaient acceptées :
TABLEAU ALPHA(80;tAmontab;10)
...
TABLEAU ALPHA(255;tAmontab{2};10)

  En version 11, cette modification n'est plus permise et génèrera une erreur de syntaxe dès le mode interprété.

TABLEAU VERS SELECTION

Dans les versions précédentes de 4D, il était possible de passer des tableaux de tailles différentes à cette commande (la taille du premier tableau déterminait le nombre d'éléments à copier).
Le contrôle de programmation ayant été renforcé dans 4D v11, il est désormais obligatoire que tous les tableaux passés à cette commande aient le même nombre d'éléments. Dans le cas contraire, une erreur de syntaxe "La taille des tableaux ne correspond pas" est générée.

Thème : Tableaux.

FIXER FICHIER HISTORIQUE

FIXER FICHIER HISTORIQUE(historique | *)

Le principe de fonctionnement de cette commande a été modifié : désormais, il n'est plus possible d'ouvrir de fichier d'historique existant. Seules la création d'un nouvel historique et la fermeture de l'historique courant sont permises par cette commande.

En outre, en cas de création, il est à noter que le nouveau fichier d'historique n'est pas généré immédiatement après l'exécution de la commande, mais après la prochaine sauvegarde (le paramètre est conservé dans le fichier de données, il sera pris en compte même si la base est refermée entre-temps. Vous pouvez appeler la commande SAUVEGARDER pour provoquer la création du fichier d'historique.

Ce principe n'est pas nécessaire en cas de fermeture (le fichier d'historique est immédiatement refermé).

Thème : Sauvegarde.

FORMULAIRE ENTREE, FORMULAIRE SORTIE

Lorsque le formulaire désigné par le paramètre *formulaire* est invalide, ces commandes retournent désormais l'erreur 81 : "Le formulaire n'a pas été trouvé".

Thème : Formulaires.

Changements de noms

Afin d'harmoniser les noms des commandes et de clarifier leur usage, deux commandes du thème "Tableaux" et deux commandes du thème "Listes hiérarchiques" ont été renommées dans 4D v11. La commande LIRE LISTE PLUGINS a également été renommée (et changée de thème, cf. paragraphe suivant). Leur fonctionnement est inchangé :

Ancien nom (4D 2004.x)	Nouveau nom (4D v11.x)
INSERER LIGNES	**INSERER DANS TABLEAU**
SUPPRIMER LIGNES	**SUPPRIMER DANS TABLEAU**
INSERER ELEMENT	**INSERER DANS LISTE**
SUPPRIMER ELEMENT	**SUPPRIMER DANS LISTE**
LIRE LISTE PLUGINS	**LISTE PLUGINS**

Changements de thèmes

Pour plus de clarté, les modifications suivantes ont été effectuées dans les thèmes de commandes existants (ces changements n'influent pas sur le fonctionnement des commandes) :

Commande	Ancien thème	Nouveau thème 4D v11
AFFICHER BARRE OUTILS CACHER BARRE OUTILS	Barres d'outils (*thème supprimé*)	Interface utilisateur
AFFICHER BARRE MENUS CACHER BARRE MENUS APPELER SUR A PROPOS	Menus	Interface utilisateur
FIXER PARAMETRE BASE, Lire parametre base	Définition structure	Environnement 4D
GENERER APPLICATION	Outils	Environnement 4D
LANCER PROCESS EXTERNE FIXER VARIABLE ENVIRONNEMENT	Environnement système	Outils
Licence disponible	Environnement 4D	Utilisateurs et groupes
(LIRE) LISTE PLUGINS	Utilisateurs et groupes	Environnement 4D
EXECUTER FORMULE	Langage	Formules

Commandes obsolètes

Les commandes obsolètes suivantes ne sont plus prises en charge dans 4D v11 ; elles ont été supprimées du programme. Elles ne sont plus reconnues par l'éditeur de méthodes.

- **CHERCHER SUR CLE**
- **STOCKER ANCIEN**
  Cette commande est remplacée par la commande STOCKER SUR LIEN.

- **TRIER SUR INDEX**
  Cette commande est remplacée par la commande TRIER.

  La commande suivante devient obsolète dans 4D v11. Elle est toutefois conservée par compatibilité. Dans la documentation de 4D, elle est transférée dans le thème “Commandes obsolètes” :

- **Lire ID ressource composant** (thème “Ressources”) : cette commande fonctionnait avec les composants d’ancienne génération, désormais incompatibles avec 4D v11 (cf. paragraphe “Nouvelle architecture des composants”, page 50). Elle est désormais sans effet.

A Codes de langue

Le tableau suivant liste les codes de langues pris en charge par 4D v11. Pour plus d'informations, reportez-vous au paragraphe "Prise en charge du standard XLIFF", page 101.

Langues	ISO639-1	"Legacy"	ISO3166
AFRIKAANS	af	afrikaans	
ALBANIAN	sq	albanian	
ARABIC_SAUDI_ARABIA	ar	arabic	sa
ARABIC_IRAQ	ar	arabic	iq
ARABIC_EGYPT	ar	arabic	eg
ARABIC_LIBYA	ar	arabic	ly
ARABIC_ALGERIA	ar	arabic	dz
ARABIC_MOROCCO	ar	arabic	ma
ARABIC_TUNISIA	ar	arabic	tn
ARABIC_OMAN	ar	arabic	om
ARABIC_YEMEN	ar	arabic	ye
ARABIC_SYRIA	ar	arabic	sy
ARABIC_JORDAN	ar	arabic	jo
ARABIC_LEBANON	ar	arabic	lb
ARABIC_KUWAIT	ar	arabic	kw
ARABIC_UAE	ar	arabic	ae
ARABIC_BAHRAIN	ar	arabic	bh
ARABIC_QATAR	ar	arabic	qa
BASQUE	eu	basque	
BELARUSIAN	be	belarusian	
BULGARIAN	bg	bulgarian	
CATALAN	ca	catalan	

CHINESE_TRADITIONAL	zh	chinese	cht
CHINESE_SIMPLIFIED	zh	chinese	chs
CHINESE_HONGKONG	zh	chinese	hk
CHINESE_SINGAPORE	zh	chinese	sg
CROATIAN	hr	croatian	
CZECH	cs	czech	
DANISH	da	danish	
DUTCH	nl	dutch	
DUTCH_BELGIAN	nl	dutch	be
ENGLISH_US	en	english	us
ENGLISH_UK	en	english	gb
ENGLISH_AUSTRALIA	en	english	au
ENGLISH_CANADA	en	english	ca
ENGLISH_NEWZEALAND	en	english	nz
ENGLISH_EIRE	en	english	ie
ENGLISH_SOUTH_AFRICA	en	english	za
ENGLISH_JAMAICA	en	english	jm
ENGLISH_CARIBBEAN	en	english	cb
ENGLISH_BELIZE	en	english	bz
ENGLISH_TRINIDAD	en	english	tt
ESTONIAN	et	estonian	
FAEROESE	fo	faorese	
FARSI	fa	persian	
FINNISH	fi	finnish	
FRENCH	fr	french	
FRENCH_BELGIAN	fr	french	be
FRENCH_CANADIAN	fr	french	ca
FRENCH_SWISS	fr	french	ch
FRENCH_LUXEMBOURG	fr	french	lu
GERMAN	de	german	
GERMAN_SWISS	de	german	ch
GERMAN_AUSTRIAN	de	german	at
GERMAN_LUXEMBOURG	de	german	lu
GERMAN_LIECHTENSTEIN	de	german	li
GREEK	el	greek	
HEBREW	he	hebrew	

HUNGARIAN	hu	hungarian	
ICELANDIC	is	iceland	
INDONESIAN	id	indonesian	
ITALIAN	it	italian	
ITALIAN_SWISS	it	italian	ch
JAPANESE	ja	japanese	
KOREAN_WANSUNG	ko	korean	
KOREAN_JOHAB	ko	korean	
LATVIAN	lv	latvian	
LITHUANIAN	lt	lithuanian	
NORWEGIAN	no	norwegian	
NORWEGIAN_NYNORSK	nn	nynorsk	no
POLISH	pl	polish	
PORTUGUESE	pt	portuguese	
PORTUGUESE_BRAZILIAN	pt	portuguese	br
ROMANIAN	ro	romanian	
RUSSIAN	ru	russian	
SERBIAN_LATIN	sr	serbian	latn
SERBIAN_CYRILLIC	sr	serbian	cyrl
SLOVAK	sk	slovak	
SLOVENIAN	sl	slovenian	
SPANISH_CASTILLAN	es	spanish	
SPANISH_MEXICAN	es	spanish	mx
SPANISH_MODERN	es	spanish	
SPANISH_GUATEMALA	es	spanish	gt
SPANISH_COSTA_RICA	es	spanish	cr
SPANISH_PANAMA	es	spanish	pa
SPANISH_DOMINICAN_REPUBLIC	es	spanish	do
SPANISH_VENEZUELA	es	spanish	ve
SPANISH_COLOMBIA	es	spanish	co
SPANISH_PERU	es	spanish	pe
SPANISH_ARGENTINA	es	spanish	ar
SPANISH_ECUADOR	es	spanish	ec
SPANISH_CHILE	es	spanish	cl
SPANISH_URUGUAY	es	spanish	uy
SPANISH_PARAGUAY	es	spanish	py

SPANISH_BOLIVIA	es	spanish	bo
SPANISH_EL_SALVADOR	es	spanish	sv
SPANISH_HONDURAS	es	spanish	hn
SPANISH_NICARAGUA	es	spanish	ni
SPANISH_PUERTO_RICO	es	spanish	pr
SWEDISH	sv	swedish	
SWEDISH_FINLAND	sv	swedish	fi
THAI	th	thai	
TURKISH	tr	turkish	
UKRAINIAN	uk	ukrainian	
VIETNAMESE	vi	vietnamese	

B Equivalences des types de données SQL

Les tableaux suivants précisent les équivalences entre les types de données du moteur SQL de 4D, les types de données du langage standard de 4D et ceux des bases de données SQL les plus répandues :

- 4D SQL et ORACLE
- 4D SQL et MySQL
- 4D SQL et PostgreSQL
- 4D SQL et Access
- 4D SQL et MS SQL Server
- 4D SQL et Sybase
- 4D SQL et IBM DB2.

4D SQL et ORACLE

4D SQL	Langage 4D v11	ORACLE
VARCHAR(n)	Texte	VARCHAR2(n [BYTE | CHAR])
		CLOB
	-	CHAR
	-	NCHAR
	-	NVARCHAR2
REAL	Réel (Numérique)	NUMBER(p,s)
NUMERIC		
FLOAT	Réel (Numérique)	FLOAT[(p)]
SMALLINT	Entier	NUMBER(p,0)
INT	Entier long	
	-	LONG
BIT	Booléen	-
BOOLEAN		-
BLOB	BLOB	BLOB
	-	RAW(n)
	-	LONG RAW
BIT VARYING	BLOB	BLOB
CLOB	Texte	CLOB
	-	NCLOB
TEXT	Texte	VARCHAR2
		CLOB
TIMESTAMP	Date et Heure traitées séparément (conversion automatique)	TIMESTAMP
	-	DATE
	-	INTERVAL YEAR TO MONTH
DURATION	Heure	INTERVAL DAY TO SECOND
INTERVAL		
PICTURE	Image	BLOB
-	-	BFILE
-	-	ROWID
-	-	UROWID[(n)]

4D SQL et MySQL

4D SQL	4D v11	MySQL
VARCHAR(n)	Texte	LONGTEXT
REAL	Réel (Numérique)	REAL[(p,s)] DOUBLE[(p,s)] DOUBLE PRECISION[(p,s)]
NUMERIC	Réel (Numérique)	NUMERIC[(p[,s])] DECIMAL[(p[,s])] DEC[(p[,s])]
	-	BIGINT [UNSIGNED]
FLOAT	Réel (Numérique)	FLOAT[(p[,s])]
	-	DECIMAL[(p[,s])] [UNSIGNED]
SMALLINT	Entier	SMALLINT
	-	TINYINT [UNSIGNED] (BYTE ??)
INT	Entier long	INT INTEGER
	-	MEDIUMINT [UNSIGNED]
BIT	Booléen	BOOL TINYINT BIT
BOOLEAN		
BLOB	BLOB	BLOB
BIT VARYING		
BLOB	-	TINYBLOB MEDIUMBLOB LONGBLOB
CLOB	Texte	LONGTEXT
TEXT		
CLOB	-	TINYTEXT MEDIUMTEXT LONGTEXT
TIMESTAMP	Date et Heure traitées séparément (conversion automatique)	TIMESTAMP[(p)]
		DATE
DURATION	Heure	TIME
INTERVAL		
PICTURE	Image	BLOB
-	-	[NATIONAL]CHAR[(n)] [BINARY]

4D SQL	4D v11	MySQL
-	-	YEAR[(2|4)]
-	-	ENUM('value1','value2',...)
-	-	SET('value1','value2',...)

4D SQL et PostgreSQL

4D SQL	4D v11	PostgreSQL
VARCHAR(n)	Texte	varchar(n) character varying(n)
	-	character(n) char(n)
REAL	Réel (Numérique)	real float4
NUMERIC	Réel (Numérique)	numeric(p,s) decimal(p,s)
	-	bigint int8
FLOAT	Réel (Numérique)	Float8
	-	double precision float8 float
SMALLINT	Entier	smallint Int2
INT	Entier long	int integer int4
BIT	Booléen	boolean bool
BOOLEAN	Booléen	boolean bool
BLOB	BLOB	bytea
	-	bit(n)
	-	bit varying (n)
	-	varbit(n)
BIT VARYING	BLOB	bytea
CLOB	Texte	text
TEXT	Texte	text
TIMESTAMP	Date et Heure traitées séparément (conversion automatique)	timestamp [(p)] [without time zone]
	-	timestamp [(p)] with timezone
	-	date
	-	time [(p)] [without time zone]

4D SQL	4D v11	PostgreSQL
DURATION	Heure	time
	-	time [(p)] with timezone
INTERVAL	Heure	interval(p)
PICTURE	Image	bytea
-	-	money
-	-	serial
-	-	bigserial
-	-	box
-	-	line
-	-	lseg
-	-	circle
-	-	path
-	-	point
-	-	polygon
-	-	cidr
-	-	inet
-	-	macadr
-	-	oid
-	-	xid

4D SQL et Access

4D SQL	4D v11	Access
VARCHAR(n)	Texte	Text
REAL	Réel (Numérique)	Number (Double)
NUMERIC		
FLOAT		
SMALLINT	Entier	Number (Integer)
INT	Entier long	Number (Long Integer)
BIT	Booléen	Number (byte)
BOOLEAN		
BLOB	BLOB	-
BIT VARYING		
CLOB	Texte	Memo
TEXT		
TIMESTAMP	Date et Heure traitées séparément (conversion automatique)	Time
DURATION	Heure	Time
INTERVAL		
PICTURE	Image	-
-	-	Currency
-	-	AutoNumber
-	-	OLE Object
-	-	Hyperlink
-	-	Lookup Wizard

4D SQL et MS SQL Server

4D SQL	4D v11	MS SQL Server
VARCHAR(n)	Texte	nvarchar[(n)]
	-	char[(n)]
	-	nchar[(n)]
	-	ntext
REAL	Réel (Numérique)	real

4D SQL	4D v11	MS SQL Server
NUMERIC	Réel (Numérique)	numeric[(p[,0])]
	-	bigint
FLOAT	Réel (Numérique)	float[(n)]
	-	decimal[(p[,s])]
SMALLINT	Entier	smallint
	-	tinyint
INT	Entier long	int
BIT	Booléen	bit
BOOLEAN		
BLOB	BLOB	Varbinary
	-	binary[(n)]
	-	varbinary[(n)]
BIT VARYING	BLOB	Varbinary
CLOB	Texte	ntext
TEXT		
TIMESTAMP	Date et Heure traitées séparément (conversion automatique)	datetime
	-	smalldatetime
DURATION	Heure	datetime
INTERVAL		
PICTURE	Image	image
		money
		smallmoney
		uniqueidentifier

4D SQL et Sybase

4D SQL	4D v11	Sybase
VARCHAR(n)	Texte	varchar[(n)]
	-	char[(n)]
	-	nchar[(n)]
	-	nvarchar[(n)]
REAL	Réel (Numérique)	real
NUMERIC	Réel (Numérique)	numeric[(p[,0])]
FLOAT	Réel (Numérique)	float[(precision)]
	-	decimal[(p[,s])]
	-	double precision
SMALLINT	Entier	smallint
	-	tinyint
INT	Entier long	int
BIT	Booléen	bit
BOOLEAN		
BLOB	BLOB	varbinary
	-	binary[(n)]
	-	varbinary[(n)]
BIT VARYING	BLOB	varbinary
CLOB	Texte	text
TEXT		
TIMESTAMP	Date et Heure traitées séparément (conversion automatique)	datetime
	-	smalldatetime
DURATION	Heure	datetime
INTERVAL		
PICTURE	Image	image
-	-	money
-	-	smallmoney

4D SQL et IBM DB2

4D SQL	4D v11	IBM DB2
VARCHAR(n)	Texte	VARCHAR(n) [FOR BIT DATA] CHARACTER VARYING(n) [FOR BIT DATA] CHAR VARYING(n) [FOR BIT DATA]
	-	CHAR[(n)] [FOR BIT DATA] CHARACTER[(n)] [FOR BIT DATA]
REAL	Réel (Numérique)	REAL
NUMERIC	Réel (Numérique)	NUMERIC[(p[,0)] NUM[(p[,s])] DECIMAL[(p[,s])] DEC[(p[,s])]
	-	BIGINT
FLOAT	Réel (Numérique)	FLOAT[(n)]
		DOUBLE [PRECISION]
SMALLINT	Entier	SMALLINT
INT	Entier long	INTEGER INT
BIT	Booléen	-
BOOLEAN		
BLOB	BLOB	BLOB(n[K|M|G])
BIT VARYING		
CLOB	Texte	CLOB(n[K|M|G])
	-	DBCLOB(n[K|M|G])
TEXT	Texte	CLOB(n[K|M|G])
	-	LONG VARCHAR [FOR BIT DATA]
TIMESTAMP	Date et Heure traitées séparément (conversion automatique)	TIMESTAMP
		DATE
DURATION	Heure	TIME
INTERVAL		
PICTURE	Image	BLOB(n[K|M|G])
		GRAPHIC[(n)]
		VARGRAPHIC(n)
		LONG VARGRAPHIC
		DATALINK

C 4D Pack v11

De nombreuses modifications ont été apportées la version 11 du plug-in 4D Pack. Une nouvelle commande a été ajoutée, plusieurs commandes sont désormais déconseillées et d'autres ont été supprimées.

Cette section fait le point sur ces modifications et, dans le cas des commandes non conservées, indique les solutions de remplacement.

Note Les commandes qui n'apparaissent pas dans cette section sont maintenues intégralement dans 4D Pack v11. Leur fonctionnement est inchangé.

Nouveautés

AP Get file MD5 digest

AP Get file MD5 digest(cheminFichier; digest{; fork}) → Entier long

Paramètres	Type		Description
cheminFichier	Texte	→	Chemin complet du fichier
digest	Texte	←	Digest MD5 du fichier
fork	Entier long	→	0=Data fork, 1=Resource fork
Résultat	Entier long	←	Code d'erreur

La nouvelle commande AP Get file MD5 digest retourne la clé digest MD5 d'un document spécifique. L'algorithme MD5 (*Message Digest 5*) est une fonction de hachage utilisée pour crypter les données.

Passez dans le paramètre *cheminFichier* le nom du document contenant la clé. La clé digest MD5 est retournée dans le paramètre *digest*.

Le paramètre fork vous permet de définir, sous Mac OS uniquement, la partie du fichier à sélectionner :

- 0 pour la data fork
- 1 pour la resource fork.

▼ Exemple d'utilisation :

```
C_TEXTE($ledoc)
C_TEXTE(<>digest)
C_ENTIER LONG($resfork)

$resfork:=0
$ledoc:=Selectionner document
$erreur:=AP Get file MD5 digest($ledoc;<>digest;$resfork)
```

Thème : Utilities.

AP Does method exist

AP Does method exist (nomMéthode) → Entier

Paramètres	Type		Description
nomMéthode	Alpha	→	Nom de la méthode à tester
Résultat	Entier	←	0=la méthode n'existe pas 1=la méthode existe déjà

La nouvelle commande AP Does method exist permet de savoir si une méthode projet nommée *nomMéthode* existe déjà dans la base de données courante. Cette commande ne tient pas compte des méthodes projet des composants installés dans la base.

Dans le cadre de l'utilisation de la commande AP Create method, cette commande a pour but d'éviter l'apparition d'un message d'erreur lorsqu'une méthode de même nom existe déjà.

Thème : Utilities.

AP Create method

AP Create method(nomMéthode;tabPropriétés;codeMéthode{;nomDossier}) → Entier long

Le paramètre *tabPropriétés* admet désormais six éléments :

- Passez 1 dans *tabPropriétés*{5} si la méthode doit être partagée entre les composants et la base hôte, et 0 dans le cas contraire.

- Passez 1 dans *tabPropriétés*{6} si la méthode doit être disponible via le SQL, et 0 dans le cas contraire.

Thème : Utilities.

Commandes obsolètes

Les commandes suivantes de 4D Pack v11 peuvent toujours être utilisées mais correspondent à des technologies ou des mécanismes en cours d'abandon. Ces commandes seront supprimées dans les prochaines versions du plug-in, leur remplacement est fortement conseillé.

AP Save BMP 8 bits, AP Get picture type	Ces commandes peuvent généralement être remplacées par les commandes du thème "Images" de 4D.
AP GET PARAM AP SET PARAM	La plupart des paramètres accessibles via ces commandes sont désormais obsolètes ou peuvent être gérés par les commandes FIXER PARAMETRE BASE, Lire parametre base.
AP AVAILABLE MEMORY	A noter que cette commande retourne désormais des valeurs exprimées en kilo-octets (et non plus en octets).
AP ShellExecute	Cette commande peut généralement être avantageusement remplacée par la commande LANCER PROCESS EXTERNE. Cette commande a été supprimée sous Mac OS, elle reste utilisable sous Windows uniquement.
AP FCLOSE, AP fopen, AP FPRINT, AP fread	Les commandes du thème "ANSI Streams" peuvent généralement être remplacées par les commandes REGLER SERIE, ENVOYER PAQUET, RECEVOIR PAQUET, etc.
AP HELP ON KEY, AP HELP INDEX, AP HELP ON HELP, AP CLOSE HELP	Les commandes du thème "Windows Help Files" fonctionnent sous Windows 2000 et Windows XP mais ne sont pas compatibles avec Windows Vista.

Commandes supprimées

Les commandes suivantes ont été supprimées du plug-in, elles ne seront pas reconnues par l'interpréteur. Il est nécessaire de les remplacer pour pouvoir compiler l'application.

Le tableau suivant liste les commandes supprimées et décrit les solutions de remplacement préconisées par 4D :

Commandes supprimées	Solutions de remplacement
AP PICT DRAGGER	Utiliser les fonctions de glisser-déposer intégrées de 4D (variables système *MouseDown*, *MouseX* et *MouseY* dans le cadre des événements de type "clic")
AP PICT UPDATER et %AP Pict displayer	Utiliser un champ image
AP Read Picture BLOB	BLOB VERS IMAGE
AP Read Picture File	LIRE FICHIER IMAGE
AP Save GIF	CONVERTIR IMAGE + ECRIRE FICHIER IMAGE
AP SET PICT MODE	CHOIX FORMATAGE
AP Select document	Selectionner document
AP Set palette	Selectionner couleur RVB
AP Sublaunch	LANCER PROCESS EXTERNE
AP PrintDefault	AP BLOB to print settings + AP Print settings to BLOB
AP Text to PrintRec	AP BLOB to print settings + AP Print settings to BLOB
AP PrintRec to text	AP BLOB to print settings + AP Print settings to BLOB
AP PrValidate	AP BLOB to print settings + AP Print settings to BLOB
AP Toolbar installed	Hauteur barre outils (retourne 0 si la barre est masquée)
AP SET WEB FILTERS	FIXER PARAMETRE BASE, Lire parametre base
AP Add table and fields	Utiliser le moteur SQL de 4D
AP Create relation	*Pas de solution alternative actuellement*

Index

Symboles/Chiffres

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z