4D v12

Mise à jour
Windows®/Mac OS®

4D v12 - Mise à jour
Versions Windows® and Mac OS®

Sommaire

1 Bienvenue

Bienvenue dans 4D v12. Résolument ouverte vers les technologies les plus répandues et remplie de nouveautés destinées à augmenter la productivité et la créativité des développeurs, cette version constitue une nouvelle étape majeure dans l'évolution de la gamme 4D.

De nombreuses fonctions très demandées par les développeurs de la communauté 4D ont été intégrées, tant au niveau de l'architecture, de l'atelier de développement que du langage : amélioration de l'éditeur de langage, possibilité d'exécuter des scripts PHP, gestion de numéros UUID, amélioration des commandes de gestion des propriétés d'objets et prise en charge étendue du XML et du SVG.

La nouvelle possibilité d'installer des composants dans l'application 4D permet d'en bénéficier automatiquement dans toutes les bases.

Les fonctions d'impression ont été enrichies avec la nouvelle commande IMPRIMER OBJET, la possibilité d'imprimer les list box et la prise en charge étendue des impressions PDF.

De nouveaux objets de formulaires et *widgets* accélèrent la mise en place d'interfaces modernes et sophistiquées : steppers, textes avec styles, sélecteurs de date (*datepicker*) ou d'heure, zone de recherche textuelle ou encore listbox hiérarchiques font partie de ces nouveautés. Les sous-formulaires bénéficient de nombreuses améliorations afin notamment de les rendre plus facilement utilisables en tant que composants. La nouvelle bibliothèque d'objets préconfigurés permet un accès rapide à ces fonctionnalités.

Enfin, le langage SQL de 4D propose des nouvelles fonctions particulièrement puissantes, comme la réplication ou la synchronisation des données et la possibilité d'ouvrir et de refermer différentes bases 4D au cours de la même session.

Toutes ces nouveautés sont détaillées dans les chapitres suivants :

- Architecture et base de données
- Atelier de développement
- Formulaires et objets
- Langage
- Moteur SQL
- Administration de 4D Server

4D Server La version 64 bits de 4D Server v12 est en cours de finalisation. Une version *beta* est dès à présent disponible en téléchargement sur notre site Web (http://www.4d.com/).

Conversion des anciennes bases

Les bases de données créées avec des versions 6.x, 2003.x, 2004.x et 11.x de 4e Dimension, 4D ou 4D Server sont compatibles avec 4D version 12 (fichier de structure et fichier de données).

- Les fichiers des bases de données en version 6.x, 2003.x, 2004.x doivent être convertis par l'intermédiaire d'un assistant et ne pourront plus être ouverts avec leur version d'origine.
- Les fichiers des bases de données en version 11 sont convertis directement en version 12. Une fois converti, le fichier de structure ne pourra plus être ouvert en version 11 ; le fichier de données pourra être rouvert en version 11 sous certaines conditions (cf. paragraphe "Bases en version 11", page 16).

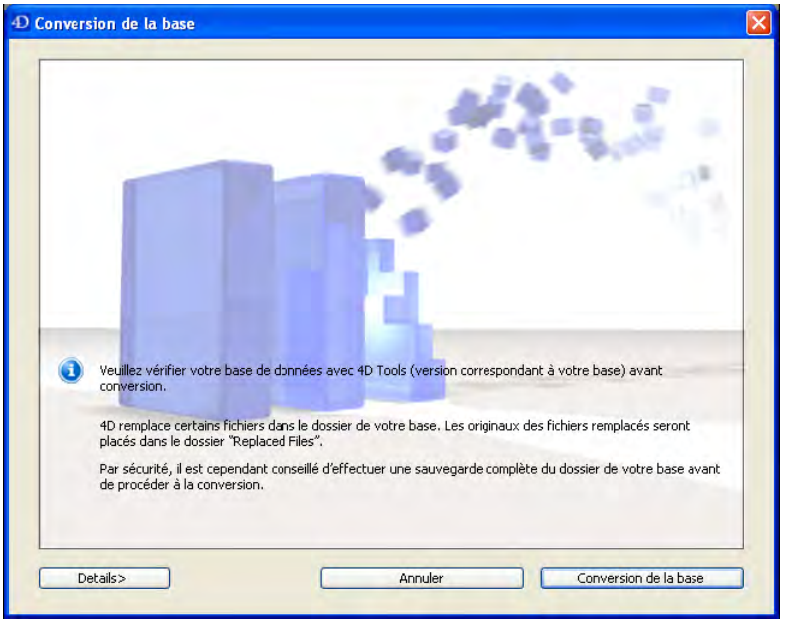
Note: Vous pouvez convertir tout fichier de structure interprété. Le fichier peut contenir le code compilé ; dans ce cas, il sera nécessaire de recompiler la base après conversion.

Bases en version 6.x, 2003.x et 2004.x

Les modifications structurelles apportées au niveau du moteur de la base de données de 4D nécessitent une conversion en profondeur de la structure et des données des bases antérieures aux versions 11 et 12. Cette conversion s'effectue via un assistant spécifique. 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.

Il est conseillé de vérifier avant la conversion l'intégrité de la base à l'aide de l'utilitaire 4D Tools (compactage, vérification et, éventuellement, réparation de la base). Utilisez la version de 4D Tools 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 v12. L'assistant de conversion apparaît automatiquement :

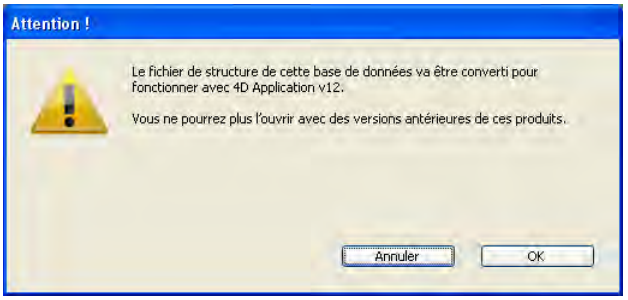


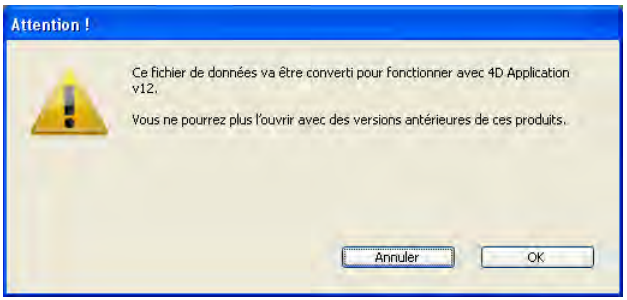
Cliquez sur le bouton **Conversion de la base** pour débuter le processus standard de conversion des fichiers de structure et de données.

- Pour plus d'informations sur cette boîte de dialogue et sur le processus de conversion, reportez-vous au paragraphe « Conversion des anciennes bases de données » dans le manuel *Mode Développement* de 4D.
- Certains mécanismes obsolètes ou anciens ne sont plus pris en charge et seront supprimés ou remplacés au cours de la conversion. Pour plus d'informations, reportez-vous au manuel *Mise à jour 4D v11 SQL*. Pour une description détaillée de toutes les modifications, reportez-vous au document *Guide de migration 4D v11 SQL* disponible en téléchargement à partir de l'adresse : http://doc.4d.com/home.fr.html

Bases en version 11

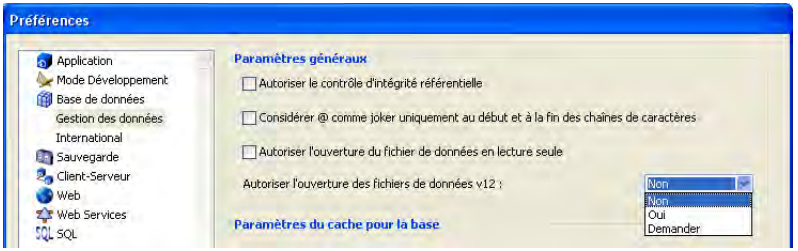
La conversion d'une base en version 11 s'effectue directement lors de l'ouverture du fichier de structure avec 4D v12. Des boîtes de dialogue d'alerte successives vous indiquent que les fichiers vont être convertis et qu'il ne sera plus possible de les ouvrir avec une version précédente :





Il reste possible de rouvrir en version 11 le fichier de données converti. Cette possibilité doit être autorisée explicitement côté version 11 à l'aide de la préférence **Autoriser l'ouverture des fichiers de données v12** située dans la page "Base de données/Gestion des données" des Préférences (à compter de 4D v11 SQL r6 uniquement) :

Préférences de 4D v11 SQL r6
Option d'ouverture d'un fichier de données v12



Cette option doit être utilisée avec précaution dans le cas où des attributs spécifiques de la version 12 ont été appliqués aux tables de la base (risque d'altération des données).

Mise à jour du fichier Macros.xml

Le fichier "Macros.xml" permet de bénéficier de macros-commandes dans l'éditeur de méthodes (cf. manuel *Mode Développement*). Il est stocké dans le dossier de préférences utilisateur :

- **Windows XP** :
  C:\Documents and Settings*NomUtilisateur*\
  Application Data\4D\Macros v2\
- **Windows Vista / Windows 7 :**
  C:\Users*NomUtilisateur*\AppData\Roaming\4D\Macros v2\
- **Macintosh** :
  {disque}:Users:*NomUtilisateur*:Library:Preferences:4D:Macros v2:

Dans la version 12 de 4D, de nouvelles macro-commandes sont disponibles afin de faciliter l'utilisation des nouvelles commandes SQL (cf. Chapitre 6, "SQL"). Comme le fichier "Macros.xml" peut être personnalisé, l'installation de la nouvelle version de 4D n'écrase pas automatiquement la version existante de ce fichier. Pour pouvoir bénéficier des nouvelles macro-commandes SQL de 4D v12, vous devez soit :

- supprimer le fichier "Macros.xml" dans le dossier "Macros v2" (si vous ne l'avez jamais modifié) puis lancer 4D. Le nouveau fichier sera automatiquement recréé.
- ajouter manuellement les nouvelles macros dans le fichier "Macros.xml" du dossier "Macros v2" (si vous avez déjà personnalisé son contenu). Le nouveau fichier modèle de macros est disponible dans le dossier de l'application 4D, à l'emplacement suivant : 4D\Resources\en.lproj ou 4D\Resources\fr.lproj.

Arrêt programmé de fonctions obsolètes

Certaines fonctions anciennes, maintenues dans 4D v12 pour des raisons de compatibilité, seront définitivement abandonnées dans la prochaine version majeure de 4D. Dans une perspective d'évolution de vos applications, il est recommandé de ne plus les utiliser et d'adapter dès à présent vos développements.

Affichage des apparences de plate-forme nommées

Jusqu'à la version 2003 de 4D, il était possible d'affecter une apparence de plate-forme spécifique à la base, aux formulaires ou aux objets. Les différentes options étaient : **Mac OS 7**, **Windows 3.11/NT 3.51**, **Windows 95/95/2000/NT4**, **Mac OS9** et **Thème Mac**.

Ce principe étant obsolète, 4D v12 est la dernière version qui permet d'afficher ces apparences nommées dans les bases de données converties. Dans les prochaines versions, 4D attribuera automatiquement l'apparence **Système** à tous les objets disposant de ce type d'attribut.

Mode contextuel du serveur Web

4D v12 est la dernière version prenant en charge le serveur Web en mode contextuel. Ce mode spécifique et les mécanismes qui lui sont associés sont conservés par compatibilité uniquement et ne fonctionneront plus dans la prochaine version majeure du programme.

En conséquence, dans 4D v12, les options liées à la configuration du mode contextuel (**Mode contextuel au démarrage** et **Utiliser Javascript pour les contrôles de saisie**) ont été supprimées des pages "Web" de la boîte de dialogue des Propriétés de la base. Elles resteront accessibles dans la page "Compatibilité", pour les bases converties uniquement.

Configuration minimale

Les applications de la gamme 4D version 12 requièrent au minimum les configurations suivantes :

	Windows	Mac OS
Processeur	Processeur Pentium IV	Processeur Intel®
Système	Windows Vista, Windows XP, Windows 7	Mac OS version 10.5 et ultérieure
Mémoire RAM	1 Go (2 Go recommandés)	
Résolution écran	1280 * 1024 pixels	

2 Architecture et base de données

Plusieurs nouveautés de 4D v12 concernent l'architecture des applications 4D et le moteur de base de données intégrée :

- Prise en charge des UUID
- Nouvelle option pour la prise en charge des balises de style dans les champs alpha et texte
- Nouvelle option de réplication des données via le SQL
- Nouvelles options d'import et d'export
- Installation des composants au niveau de l'application 4D.

Prise en charge des UUID

4D v12 propose une prise en charge complète des identifiants UUID (*Universally Unique Identifier*) permettant aux développeurs 4D de les intégrer dans leurs bases de données.

Présentation

Un UUID est un type d'identifiant unique, standardisé à l'origine par la Fondation Open Software (OSF). Un identifiant UUID est conçu pour être unique dans le monde, le risque que deux ou plusieurs identifiants UUID identiques soient générés est quasiment nul. Ainsi, les informations labellisées par des UUID peuvent être combinées ultérieurement sans qu'il y ait risque de conflit de nom.

Un UUID est un nombre d'une taille de 16 octets (128 bits). Il contient 32 caractères hexadécimaux. Il peut être exprimé sous forme non canonique (suite de de 32 lettres [A-F, a-f] et/ou chiffres [0-9], par exemple 550e8400e29b41d4a716446655440000) ou sous forme canonique (groupes de 8,4,4,4,12, par exemple 550e8400-e29b-41d4-a716-446655440000).

Dans 4D v12, les numéros UUID sont stockés dans un format particulier, hébergé dans des champs de type alphanumérique spécifiques (champs dont la propriété "Format UUID" est cochée). Les valeurs stockées dans ces champs peuvent être générées automatiquement ou par programmation.

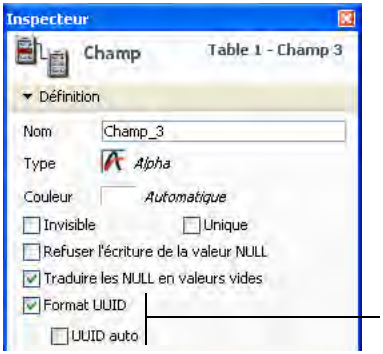
4D v12 permet :

- de désigner des champs dont les valeurs seront stockées au format UUID (16 octets).
- de générer automatiquement des UUID dans les champs au format UUID via une nouvelle option.
- de stocker dans les champs au format UUID tout identifiant UUID généré par la nouvelle commande Generer UUID ou un algorithme personnalisé.
- de rechercher des enregistrements en fonction de leur valeur UUID (attention, bien qu'hébergés dans des champs Alpha, les UUID sont des nombres : il n'est pas possible d'effectuer de recherches de type "contient" ni d'utiliser le @ comme joker).

Note Les propriétés de champs liées aux UUID sont accessibles via les commandes SQL de 4D (cf. paragraphe "Prise en charge des champs UUID", page 310).

Nouvelles propriétés UUID de champs

Dans 4D v12, deux nouvelles options ont été ajoutées dans l'Inspecteur de champ afin de configurer la prise en charge des UUID :



Pour accéder à ces options, vous devez sélectionner le type de champ Alpha.

Format UUID

Cette propriété indique que le champ est utilisé pour stocker des identifiants UUID. Les données stockées dans le champ devront respecter le format UUID (combinaison de 32 lettres (A-F, a-f) et chiffres (0-9). Vous pouvez utiliser pour cela la propriété **UUID Auto**, la nouvelle commande Generer UUID ou tout algorithme personnalisé.

Si vous tentez de stocker dans ce champ une chaîne qui n'est pas conforme au format UUID, elle est automatiquement reformatée : 4D convertit les codes des 16 premiers caractères de la chaîne en format hexadécimal et stocke le résultat. Le reste de la chaîne est tronqué. Si la chaîne contient moins de 16 caractères, les vides sont comblés par des espaces ("20" en hexadécimal). Ce mécanisme est également appliqué au contenu des champs existants non alpha transformés en champs UUID : au chargement des enregistrements, les valeurs sont reformatées puis stockées à nouveau.
Attention, ce mécanisme permet uniquement d'obtenir des valeurs correctement formatées. Il ne garantit en aucun cas l'unicité des valeurs stockées. Il vous appartient de générer des UUID valides via les outils appropriés (option "UUID Auto", commande Generer UUID ou algorithme personnalisé).

Les champs ayant la propriété **Format UUID** peuvent être affichés dans les formulaires et restent saisissables. Leur contenu est affiché en caractères majuscules. Vous devez passer par une variable si vous souhaitez afficher les minuscules.

Notes - Les champs au format UUID ne peuvent pas être associés à des index de mots-clés ni à des énumérations.
- Il est possible de créer un lien entre deux champs au format UUID mais pas de lier un champ Alpha standard à un champ au format UUID.

UUID auto

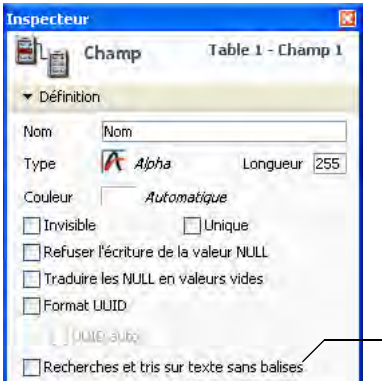
Cette option n'est active que lorsque la propriété **Format UUID** est cochée.

La propriété **UUID Auto** permet de générer automatiquement un numéro UUID dans le champ à chaque création d'enregistrement.

Note Lors de l'importation de données, même si cette propriété est cochée, 4D ne génère pas de nouveau numéro mais utilise les valeurs importées (et les transforme éventuellement si le format n'est pas valide). Toutefois, si la valeur du champ importé est vide, un UUID est automatiquement généré.

Recherche et tris dans les champs avec balises de style

La nouvelle propriété **Recherches et tris sur texte sans balise** est disponible pour les champs texte et alpha :



Option permettant d'ignorer les balises de style pour les recherches et les tris

Lorsque cette option est cochée, les recherches et les tris effectués parmi les données stockées dans le champ ne tiendront pas compte des éventuelles balises de style qu'il contient.

Cette option est liée à la nouvelle possibilité dans 4D v12 d'appliquer des styles différents à l'intérieur d'une même zone de texte dans un formulaire.

Cette nouveauté est décrite dans le paragraphe “Zones de texte riche”, page 96. La définition des styles s’effectue via l’insertion de balises HTML dans le texte. Ces balises sont interprétées au moment de l’affichage de la zone de texte.

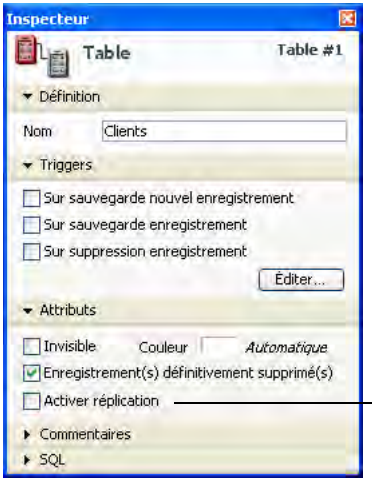
Les balises de style sont stockées avec les données. Si, par exemple, vous écrivez "week end" dans un champ texte, 4D stockera "week <SPAN STYLE = "color : #D81E05">end</SPAN>". Ce fonctionnement est transparent pour l’utilisateur au niveau du formulaire. Toutefois, pour les recherches et les tris, un paramétrage spécifique est nécessaire pour que 4D ignore les balises de style. Pour le mot "week end", la recherche n’aboutira que si vous avez coché l’option **Recherches et tris sur texte sans balise** pour le champ dans l’éditeur de structure.

Note Avec cette option, une recherche de *lavaleur* parmi les données de *lechamp* équivaut en interne à l’exécution de l’instruction CHERCHER PAR FORMULE(OBJET Lire texte brut(lechamp)="lavaleur").

Réplication des données

4D v12 propose de nouveaux mécanismes permettant de mettre en place un système élaboré de réplication des données. Ces mécanismes s’appuient sur le langage SQL.

Côté structure de la base de données 4D, une nouvelle propriété permet d’activer pour chaque table l’enregistrement des informations nécessaires à la réplication : **Activer réplication**.



Activation de la réplication

Pour plus d'informations sur cette option ainsi que sur les mécanismes de réplication des enregistrements, reportez-vous au paragraphe "Réplication via le SQL", page 297.

Définir la clé primaire

4D v12 vous permet de gérer la clé primaire d'une table directement depuis l'éditeur de structure.

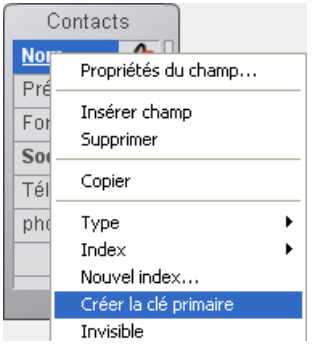
Dans le langage SQL, la clé primaire permet d'identifier dans une table la ou les colonnes (champs) chargée(s) de désigner de façon unique les enregistrements (lignes). La définition d'une clé primaire est notamment nécessaire à la fonction de réplication des enregistrements d'une table de 4D (cf. paragraphe "Réplication via le SQL", page 297). En langage SQL, la clé primaire est définie à l'aide de la clause PRIMARY KEY suivie de la liste de colonnes.

4D vous permet de créer et de supprimer directement une clé primaire via le menu contextuel de l'éditeur de structure.

▶ Pour créer une clé primaire :

1 **Sélectionnez le ou les champ(s) devant constituer la clé primaire de la table.**

2 **Cliquez avec le bouton droit de la souris et choisissez la commande Créer la clé primaire dans le menu contextuel :**



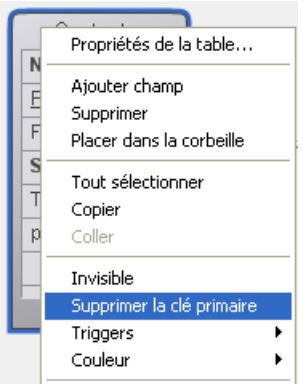
Les champs inclus dans la clé primaire apparaissent alors soulignés dans l'éditeur et leur description SQL fait apparaître le mot-clé PRIMARY KEY.

Le ou les champ(s) appartenant à la clé primaire ne doivent pas comporter de valeurs dupliquées. Si des valeurs dupliquées existaient déjà dans les enregistrements de la table, une boîte de dialogue d'alerte vous le signale.

Note La ou les colonne(s) appartenant à la clé primaire n'acceptent pas la valeur NULL.

▶ Pour supprimer la clé primaire d'une table :

1 Cliquez avec le bouton droit de la souris sur la table contenant la clé primaire et choisissez la commande Supprimer la clé primaire dans le menu contextuel :



Une boîte de dialogue de confirmation apparaît. Si vous cliquez sur **OK**, la clé primaire est immédiatement supprimée.

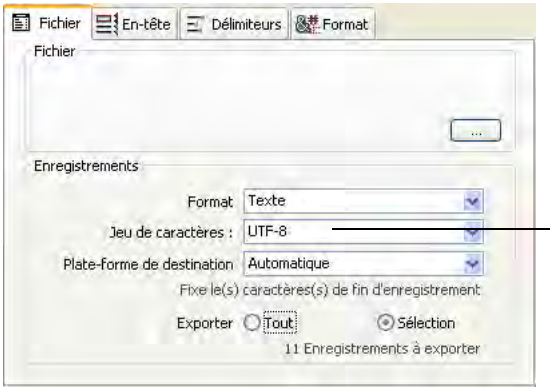
Import et export de données

De nouvelles options ont été ajoutées dans les boîtes de dialogue standard d'import et d'export de données de 4D afin de permettre une meilleure prise en charge des échanges de texte en Unicode et du format XML :

- sélection du jeu de caractères (import et export),
- intégration de la marque d'ordre des octets (export),
- nouvelles options d'export XML.

Sélection du jeu de caractères

Les boîtes de dialogue d'import et d'export de 4D v12 permettent désormais de définir le jeu de caractères utilisé pour l'échange des données dans certains formats :



Ce menu est disponible pour les formats de fichiers Texte, SYLK et XML (export). Il contient une liste standard de jeux de caractères telle que définie par l'IANA (pour plus d'informations sur ce point, reportez-vous à l'adresse http://www.iana.org/assignments/character-sets).

Note Le menu Jeu de caractères est verrouillé sur le jeu "IBM437" pour les formats DIFF et DBF et n'est pas proposé pour le format 4D.

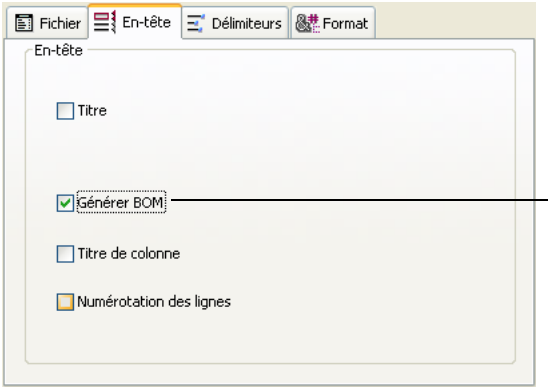
- Dans le cadre d'un export, ce menu permet de définir l'encodage à utiliser pour les données exportées.
- Dans le cadre d'un import, ce menu permet de définir l'encodage des données importées. Ce menu est inactivé si l'en-tête du fichier d'export comporte un BOM (cf. paragraphe suivant) car dans ce cas, l'encodage des données importées est automatiquement prédéfini.

  L'encodage sélectionné par défaut pour les opérations d'import et d'export est UTF-8 ou le jeu de caractères défini par la commande UTILISER FILTRE, si elle a été exécutée. A noter que la sélection d'un encodage dans la boîte de dialogue d'import ou d'export ne modifie pas le jeu de caractères courant de l'application.

Note Le menu **Plate-forme de destination**, permettant de prédéfinir le(s) caractère(s) de fin d'enregistrement, comporte de nouvelles options : **Automatique** (valeur de fin d'enregistrement définie selon la plate-forme d'export), **Unix** (fin d'enregistrement = saut de ligne) et **Personnalisée** (affiche la page des délimiteurs).

Intégrer la marque d'ordre des octets

L'option **Générer BOM pour l'Unicode** a été ajoutée dans la page "En-tête" de la boîte de dialogue standard d'export de données :



Cette option est disponible uniquement lors de l'export de données au format texte (Texte et Texte longueur fixe).

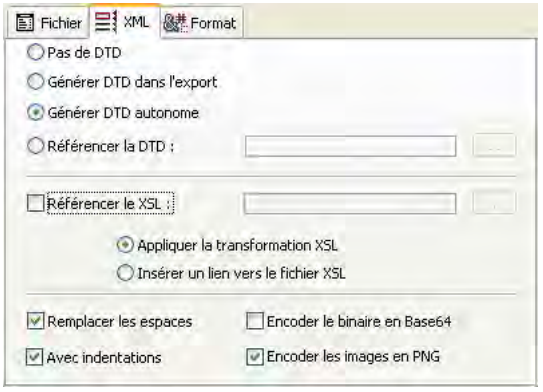
Lorsque l'option **Générer BOM pour l'Unicode** est sélectionnée, 4D insère dans l'en-tête du fichier d'export une marque d'ordre des octets (*Byte Order Mark* ou *BOM*).

Cette information supplémentaire facilite l'interprétation du texte par le logiciel d'import, s'il prend en charge cette fonction.

Cette option est cochée par défaut. Elle n'est prise en compte que si un jeu de caractères Unicode est sélectionné pour l'export. Dans le cas contraire, aucune BOM n'est ajoutée.

Nouvelles options d'export XML

De nouvelles options sont disponibles dans la page **XML** de la boîte de dialogue d'export :



- **Encoder le binaire en Base64** : si vous cochez cette option, 4D ajoute l'en-tête "data:;base64," aux champs binaires exportés (champs de type BLOB et image). Lorsque cette option n'est pas cochée, les champs sont encodés en base64 mais sans l'en-tête.
- **Avec indentations** : cette option applique une indentation automatique aux données exportées. L'indentation permet de visualiser la hiérarchie des éléments XML.
- **Encoder les images en PNG** : lorsque vous cochez cette option, les images exportées sont automatiquement encodées au format PNG, quel que soit leur format d'origine. Lorsque cette option n'est pas cochée, les images sont encodées dans leur format natif.
  A noter que dans le cas d'export d'images en SVG, il est recommandé de ne pas cocher cette option afin que les images conservent leurs propriétés.

Composants installés dans l'application 4D

Dans les versions précédentes de 4D, les composants 4D devaient être impérativement installés dans le sous-dossier **Components** de chaque base dans laquelle vous souhaitiez les utiliser. Il était donc nécessaire de dupliquer les composants ou d'utiliser des alias pour les installer à plusieurs emplacements.

Dans la version 12 de 4D, à l'image des plug-ins, il est possible d'installer les composants 4D dans l'application 4D ou 4D Server. Un composant installé de cette façon est automatiquement disponible dans toutes les bases ouvertes par l'application 4D.

Emplacements possibles pour le dossier Components

Dans 4D v12, le dossier **Components** peut donc être installé à deux endroits :

- **au niveau de l'application 4D exécutable**.
  - Windows : à côté du fichier .exe
  - Mac OS : au premier niveau du dossier Contents, à l'intérieur du package de l'application.
  Dans ce cas, les composants sont disponibles dans toutes les bases de données ouvertes par cette application.
- **au même niveau que le fichier de structure de la base** (emplacement des versions précédentes de 4D).
  Dans ce cas, les composants sont disponibles dans cette base de données uniquement.

Le choix de l'emplacement dépend de votre mode d'utilisation du composant.

Si un même composant est installé aux deux endroits, 4D charge uniquement le composant situé à côté de la structure.
Dans le cadre d'une application compilée et fusionnée avec *4D Volume Desktop*, la présence de plusieurs instances d'un même composant empêchera l'ouverture de l'application.

Note Dans 4D v12, une nouvelle propriété permet de publier un formulaire projet d'un composant en tant que sous-formulaire de la base hôte. Pour plus d'informations sur ce point, reportez-vous au paragraphe "Sous-formulaires en composants", page 81.

3 Atelier de développement

Cette section présente les nouveautés et modifications apportées à l'atelier de développement de 4D v12, au niveau des éditeurs, des préférences ou de l'interface globale :

- Réorganisation des préférences,
- Nouvelles fonctions dans l'éditeur de structure,
- Refonte de la recherche dans le développement,
- Nouvelles fonctions de recherche des éléments inutilisés,
- Nouvel éditeur de code,
- Possibilité d'exécuter du code PHP,
- Nouvelle fonction de récupération des données dans le CSM.

A noter que les nouveautés liées aux formulaires et aux objets de formulaire sont traitées dans une section spécifique, le chapitre 4, "Formulaires et objets", page 73.

Réorganisation des préférences

La gestion des préférences de 4D v12 a été entièrement réorganisée afin, notamment, de dissocier les paramétrages communs à toutes les applications 4D (les "préférences utilisateurs") et les paramétrages spécifiques à chaque base de données (les "propriétés de la base").

En outre, de nouvelles options sont disponibles.

Préférences et Propriétés

Dans 4D v12, deux boîtes de dialogue distinctes permettent de paramétrer deux ensembles d'options dont la portée est différente : la boîte de dialogue des "Préférences utilisateurs" et la boîte de dialogue des "Propriétés de la base".

Les options présentes dans la boîte de dialogue Préférences de la version précédente de 4D ont été réparties entre ces deux boîtes de dialogue en fonction de leur portée et de leur lieu de stockage (cf. paragraphes suivants). Leur fonctionnement est inchangé.

Note L'option **Langue de comparaison de texte** apparaît dans la boîte de dialogue des Préférences (page "Générale") et dans celle des Propriétés (page "Base de données"). Dans le premier cas, elle s'applique par défaut à la création de toutes les nouvelles bases et dans l'autre, elle modifie le paramétrage courant de la base ouverte.

Note Dans ce manuel, seules les nouvelles options ou celles dont le fonctionnement a été modifié sont décrites. Pour plus d'informations sur les options existantes, reportez-vous au manuel *Mode Développement*.

Préférences utilisateurs

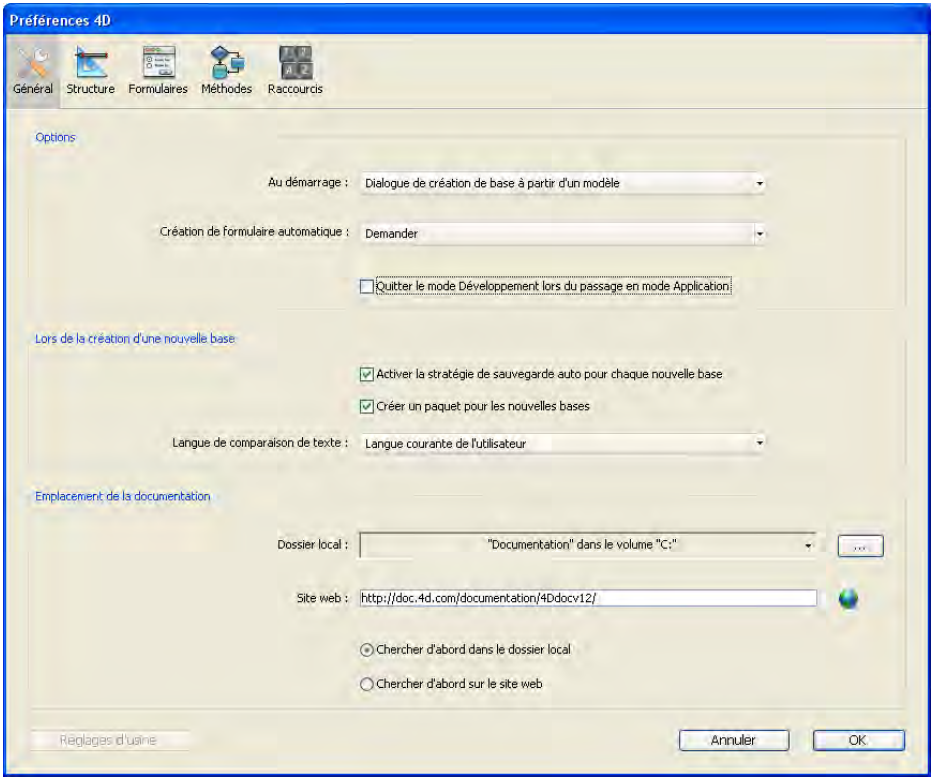
L'accès à la boîte de dialogue des Préférences s'effectue via la commande **Préférences...** du menu **Edition** (Windows) ou du menu **Application** (Mac OS) de 4D :

Cette commande de menu est disponible même lorsqu'aucune base n'est ouverte.

Les Préférences utilisateurs définissent les options de comportement par défaut des applications 4D (4D et 4D Server). Schématiquement, ces paramétrages influent sur l'environnement de travail du développeur. Ils incluent par exemple les couleurs de l'éditeur de

méthodes, l'option de création de formulaire automatique, l'affichage par défaut dans les formulaires, etc. :

Préférences utilisateurs de 4D



Les paramétrages effectués dans cette boîte de dialogue sont stockés dans un fichier de Préférences au format XML. Ce fichier est nommé **4D v12 Preferences.4DPreferences** et est stocké dans le dossier de préférences de l'utilisateur courant :

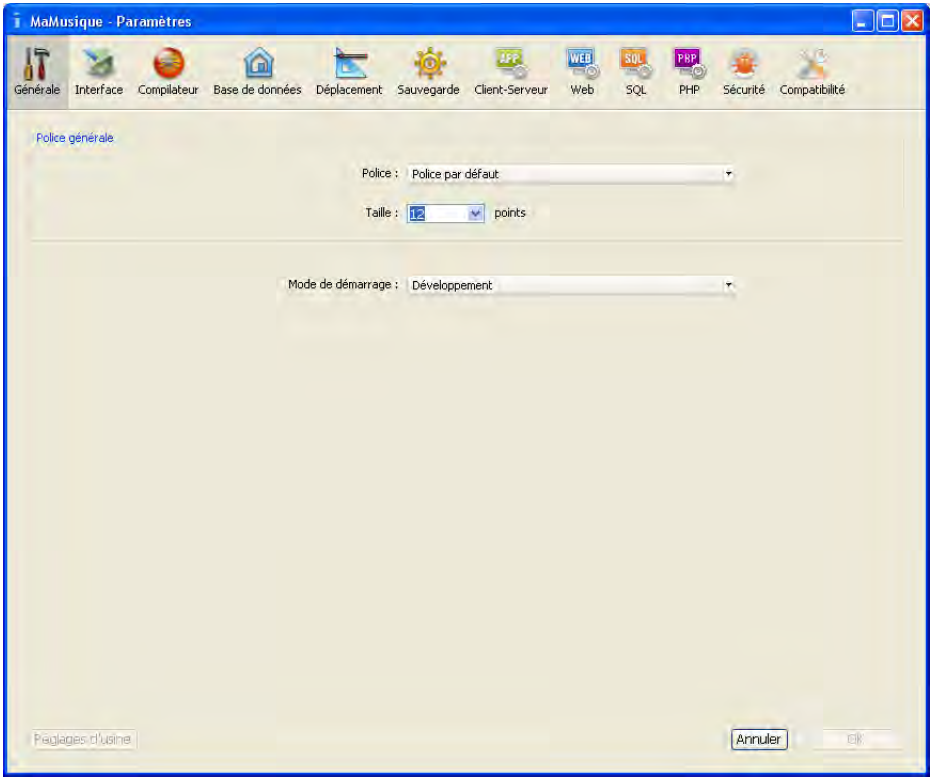
- **Windows XP** :
  C:\Documents and Settings*NomUtilisateur*\Application Data\4D\
- **Windows Vista / Windows 7** :
  C:\Users*NomUtilisateur*\AppData\Roaming\4D\
- **Mac OS** :
  {disque}:Users:*NomUtilisateur*:Library:Preferences:4D:

Propriétés de la base

L'accès à la boîte de dialogue des Réglages de la base s'effectue via la commande **Propriétés de la base...** du menu **Développement** ou le bouton correspondant de la barre d'outils de 4D :

Les Propriétés de la base configurent le fonctionnement de la base de données courante. Ces paramètres sont stockés avec la base et peuvent différer entre chaque base. Ils incluent par exemple les ports d'écoute, les droits d'accès en structure, les configurations SQL, etc. :

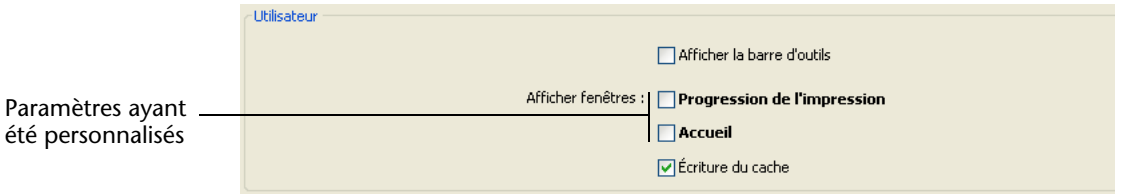
Propriétés de la base



Pour des raisons de sécurité, les propriétés de la base ne sont pas accessibles autrement que via cette boîte de dialogue.

Personnalisation des paramètres et "Réglages d'usine"

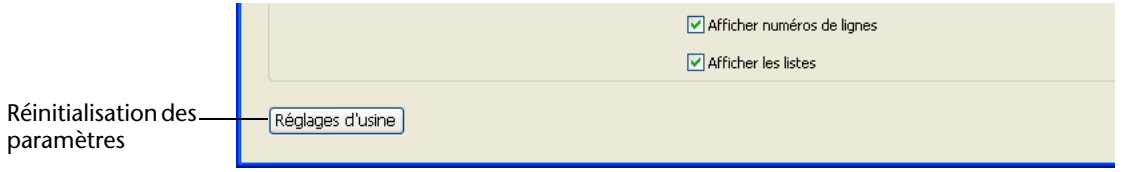
Dans les boîtes de dialogue de Préférences et de Propriétés, les libellés des paramètres dont la valeur a été modifiée apparaissent en **caractères gras** :



La modification peut avoir été effectuée dans la boîte de dialogue ou, dans le cas d'une base convertie, provenir d'une préférence modifiée précédemment.

La marque de modification est conservée même si la valeur du paramètre est manuellement replacée sur la valeur par défaut. Ce principe permet de toujours pouvoir identifier visuellement les paramètres ayant été personnalisés.

Pour réinitialiser les paramètres à leur valeur par défaut et supprimer la marque de modification des libellés, vous devez cliquer sur le nouveau bouton libellé **Réglages d'usine :**



Ce bouton réinitialise tous les paramètres de la page courante. Il est actif si au moins un paramètre a été modifié dans la page courante.

Interaction avec la commande FIXER PARAMETRE BASE

Certains paramétrages (tels que la définition du numéro d'IP de l'interpréteur PHP) peuvent être effectués soit via la boîte de dialogue des Propriétés de la base, soit via la commande FIXER PARAMETRE BASE.

Ces deux outils fonctionnent de manière complémentaire :

- les paramètres définis dans la boîte de dialogue des Propriétés de la base sont toujours utilisés par défaut pour tous les postes,
- si des paramètres sont modifiés via la commande FIXER PARAMETRE BASE, ils s'appliquent uniquement pour le poste qui a exécuté la commande et durant la session courante. Ces paramètres ne modifient pas la boîte de dialogue des Propriétés.

Ce principe permet au développeur de modifier les paramétrages localement en fonction de critères qui lui sont propres (par exemple utiliser un interpréteur PHP externe).

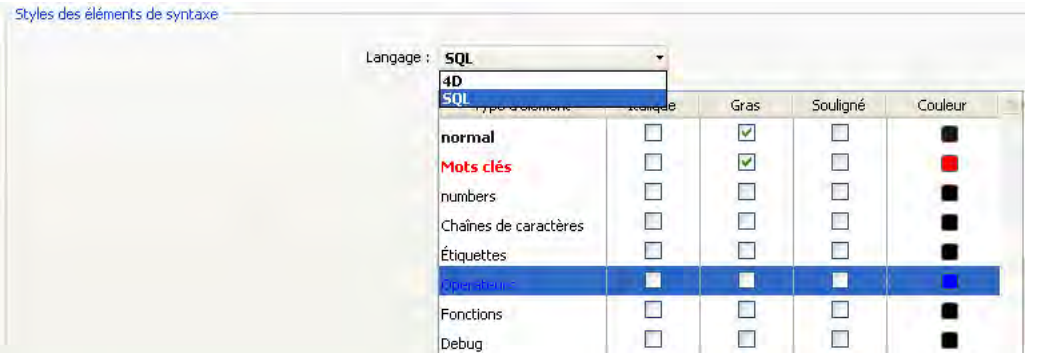
Note Certaines "anciennes" options de la commande FIXER PARAMETRE BASE ont une portée différente. Pour plus d'informations, reportez-vous à la description de cette commande.

Nouvelles préférences

De nouvelles préférences utilisateur permettant de personnaliser l'environnement de travail sont proposées dans 4D v12.

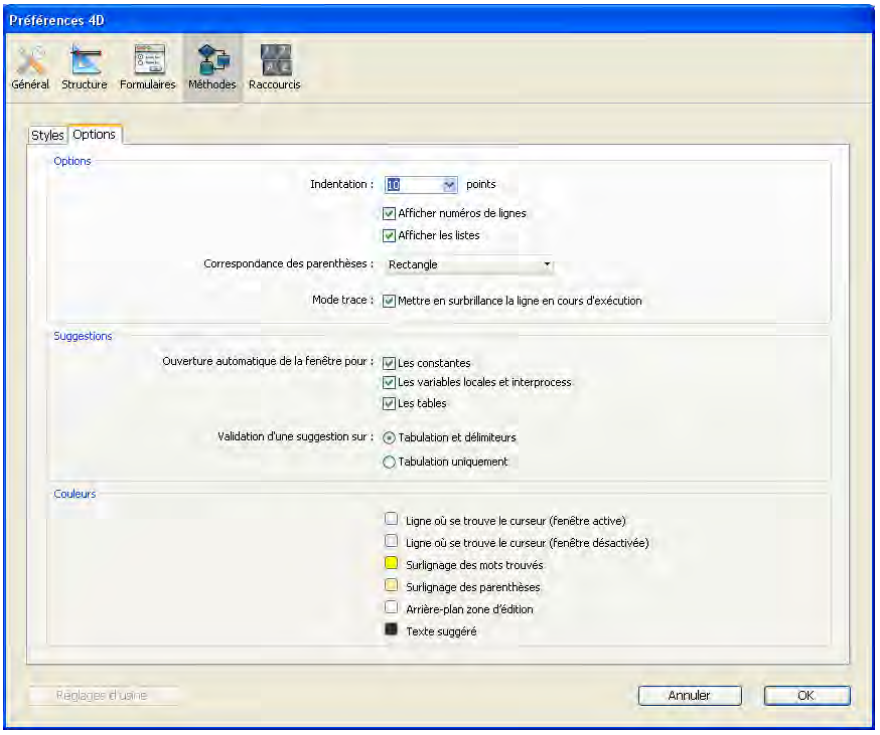
Style des éléments de syntaxe SQL

L'analyseur syntaxique du nouvel éditeur de code de 4D v12 dissocie les éléments du langage SQL (cf. paragraphe "Nouvel éditeur de code", page 57). Il est désormais possible d'attribuer un style personnalisé à chaque type d'élément via l'onglet "Styles" de la page "Méthodes" des Préférences :



Options de l'éditeur de méthode et du débogueur

L'onglet "Options" de la page "Méthodes" des Préférences propose plusieurs nouvelles options relatives à l'éditeur de méthodes et au débogueur :



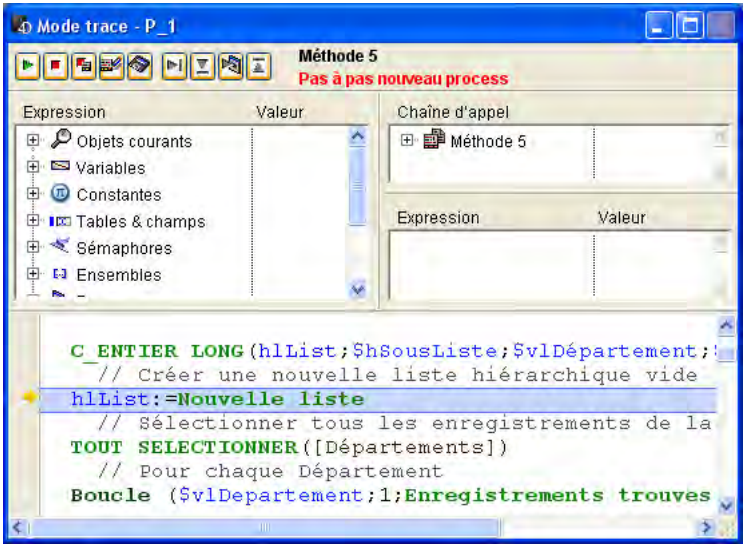
- **Correspondance des parenthèses** : cette option permet de modifier la signalisation graphique des parenthèses correspondantes dans le code. Cette signalisation apparaît lorsqu'une parenthèse est sélectionnée. Vous disposez des options suivantes :
  - **Aucune** : pas de signalisation
  - **Rectangle** : parenthèses encadrées par un filet noir
  - **Couleur de fond** : parenthèses surlignées (la couleur est définie dans la zone "Couleurs", cf. paragraphe "Couleurs", page 39)
  - **Gras** : parenthèses affichées en gras.

  Par défaut, l'option **Rectangle** est sélectionnée.

```
INSERER LIGNE MENU(main_Bar;-1;Lire chaine dans liste(79;1);FileMenu)
```

Apparence "Rectangle"

- **Mettre en surbrillance la ligne en exécution** : permet de mettre en surbrillance la ligne en exécution dans le débogueur, en plus de la traditionnelle flèche jaune :



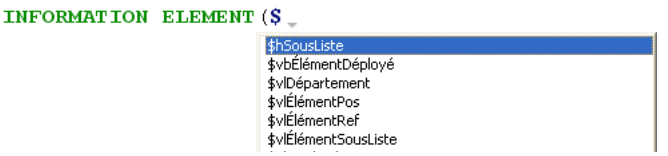
Si vous désélectionnez cette option, seule la flèche jaune est affichée (comme dans les versions précédentes de 4D).

Suggestions

Les mécanismes de saisie prédictive dans l'éditeur de méthodes ont été affinés et étendus dans 4D v12 (cf. paragraphe "Saisie prédictive", page 58). La zone "Suggestions" vous permet de configurer ces mécanismes afin de les adapter à vos habitudes de travail.

Note de compatibilité L'option "Autoriser l'aide automatique à la saisie", présente dans les versions précédentes de 4D, a été supprimée et remplacée par ces nouvelles options.

- **Ouverture automatique de la fenêtre pour** :
Cette option déclenche ou non l'affichage automatique de la fenêtre de suggestion pour les constantes, les variables interprocess et locales et les tables.
Par exemple, lorsque l'option "Les variables interprocess et locales" est cochée, la fenêtre apparaît dès que vous saisissez le caractère $:



Vous pouvez désactiver ce fonctionnement pour certains éléments de langage en désélectionnant l'option correspondante.

- **Validation d'une suggestion sur** :
Cette option définit le contexte de saisie autorisant l'éditeur de méthodes à valider automatiquement la suggestion courante affichée dans la fenêtre d'aide à la saisie.
  - **Tabulation et délimiteurs** : lorsque cette option est cochée, vous pouvez valider la suggestion courante en appuyant sur la touche **Tabulation** ou tout délimiteur pertinent dans le contexte. Par exemple, si vous saisissez "ALE" puis "(", 4D inscrira automatiquement "ALERTE(" dans l'éditeur. Voici la liste des délimiteurs pris en compte :
(; : = < [{
Ce fonctionnement équivaut à celui de la version précédente de 4D.
  - **Tabulation uniquement** : lorsque cette option est cochée, seule la touche **Tabulation** permet d'insérer la suggestion courante. Ce fonctionnement spécifique permet notamment de faciliter la saisie de caractères délimiteurs dans les noms d'éléments, par exemple ${1}.

Note Il est toujours possible de valider une suggestion via un double-clic dans la fenêtre ou la touche **Retour chariot**.

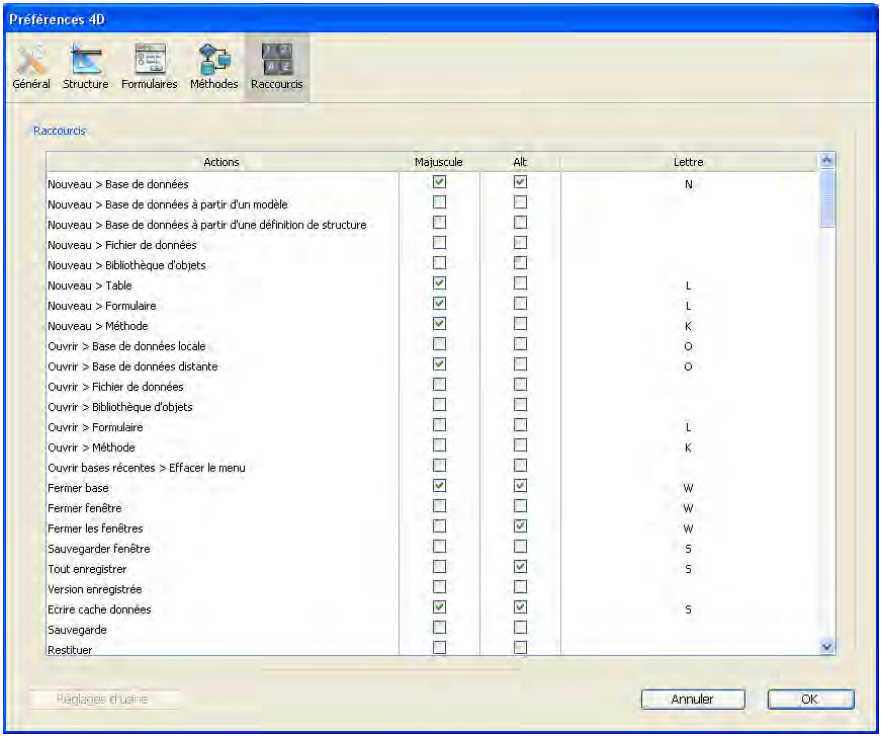
Couleurs

Ce groupe d'options vous permet de paramétrer les diverses couleurs utilisées dans l'interface de l'éditeur de méthodes.

- **Ligne où se trouve le curseur (fenêtre active) / Ligne où se trouve le curseur (fenêtre désactivée) :** Couleur de fond de la ligne contenant le curseur.
- **Surlignage des mots trouvés** : Couleur de surlignage des mots trouvés par une recherche.
- **Surlignage des parenthèses** : Couleur de surlignage des parenthèses correspondantes (utilisée lorsque les paires de parenthèses sont signalées par surlignage, cf. paragraphe "Options de l'éditeur de méthode et du débogueur", page 37).
- **Arrière-plan zone d'édition** : Couleur d'arrière-plan de la fenêtre de l'éditeur de méthodes.
- **Texte suggéré** : Couleur du texte de complètement proposé par l'éditeur de méthode en cours de saisie.

Raccourcis

Vous pouvez désormais paramétrer les raccourcis clavier via la page **Raccourcis**.



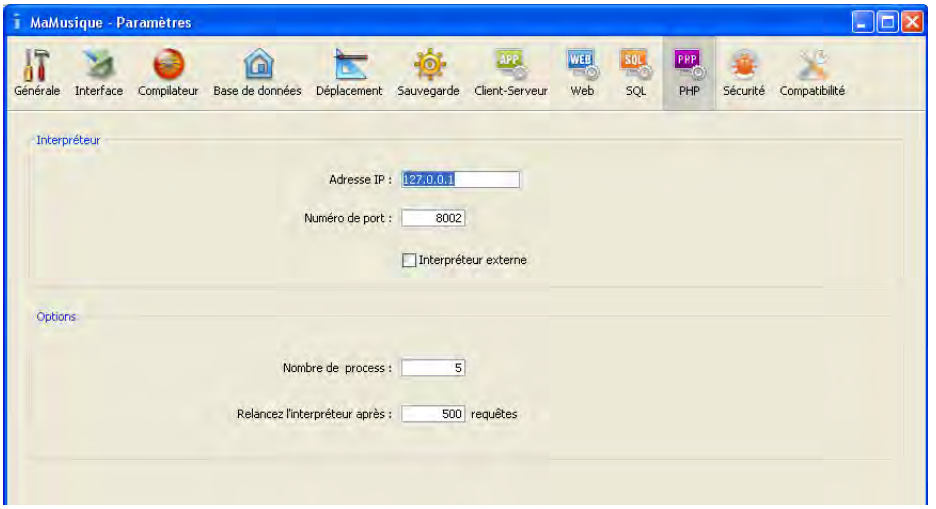
Cette page affiche la liste de tous les raccourcis-clavier utilisés dans le mode Développement de 4D (hormis les raccourcis "système" standard, tels que *Ctrl+c/Commande+c* pour la commande **Copier**). Pour modifier un raccourci, il suffit de sélectionner/désélectionner l'élément à modifier (Majuscule, Option ou touche du clavier) dans la liste des raccourcis. Vous pouvez également double-cliquer sur la ligne d'un raccourci afin de le configurer via une boîte de dialogue spécifique. A noter que chaque raccourci clavier inclut implicitement la touche **Ctrl** (Windows) ou **Commande** (MacOS).

Note Cette liste est basée sur le fichier **4DShortcutsv12.xml** placé dans le sous-dossier [4D Extensions]. Si vous personnalisez cette liste dans la boîte de dialogue, ce fichier est dupliqué dans le dossier de Préférences utilisateur et est utilisé à la place du fichier standard. Ainsi, à chaque mise à jour de 4D vous conservez vos préférences de raccourcis clavier.

Nouvelles propriétés de la base

PHP

4D v12 vous permet d'exécuter directement des scripts PHP (cf. paragraphe "Exécuter des scripts PHP dans 4D", page 65). Cette possibilité peut être configurée via la page PHP des Propriétés de la base :



- **Adresse et port de l'interpréteur**

  Par défaut, 4D fournit un interpréteur PHP compilé en Fast CGI. Pour des raisons d'architecture interne, les requêtes d'exécution sont envoyées à l'interpréteur PHP sur une adresse HTTP spécifique. Par défaut, 4D utilise l'adresse 127.0.0.1 et le port 8002. Vous pouvez modifier cette adresse et/ou ce port, par exemple s'ils sont déjà utilisés par un autre service ou si vous souhaitez installer plusieurs interpréteurs sur la même machine. Pour cela, vous pouvez modifier les paramètres **Adresse IP** et **Numéro de port** dans la page PHP des Propriétés de la base.

  Attention, l'adresse HTTP doit être située sur la même machine que 4D.

- **Interpréteur externe**

  Vous pouvez utiliser un autre interpréteur PHP que celui fourni par 4D. Il doit être compilé en FastCGI et se trouver sur la même machine que 4D (cf. paragraphe "Utiliser un autre interpréteur PHP ou un autre fichier php.ini", page 66).

Dans ce cas, vous devez cocher cette option afin que 4D ne démarre pas de connexion avec l'interpréteur interne lorsqu'une requête PHP est exécutée. A noter que dans cette configuration, vous devez gérer vous-même l'exécution et le contrôle de l'interpréteur externe.

- **Options PHP**

  Ces options concernent la gestion automatique de l'interpréteur PHP fourni avec 4D. Elles sont désactivées lorsque l'option **Interpréteur externe** est cochée.

  - **Nombre de process** : L'interpréteur PHP de 4D pilote un ensemble de process d'exécution système appelés "process enfants". Pour des raisons d'optimisation, par défaut jusqu'à cinq process enfants peuvent être lancés simultanément et conservés en permanence par l'interpréteur PHP. Vous pouvez modifier le nombre de process enfants en fonction de vos besoins via cette option. Par exemple, si vous faites intensivement appel à l'interpréteur PHP, il peut être utile d'augmenter cette valeur. Pour plus d'informations, reportez-vous au paragraphe "Architecture", page 65.

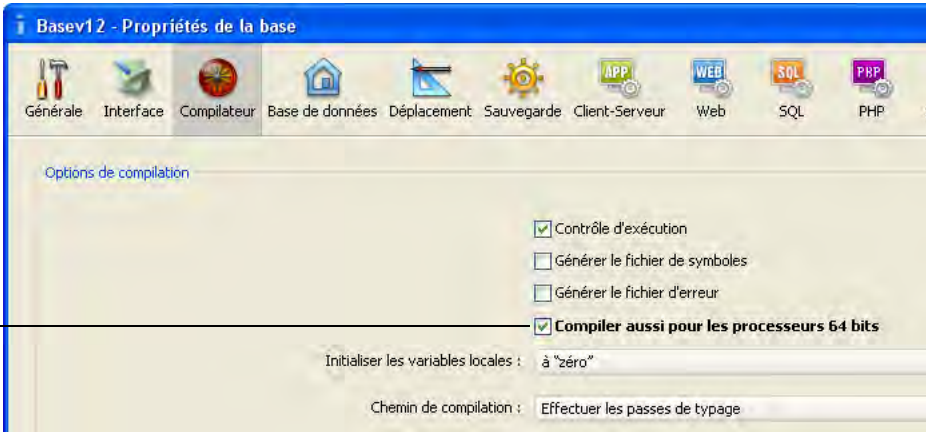
Note Sous Mac OS, tous les process enfants partagent le même port. Sous Windows, chaque process enfant utilise un numéro de port spécifique. Le premier numéro est celui défini pour l'interpréteur PHP, les autres process enfants l'incrémentent. Par exemple, si le port par défaut est le 8002 et que vous lancez 5 process enfants, ils utiliseront les ports 8002 à 8006.

  - **Relancer l'interpréteur après X requêtes** : Permet de définir le nombre maximum de requêtes acceptées par l'interpréteur PHP de 4D. Lorsque ce nombre est atteint, l'interpréteur est relancé. Pour plus d'informations sur ce paramètre, reportez-vous à la documentation de fastcgi-php.

Note Dans cette boîte de dialogue, les paramètres sont définis par défaut pour tous les postes connectés et pour toutes les sessions. Vous pouvez également les modifier et les lire séparément pour chaque poste et chaque session via les commandes FIXER PARAMETRE BASE, Lire parametre base. Les paramètres modifiés par la commande FIXER PARAMETRE BASE sont prioritaires pour la session courante.

Compilation 64 bits

Les applications 4D v12 peuvent désormais être compilées pour les processeurs 32 bits et 64 bits. Pour cela, la nouvelle option **Compiler aussi pour les processeurs 64 bits** est disponible dans la page "Compilateur" des Propriétés de la base :



Notes - La version 64 bits de 4D Server v12 est actuellement disponible en version *beta*. Pour plus d'informations, reportez-vous au paragraphe "A propos de 4D Server 64 bits", page 323.
- Il n'est pas possible de compiler en 64 bits une base 4D non convertie en mode Unicode.

Editeur de structure

De nouvelles fonctions liées à la gestion de l'affichage sont disponibles dans l'éditeur de structure.

Masquer les autres

Il est désormais possible de masquer les tables et les liens non sélectionnés dans l'éditeur de structure. Dans les versions précédentes de 4D, les tables pouvaient être masquées par dossier uniquement.

La nouvelle fonction **Masquer les autres** est disponible :

- dans le menu de gestion des dossiers de l'éditeur,

- dans le menu contextuel de l'éditeur de structure (clic sur une table ou un lien)

Lorsque vous choisissez cette commande, les tables et les liens non sélectionnés ou non connectés à des éléments sélectionnés sont masqués dans la fenêtre de l'éditeur.

Tout montrer

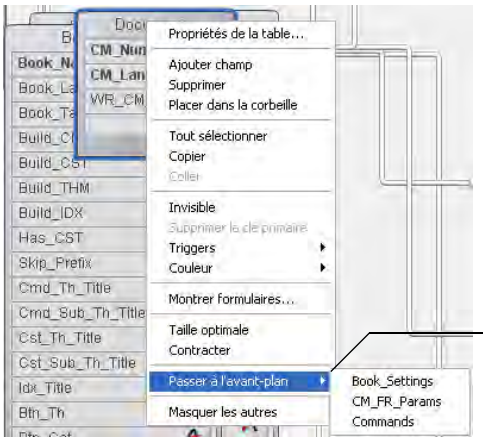
Lorsque des éléments sont masqués via la commande **Masquer les autres** ou une commande du menu de gestion des dossiers, la nouvelle commande **Tout montrer** permet de réafficher la totalité du contenu de l'éditeur. Cette commande est disponible dans le menu contextuel de l'éditeur de structure.

Passer à l'avant-plan

Lorsqu'un ensemble de deux ou plusieurs tables sont superposées dans l'éditeur de structure, il peut être parfois difficile d'afficher une des tables de l'ensemble.

Dans ce cas, il suffit de cliquer avec le bouton droit de la souris sur la table située au-dessus de toutes les autres afin de faire apparaître le sous-menu **Passer à l'avant-plan**. Ce sous-menu liste les noms de toutes

les tables empilées sous la première table. Il suffit alors de sélectionner dans ce menu le nom de la table à faire passer au premier plan :

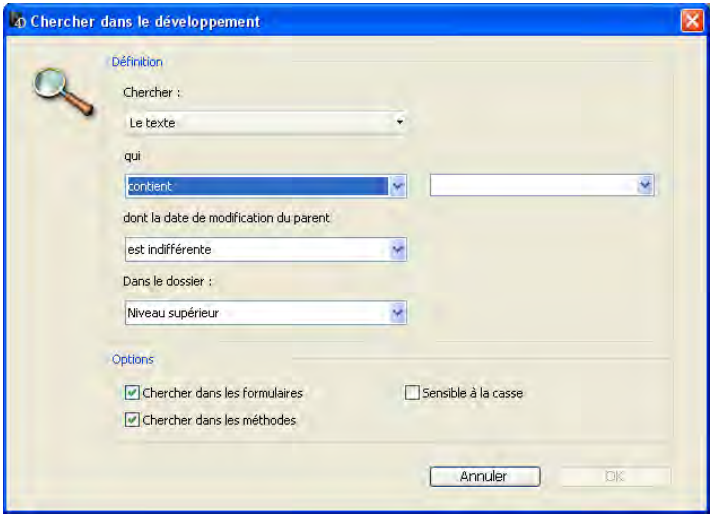


Optimisation de la recherche dans le développement

Les mécanismes de recherche et remplacement dans le développement ont été remaniés afin de mieux répondre aux besoins des développeurs 4D. Cette section décrit les nouveautés et modifications apportées à cette fonction dans 4D v12.

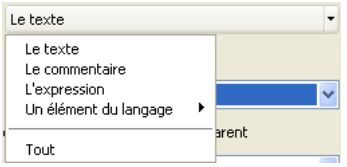
Boîte de dialogue de recherche

La boîte de dialogue de recherche apparaît lorsque vous sélectionnez la commande **Chercher dans le développement** du menu **Edition**. Dans 4D v12, les options de cette boîte de dialogue ont été modifiées :



Définition de la recherche

Le menu **Chercher** vous permet de définir le type d'élément à rechercher. Vous disposez des choix suivants :



- **Le texte** : dans ce cas, 4D recherchera simplement une chaîne de caractères parmi la totalité du développement.
  La recherche est effectuée en mode texte brut, sans tenir compte du contexte. Par exemple, vous pouvez rechercher le texte "Alerte("Erreur numéro :"+" ou "bouton27"
  Dans ce mode, il n'est pas possible d'utiliser de caractère joker. Le "@" est considéré comme un caractère standard.
- **Le commentaire** : une recherche de ce type équivaut à la précédente mais est restreinte au contenu des commentaires (lignes débutant par //). Ce principe permet par exemple de rechercher tous les commentaires contenant la chaîne "A vérifier".

Note Le résultat final de ces deux types de recherches dépend étroitement du paramétrage du menu **Qui** (cf. paragraphe "Mode de recherche", page 47).

- **L'expression de langage** : permet de rechercher toute expression 4D *valide*. La notion de validité est importante car 4D doit pouvoir évaluer l'expression pour la rechercher. Par exemple, une recherche sur l'expression "[clients" n'aboutira pas (expression invalide) alors que "[clients]" est correct.
  Cette option est particulièrement adaptée aux recherches des affectations et des comparaisons de valeurs. Par exemple :
  - recherche de "mavar:=" (affectation)
  - recherche de "mavar=" (comparaison)
- **Un élément du langage** : permet de rechercher précisément un élément de langage via son nom. 4D distingue les éléments suivants :
  - **Méthode projet** : nom de méthode projet, par exemple "M_Ajout". A noter que cette recherche (associée au mode **est exactement**) équivaut à la commande contextuelle **Chercher les références** dans l'éditeur de méthodes (cf. paragraphe "Chercher les références", page 62).

- **Formulaire** : nom de formulaire, par exemple "Entrée". La commande effectue une recherche parmi les formulaires projet et les formulaires table.
- **Table ou champ** : nom de table ou de champ, par exemple "Clients".
- **Variable** : par exemple "$mavar".
- **Constante 4D** : par exemple "est une image".
- **Chaîne entre guillemets** : constante texte littérale, c'est-à-dire toute valeur incluse entre des guillemets dans l'éditeur de code ou insérée dans les zones de texte de l'éditeur de formulaires (texte statique ou zone de groupe). Par exemple, la recherche de "Martin" aboutira si votre code contient la ligne
CHERCHER ([Clients];[Clients]Nom="Martin")
- **Commande 4D** : commande 4D, par exemple "Alerte".
- **Commande de Plug-in** : commande de plug-in installé dans l'application, par exemple "WR Chercher".

- **Tout** : cette option permet d'effectuer une recherche parmi tous les objets du mode Développement. Dans ce cas, seul le menu de date de modification est disponible. Typiquement, cette option permet par exemple de rechercher "tous les objets modifiés aujourd'hui".

Mode de recherche

Le menu de mode de recherche (nommé **qui**, **qui est** ou **dont le nom**, en fonction de la définition de la recherche) permet de préciser la manière dont la valeur saisie va être recherchée. Le contenu de ce menu varie en fonction du type d'élément à rechercher sélectionné dans le menu précédent.

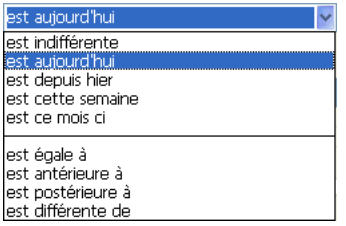
- Recherche de **texte** ou de **commentaire** : lorsque ce type de recherche est sélectionné, le menu propose les options suivantes :
  - **Contient** : recherche la chaîne parmi les textes du mode Développement. La recherche de "var" trouvera "mavar", "variable1" ou "avarie".
  - **Contient le mot** : recherche la chaîne en tant que mot parmi les textes du mode Développement. La recherche de "var" ne trouvera que les occurrences exactes de "var", elle ne trouvera pas "mavar". En revanche, elle trouvera "var:=10" ou "ID+var" car les symboles : ou + sont des séparateurs de mots.

- **Commence par / Se termine par** : recherche la chaîne en début ou en fin de mot (recherche de texte) ou de ligne de commentaire (recherche de commentaire). En mode "Texte se termine par", la recherche de "var" trouvera "mavar".

- Recherche d'**élément du langage** : lorsque ce type de recherche est sélectionné, le menu propose les options standard (**est exactement**, **contient**, **commence par**, **se termine par**). Ces options sont semblables à celles proposées dans les versions précédentes de 4D.
A noter que vous pouvez utiliser le joker de recherche (@) avec l'option **est exactement** (retourne tous les objets du type défini).

Date de modification du parent

Ce menu contient plusieurs nouvelles options vous permettant de définir en un clic une période de recherche standard :

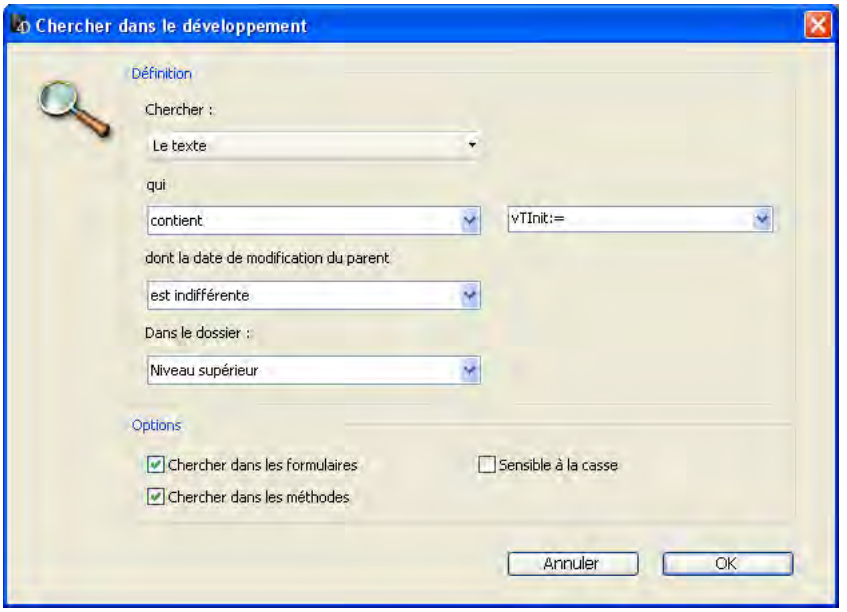


- **est aujourd'hui** : période débutant à 00:00 h du jour en cours.
- **est depuis hier** : période incluant le jour en cours et le jour précédent.
- **est cette semaine** : période débutant le lundi de la semaine en cours.
- **est ce mois-ci** : période débutant le 1er jour du mois courant.

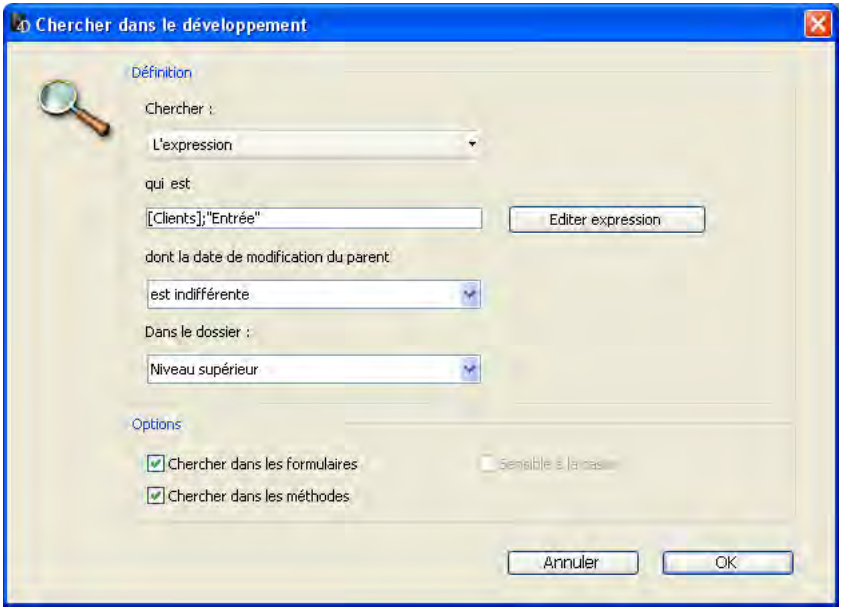
Exemples de recherches

Une recherche efficace résulte de la combinaison judicieuse des options des menus **Quoi** et **Qui**. Afin d'illustrer le fonctionnement de la recherche dans 4D v12, voici des exemples de recherches types et les paramétrages à effectuer.

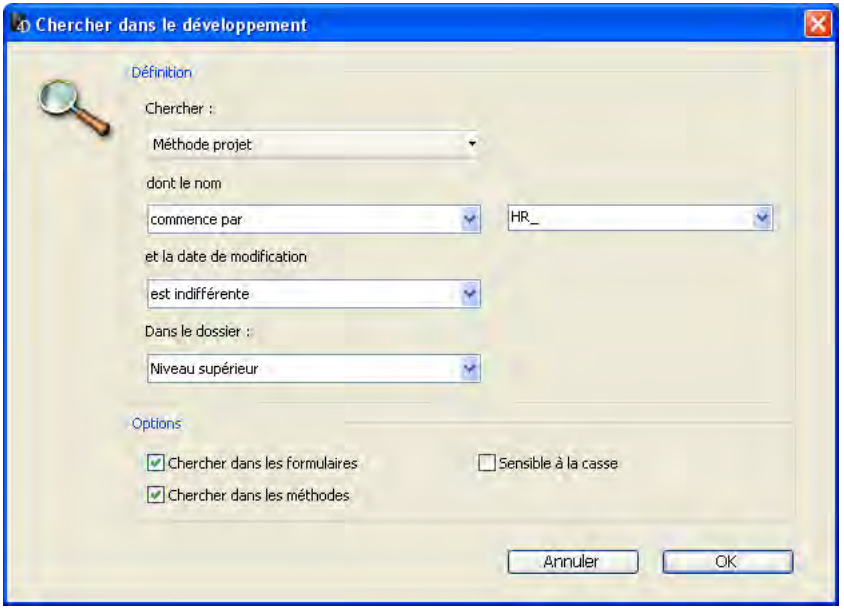
- Recherche de tous les endroits où une valeur a été affectée directement à la variable *vTInit* :



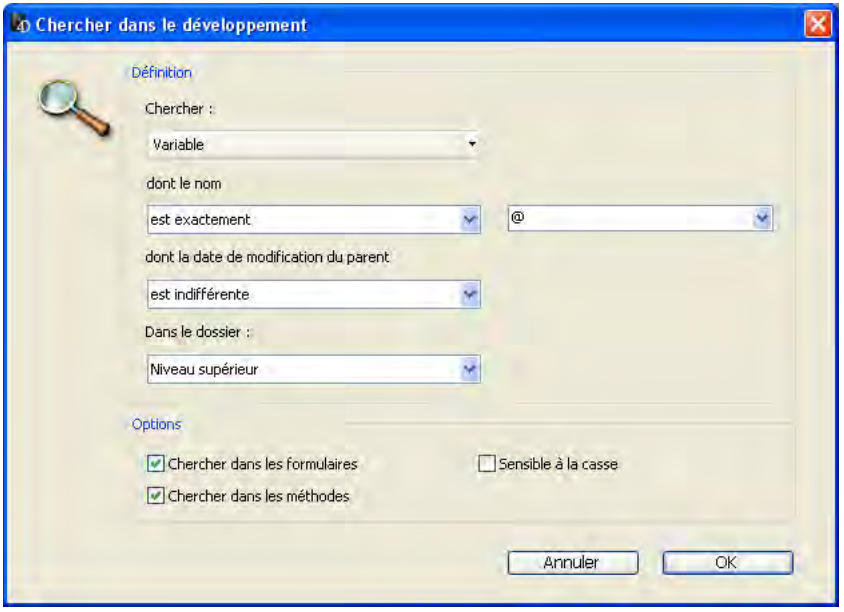
- Recherche des références au formulaire "Entrée" de la table [Clients] :



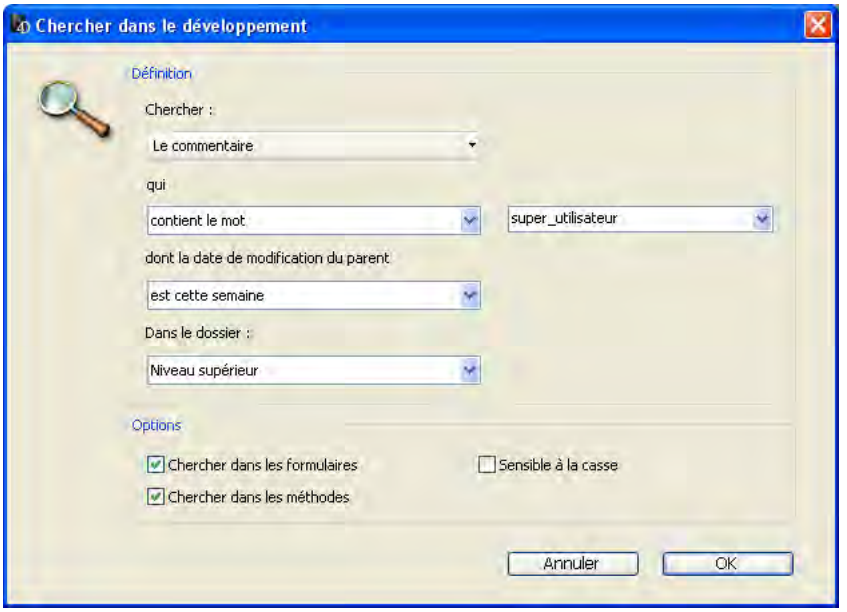
- Recherche des appels des méthodes dont le nom débute par "HR_"



- Liste de toutes les variables de la base :



- Recherche du mot-clé "super_utilisateur" dans les commentaires écrits cette semaine :



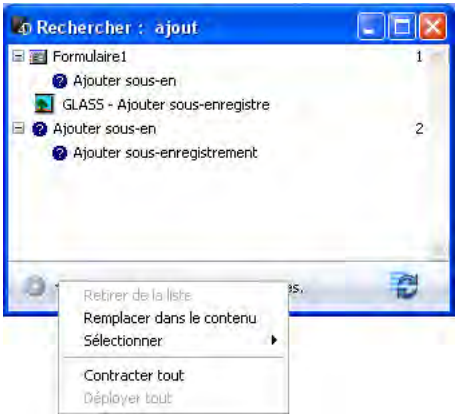
Remplacer et Renommer

Les fonctions de renommage et de remplacement ont été simplifiées dans 4D v12. Seules deux commandes sont désormais disponibles :

- **Remplacer dans le contenu**, accessible depuis la fenêtre de résultat d'une recherche. Cette commande permet de remplacer une chaîne de caractères par une autre à l'intérieur des objets listés.
- **Renommer**, accessible depuis l'Explorateur et le Nouvel éditeur de code. Cette commande permet de renommer une méthode projet ou une variable et de propager le nouveau nom parmi tous les appelants de l'objet.

Remplacer dans le contenu

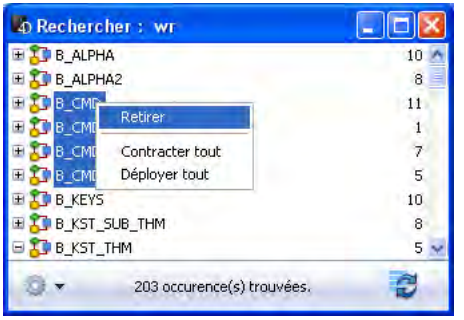
Tout remplacement est désormais effectué via une nouvelle commande disponible dans le menu d'options de la fenêtre de résultat : **Remplacer dans le contenu**.



Lorsque vous sélectionnez cette commande, une boîte de dialogue vous permet de saisir la chaîne de caractères qui remplacera toutes les occurrences trouvées par la recherche initiale :

Les principes de fonctionnement du remplacement sont désormais les suivants :

- Le remplacement est toujours effectué parmi la totalité des éléments présents dans la liste et non sur une sélection. Il est cependant possible d'affiner le remplacement en réduisant au préalable le contenu de la liste à l'aide de la nouvelle commande **Retirer** du menu contextuel :



  Le remplacement n'est effectué que sur les occurrences affichées dans la liste et après vérification des critères de recherches initiaux pour le cas où des objets auraient été modifiés entre la recherche et le remplacement.

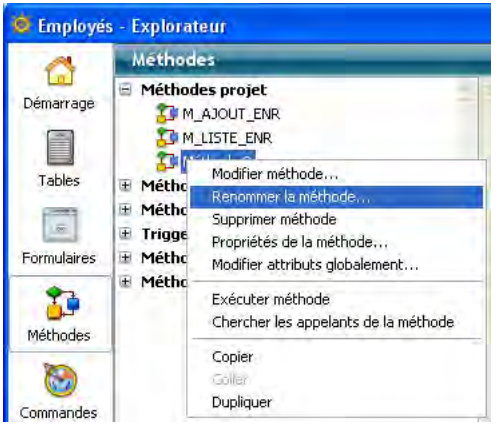
- Le remplacement est effectué dans :
  - le contenu des méthodes
  - les propriétés d'objets de formulaire
  - le contenu des messages d'aide
  - le contenu des filtres de saisie
  - le contenu des éléments des menus (libellés et appels de méthodes)
  - le contenu des énumérations
  - le contenu des commentaires sur les méthodes, les formulaires, les tables et les champs dans l'Explorateur.
- Pour chaque objet modifié, 4D teste s'il n'est pas déjà chargé par un autre poste ou une autre fenêtre. En cas de conflit, une boîte de dialogue standard s'affiche, indiquant que l'objet est verrouillé. Il est alors possible de refermer l'objet puis de réessayer ou d'annuler le remplacement. L'opération continue ensuite avec les autres objets de la liste.

- Si une méthode ou un formulaire concerné(e) par un "remplacer dans le contenu" est déjà ouvert en édition par le même 4D, l'objet sera modifié directement dans l'éditeur ouvert (aucune alerte n'est affichée). Les formulaires et méthodes modifiés de cette manière ne sont pas automatiquement sauvegardés : vous devez explicitement appeler la commande **Sauvegarder** ou **Tout enregistrer** afin de valider les modifications.
- Lorsqu'un remplacement a été effectué dans un élément de la liste, l'élément est affiché en italique. Un décompte en temps réel des remplacements effectués apparaît en bas de la fenêtre.
- Les objets ne sont jamais renommés par la fonction **Remplacer dans le contenu**, à l'exception des objets dans les formulaires. Par conséquent, il est possible que certaines occurrences de la liste qui correspondent aux critères de recherche initiaux uniquement par leur nom ne soient pas concernées par le remplacement (dans ce cas, tous les éléments de la liste ne sont pas forcément affichés en italique et le décompte des remplacements est inférieur au nombre d'occurrences trouvées par la recherche initiale).

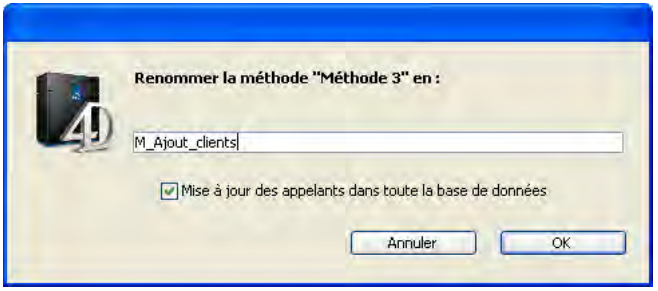
Renommer

La fonction de renommage avec propagation dans toute la base est désormais réservée aux **méthodes projet** et aux **variables**. Elle est accessible :

- via la commande **Renommer...** du menu contextuel de l'éditeur de méthodes (méthodes projet et variables),
- via la commande **Renommer la méthode...** du menu contextuel de l'Explorateur (méthodes projet).



Lorsque vous sélectionnez cette commande, une boîte de dialogue apparaît, vous permettant de saisir le nouveau nom de l'objet :



Le nouveau nom doit être conforme aux règles de nommage, sinon un message d'alerte s'affiche au moment de la validation. Par exemple, il n'est pas possible de renommer une méthode avec un nom de commande comme "Alerte".

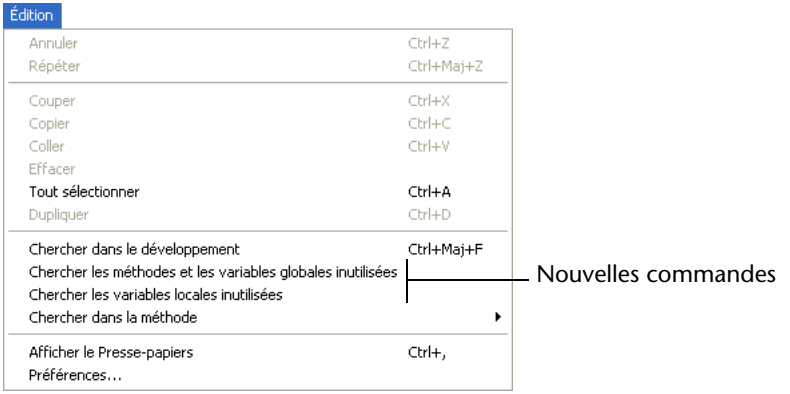
Suivant le type d'objet que vous renommez (méthode projet ou variable), la boîte de dialogue de renommage contiendra ou non une option de propagation :

- **Méthode projet** : l'option **Mise à jour des appelants dans toute la base de données** permet de propager le renommage de la méthode dans tous les objets de la base qui la référencent. Vous pouvez également désélectionner cette option pour, par exemple, renommer une méthode uniquement au niveau de l'Explorateur.
- **Variable process et interprocess** : l'option **Renommer la variable dans toute la base de données** permet de propager le renommage de la variable dans tous les objets de la base qui la référencent. Si vous désélectionnez cette option, la variable sera renommée uniquement dans la méthode courante.
- **Variable locale** : pas d'option de propagation pour cet objet, la variable est renommée uniquement dans la méthode courante.

Rechercher les éléments inutilisés

Deux nouvelles commandes de recherche vous permettent de détecter les variables et méthodes non utilisées dans votre code. Vous pourrez alors les supprimer afin, notamment, de libérer de l'espace mémoire.

Ces commandes sont accessibles dans le menu **Edition** du mode Développement :



Chercher les méthodes et les variables globales inutilisées

La commande **Chercher les méthodes et les variables globales inutilisées** recherche les méthodes projet ainsi que les variables "globales" (variables process et variables interprocess) déclarées et inutilisées. Le résultat de la recherche est affiché dans une fenêtre de résultat standard.

Une méthode projet est considérée comme inutilisée lorsque :

- elle n'est pas dans la corbeille,
- elle n'est appelée nulle part dans le code 4D,
- elle n'est pas appelée par une commande de menu,
- elle n'est pas appelée sous forme de constante chaîne dans le code 4D (4D détecte un nom de méthode dans une chaîne même s'il est suivi de paramètres entre parenthèses).

Une variable process ou interprocess est considérée comme inutilisée lorsque :

- elle est déclarée dans le code 4D au moyen d'une commande de déclaration, du type C_XXX ou TABLEAU XXX,
- elle n'est utilisée nulle part ailleurs dans le code 4D,
- elle n'est utilisée dans aucun objet de formulaire.

Attention, certains cas d'utilisation ne peuvent pas être détectés par la fonction - un élément considéré comme inutilisé peut être en réalité utilisé. C'est le cas par exemple dans le code suivant :

```
v:="methode"
EXECUTER FORMULE("ma"+v+Chaine(42))
```

Ce code construit un nom de méthode. La méthode projet *mamethode42* sera considérée comme inutilisée alors qu'elle est appelée.
Par conséquent, il est conseillé de vérifier l'inutilité des éléments déclarés inutilisés avant de les supprimer.

Chercher les variables locales inutilisées

La commande **Chercher les variables locales inutilisées** recherche les variables locales déclarées et inutilisées. Le résultat de la recherche est affiché dans une fenêtre de résultat standard.

Une variable locale est considérée comme inutilisée lorsque :

- elle est déclarée dans le code 4D au moyen d'une commande du type C_XXX ou TABLEAU XXX,
- elle n'est utilisée nulle part ailleurs au sein de la même méthode.

Nouvel éditeur de code

Dans 4D v12, l'éditeur de code a été enrichi de multiples nouvelles fonctions destinées à augmenter la productivité des développeurs.

En particulier, l'éditeur vous permet de bénéficier désormais du même confort (aide à la saisie, débogage...) pour le code 4D et le code SQL.

```
Debut SQL
   SELECT Activité.CodeActivité
     FROM Activité
     WHERE (Activité.Responsable= 'Durant')
       OR (Activité.Responsable= 'Dupond' AND Activité.Designation = 'Tennis');
Fin SQL
```

L'éditeur de 4D v12 reprend la plupart des caractéristiques et outils de l'éditeur de 4D v11 SQL et apporte des améliorations et fonctions supplémentaires. Dans ce manuel, seules les nouveautés et les différences sont détaillées. Pour une description des fonctions élémentaires de l'éditeur, veuillez vous reporter au manuel *Mode Développement* de 4D v11 SQL.

Aides à la saisie

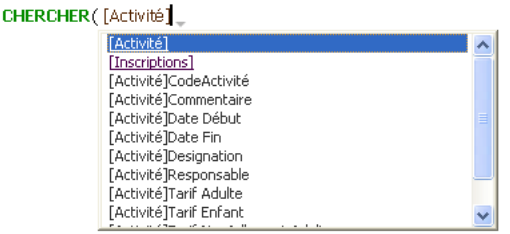
Plusieurs nouvelles fonctions d'aides à la saisie et à l'édition du code apparaissent dans 4D v12.

Saisie prédictive

Les mécanismes de saisie prédictive s'appliquent désormais au code SQL et tiennent mieux compte des paramètres déjà saisis, tels que le paramètre optionnel "*". Deux nouveautés sont proposées :

- lorsque les caractères saisis correspondent à une seule valeur possible, elle apparaît en grisé (et est insérée lorsque vous appuyez sur la touche **Tabulation**) :

- la saisie prédictive des paramètres a été étendue aux variables ainsi qu'aux noms de tables et de champs et tient mieux compte des paramètres déjà saisis, tels que le paramètre optionnel "*".



Note Les mécanismes de saisie prédictive peuvent être configurés dans les Préférences de l'application (cf. paragraphe "Suggestions", page 38).

Sélection

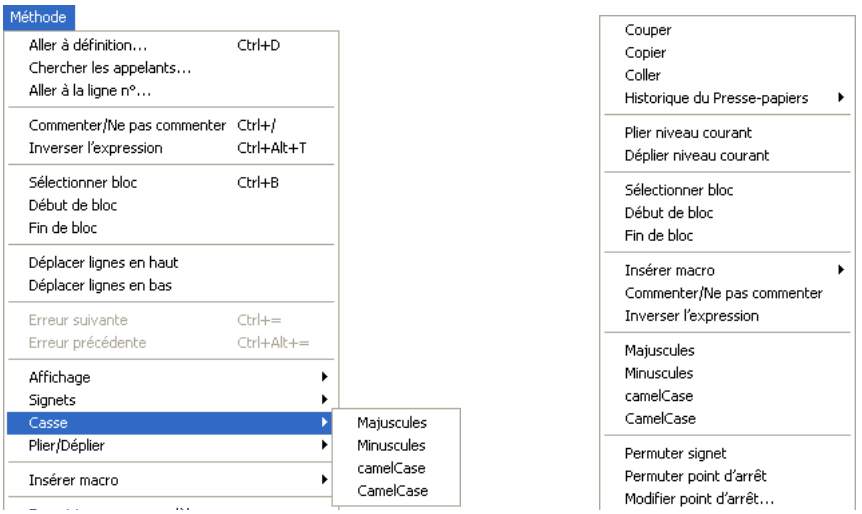
L'éditeur de méthodes bénéficie de nouvelles fonctions de sélection :

- Le double-clic permet de sélectionner des "mots". Lorsque le nom d'un élément référencé (commande, constante, méthode...) contient des espaces, vous pouvez le sélectionner entièrement en utilisant la combinaison **Alt/Option** + **Double-clic**.
- Vous pouvez déplacer directement la ligne contenant le curseur sans devoir la sélectionner au préalable : suffit d'utiliser la combinaison **Alt/Option** + **Flèche haut** ou **Flèche bas**. Cette fonction est également accessible via les nouvelles commandes **Déplacer lignes en haut** et **Déplacer lignes en bas** du menu **Méthode**.
- La fonction de sélection de bloc a été enrichie de deux nouvelles commandes : **Début de bloc** et **Fin de bloc** (menu **Méthode**) permettent respectivement de placer le curseur au début et à la fin d'un bloc de code (délimité par une structure logique du type **Si...Fin de si**). Par ailleurs, la commande **Sélectionner bloc** sélectionne désormais

dans un premier temps les blocs sans les caractères enveloppants (parenthèses, crochets, etc.).

Changer la casse

Plusieurs commandes vous permettent de modifier automatiquement la casse des caractères sélectionnés. Ces commandes sont accessibles via le sous-menu **Casse** du menu **Méthode** ou le menu contextuel de l'éditeur :



- **Majuscules** / **Minuscules** : ces commandes vous permettent de passer les caractères sélectionnés en majuscules ou minuscules.
- **camelCase** / **CamelCase** : ces commandes vous permettent de passer les caractères sélectionnés en "camel case". Cette pratique consiste à mettre en majuscule chaque première lettre d'un groupe de mots accolés. Ce type d'écriture est souvent utilisé pour les nomenclatures de variables. *jourEmbauche* et *DateAchat* sont des exemples des deux variantes des écritures en camel case.
  Lorsque vous appliquez l'une de ces commandes à une sélection de texte, les espaces et caractères "_" sont supprimé, les mots sont accolés et leur lettre initiale est passée en majuscule.

Saisie sur plusieurs lignes

Vous pouvez désormais écrire une seule instruction sur plusieurs lignes. Il suffit pour cela de terminer chaque ligne de l'instruction par le caractère "\". 4D considérera toutes les lignes en une seule fois. Par exemple, ces deux instructions sont strictement équivalentes :

```
[DOCUMENTATIONS]PlurialForm:=Majusc([DOCUMENTATIONS]PlurialForm[[1]])+Minusc(Sous chaine([DOCUMENTATIONS]PlurialForm;2))

[DOCUMENTATIONS]PlurialForm:=Majusc([DOCUMENTATIONS]PlurialForm[[1]])+\
Minusc(Sous chaine([DOCUMENTATIONS]PlurialForm;2))
```

Plier/déplier la sélection

De nouvelles commandes du menu **Méthode** vous permettent de plier ou de déplier des parties de code.

Note Ces actions sont appelées Contracter/Déployer dans les versions précédentes de 4D.

- **Plier** / **Déplier la sélection** : permet de plier ou de déplier toutes les structures de code contenues dans la sélection de texte.
- **Plier** / **Déplier le niveau courant** : permet de plier ou de déplier la structure de code au niveau de laquelle se trouve le curseur. .Ces commandes sont également accessibles via le menu contextuel de l'éditeur.

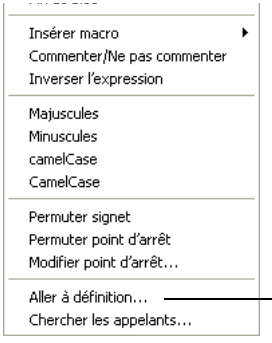
IME (Windows)

Sous Windows, le nouvel éditeur de code inclut un IME (*Input Method Editor*) permettant de faciliter l'édition du code sur des systèmes japonais ou chinois.

Navigation

Aller à définition

La nouvelle commande **Aller à définition** vous permet d'ouvrir directement dans une nouvelle fenêtre la définition d'un objet référencé dans l'éditeur de méthodes (méthode, champ, table ou formulaire). Pour cela, il vous suffit d'insérer le curseur dans le nom de l'objet (ou de le sélectionner) et de choisir la commande **Aller à définition...** dans le menu **Méthode** ou le menu contextuel de l'éditeur :



Nouvelle commande dans le menu contextuel de l'éditeur de méthodes

Cette commande fonctionne avec les noms d'objets suivants :

- *méthode projet* : affiche le contenu de la méthode dans une nouvelle fenêtre de l'éditeur de méthodes,

Note Cette fonction est accessible via le raccourci clavier **Ctrl+K** (Windows) ou **Commande+K** (Mac OS) lorsque le nom de la méthode projet est sélectionné.

- *champ* : affiche les propriétés du champ dans l'inspecteur de la fenêtre de structure,
- *table* : affiche les propriétés de la table dans l'inspecteur de la fenêtre de structure,
- *formulaire* : affiche le formulaire dans l'éditeur de formulaires,
- *variable* (locale, process, interprocess ou paramètre $n) : affiche la ligne de déclaration de la variable dans la méthode courante ou parmi les méthodes compilateur.

Aller au début ou à la fin du bloc

Deux nouvelles commandes facilitent la navigation au sein des structures de code (Si...Sinon...Fin de si, etc.) :

- **Début de bloc** : lorsque vous sélectionnez cette commande, le curseur est placé juste avant le mot-clé d'ouverture de la structure courante.
- **Fin de bloc** : lorsque vous sélectionnez cette commande, le curseur est placé juste après le mot-clé de fermeture de la structure courante.

Vous pouvez accéder à ces commandes via le menu **Méthode** ou le menu contextuel de l'éditeur. Vous pouvez également utiliser les combinaisons de touches suivantes :

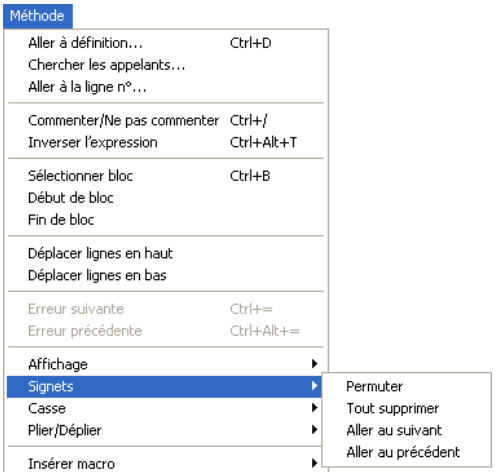
- Windows : Ctrl + ↑ ou Ctrl + ↓
- Mac OS : Commande + ↑ ou Commande + ↓

Signets

4D v12 vous permet d'associer des signets à certaines lignes de vos méthodes. Vous pouvez alors naviguer rapidement parmi le code en passant d'un signet à l'autre via des commandes spécifiques.

Un signet se déplace avec sa ligne d'origine si des lignes supplémentaires sont insérées dans la méthode. Les signets sont sauvegardés avec les méthodes.

La gestion des signets est effectuée via le sous-menu **Signets** du menu **Méthode** :



- **Permuter** : permet d'associer un signet à la ligne courante (ligne dans laquelle se trouve le curseur) si elle n'en comportait pas ou de supprimer le signet de la ligne dans le cas contraire.
  Cette fonction est accessible via la commande **Permuter signet** du menu contextuel de l'éditeur ou le raccourci clavier **Ctrl+F3** (Windows) ou **Commande+F3** (Mac OS).
- **Tout supprimer** : supprime tous les signets de la fenêtre de premier plan.
- **Aller au suivant** / **Aller au précédent** : ces commandes permettent de naviguer parmi les signets définis dans la fenêtre. La sélection d'une commande provoque le placement du curseur sur le premier caractère de la ligne associée au signet. Vous pouvez également utiliser les raccourcis clavier **F3** (aller au suivant) ou **Maj+F3** (aller au précédent).

Chercher les références

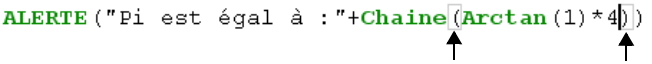
La commande **Chercher les références...**, accessible dans le menu **Méthode** ou le menu contextuel de l'éditeur, remplace la commande **Chercher les appelants**, présente dans les versions précédentes de 4D. **Chercher les appelants** affichait la liste des objets référençant (appelant) la sous-méthode sélectionnée dans une fenêtre de résultat de recherche.

La commande **Chercher les références** conserve ce fonctionnement et l'étend aux variables et aux champs. Elle permet donc de rechercher toutes les méthodes dans lesquelles la variable ou le champ sélectionné(e) est référencé.

Visualisation

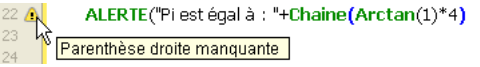
De nouveaux repères visuels facilitent la lisibilité et la maintenance du code.

- Lorsque vous cliquez à côté d'un caractère enveloppant (parenthèse ou crochet), il est entouré d'un bloc gris et le caractère correspondant l'est également :

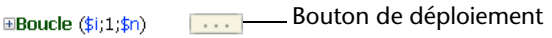


  Ce principe est aussi utilisé au moment de la saisie d'un crochet ou d'une parenthèse fermante.

- Les erreurs sont signalées par une icône accompagnée d'une info-bulle à gauche de la zone d'édition :

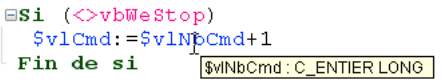


- Lorsqu'un bloc de code est plié (contracté), seule la première ligne est visible et un bouton de déploiement apparaît automatiquement :

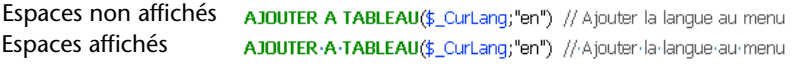


  Si vous placez le pointeur de la souris au-dessus du bouton de déploiement, une info-bulle apparaît, affichant les premières lignes du code masqué.

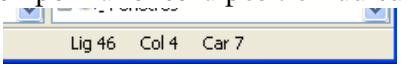
- Une info-bulle permet désormais de visualiser le type des variables :



- Il est désormais possible de matérialiser les espaces entre les mots à l'aide de symboles au lieu de "blancs". Cette fonction, accessible via la commande **Affichage** > **Espaces** du menu **Méthode**, s'applique à la totalité des éléments du code (noms de commandes, variables, commentaires, etc.) :



- La barre de statut située en bas à droite de la fenêtre de l'éditeur affiche en permanence la position du curseur :



  - **Lig** : numéro de la ligne
  - **Col** : numéro de colonne, c'est-à-dire le niveau parmi la hiérarchie des structures de programmation. Le premier niveau a le numéro 0. Le numéro de colonne est utile dans le cadre du débogage car cette information peut être fournie par l'interpréteur en cas d'erreur dans le code.
  - **Car** : Emplacement du caractère dans la ligne.

- Les nouvelles commandes **Agrandir la police** et **Réduire la police** du sous-menu **Méthode > Affichage** permettent de modifier temporairement la taille de la police dans la fenêtre de premier plan. Cette modification n'est pas conservée à la fermeture de la fenêtre. Pour modifier de façon permanente la taille de la police des fenêtres de l'éditeur de méthodes, utilisez la page "Méthodes" dans les Préférences de l'application.

- **Suivi des modifications** : des barres colorées vous permettent de visualiser instantanément les lignes de code qui ont été modifiées depuis l'ouverture de la méthode :

  Les barres de suivi changent de couleur suivant que la modification a été sauvegardée ou non :

  - jaune : la ligne a été modifiée et la méthode n'a pas encore été sauvegardée.
  - vert : la ligne a été modifiée et la méthode a été sauvegardée.

Commentaires

Le code est désormais commenté via le double caractère //. Vous pouvez commenter/décommenter une ligne ou un bloc de lignes en utilisant le raccourci **Ctrl+/** (Windows) ou **Commande+/** (Mac OS).

Note Par compatibilité, le caractère ` peut toujours être utilisé pour commenter une ligne.

Exécuter des scripts PHP dans 4D

4D v12 permet d'exécuter directement des scripts PHP. Cette nouvelle possibilité vous donne accès à toute la richesse des librairies utilitaires disponibles via PHP. Ces librairies fournissent en particulier des fonctions de :

- chiffrage (MD5) et hachage,
- manipulation des fichiers ZIP,
- manipulation d'images,
- accès LDAP,
- accès COM (pilotage des documents MS Office)...

Cette liste n'est pas exhaustive. Pour une liste complète des modules PHP disponibles par défaut avec 4D, reportez-vous à l'annexe A, "Modules PHP," page 327. A noter également qu'il est possible d'installer des modules personnalisés supplémentaires (cf.).

Architecture

4D fournit un interpréteur PHP compilé en FastCGI, protocole de communication de type client-serveur entre une application et un interpréteur PHP.

L'interpréteur PHP pilote un ensemble de process d'exécution système appelés "process enfants". Ces process sont dédiés au traitement des requêtes adressées par 4D. L'exécution des requêtes est synchrone. Pour des raisons d'optimisation, par défaut jusqu'à cinq process enfants peuvent être exécutés simultanément (ce nombre est modifiable via les Propriétés de la base ou la commande FIXER PARAMETRE BASE). Sous Mac OS, ces process sont lancés à la première requête et sont conservés en permanence par l'interpréteur PHP. Sous Windows, 4D crée les process en fonction des besoins et les recycle si nécessaire. 4D prend automatiquement en charge la gestion des process de l'interpréteur PHP fourni par défaut (lancement et fermeture).

Note Si le programme 4D quitte inopinément alors que des process PHP enfants sont encore en activité, vous devrez les supprimer manuellement via la fenêtre de gestion des process du système.

Le schéma suivant illustre l'architecture 4D/PHP de 4D v12 :

Cette architecture fonctionne avec un système de requêtes internes envoyées par 4D à une adresse TCP prédéfinie (adresse IP et numéro de port). Si nécessaire, par exemple si plusieurs interpréteurs PHP sont exécutés sur la même machine, cette adresse peut être modifiée via les Propriétés de 4D (cf. paragraphe "PHP", page 41) ou par programmation (cf. paragraphe FIXER PARAMETRE BASE, Lire parametre base).

Note Si vous lancez deux applications 4D sur la même machine et exécutez sur chacun d'elles des instructions PHP (via la commande PHP Executer, cf. ci-dessous), vous devez impérativement modifier les ports d'écoute de l'interpréteur fastcgi PHP afin qu'ils soient différents pour chaque application. Sinon, l'exécution des instructions PHP pourra échouer de façon aléatoire.

Utiliser un autre interpréteur PHP ou un autre fichier php.ini

Vous pouvez choisir d'utiliser un autre interpréteur PHP que celui fourni par 4D. Un interpréteur PHP personnalisé doit respecter deux conditions :

- il doit être compilé en fastCGI,
- il doit être présent sur la même machine que 4D.

Pour utiliser un interpréteur PHP personnalisé, il vous suffit de le configurer de manière à ce qu'il écoute à une adresse et un port TCP spécifiques et d'indiquer à 4D de ne pas activer l'interpréteur interne. Ces paramètres peuvent être définis soit via les réglages de la base (cf. paragraphe "PHP", page 41), soit pour la session via la commande FIXER PARAMETRE BASE (cf. paragraphe FIXER PARAMETRE BASE, Lire parametre base). Bien entendu, dans ce cas vous devez gérer vous-même le démarrage et le fonctionnement de l'interpréteur.

Vous pouvez également utiliser un fichier de configuration php.ini personnalisé afin, par exemple, de bénéficier de modules additionnels. Le fichier php.ini permet notamment de déclarer l'emplacement des extensions PHP.

Pour cela, il vous suffit de placer un fichier php.ini personnalisé dans le dossier de préférences de la base courante (dossier "Preferences", situé à côté du fichier de structure). Sa présence sera détectée par 4D au lancement de l'interpréteur et il sera utilisé à la place du fichier php.ini par défaut de 4D. Vous devez veiller à ce que le contenu du fichier php.ini personnalisé soit valide.

Attention : Le fichier php.ini doit fixer "display_errors" à "stderr" afin que 4D puisse être informé en cas d'erreur lors de l'exécution de code PHP. Exemple :

```
; stderr - Afficher les erreurs vers STDERR (affecte uniquement les CGI/CLI)
; Pour diriger les erreurs vers STDERR pour les CGI/CLI:
display_errors = "stderr"
```

Pour plus d'informations sur la configuration des fichiers php.ini personnalisés, reportez-vous aux commentaires placés dans le fichier *php.ini* fourni par 4D.

Modules PHP

L'interpréteur PHP fourni avec 4D v12 contient une sélection de modules intégrés, retenus notamment en fonction de leur pertinence et leur actualité.

La liste détaillée des modules livrés avec 4D ainsi que des exemples d'utilisation sont fournis à l'annexe A, "Modules PHP," page 327.

Si vous le souhaitez, vous pouvez installer des modules supplémentaires afin d'ajouter des fonctionnalités dans votre application. Pour plus d'informations sur l'ajout de modules PHP, reportez-vous au paragraphe "Installation de modules supplémentaires", page 333.

Exécuter un script PHP

Pour exécuter un script ou une fonction PHP, vous devez utiliser la nouvelle commande PHP Executer. Cette commande est décrite dans le paragraphe "PHP Executer", page 189.

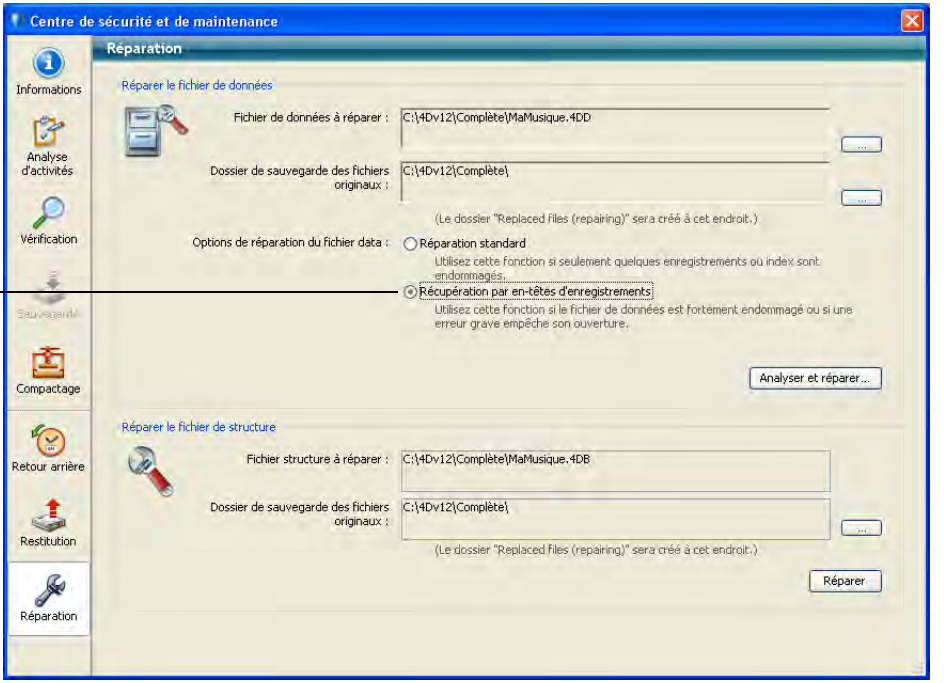
Par défaut, 4D v12 inclut la version 5.3.2 de PHP. Les scripts exécutés doivent être compatibles avec cette version et les modules installés. Pour la liste complète des modules disponibles avec la version PHP fournie par 4D, reportez-vous à l'annexe A, "Modules PHP," page 327.

Pour une description complète des commandes et de la syntaxe PHP, veuillez vous référer à l'abondante documentation PHP disponible sur Internet. A titre d'exemple, voici des adresses de sites de référence :

http://fr.php.net/manual/fr/
http://php.developpez.com/
http://www.php.fr/
http://www.phpfrance.com/

Récupération par marqueurs d'en-tête

La page **Réparation** du Centre de sécurité et de maintenance (CSM) comporte la nouvelle option **Récupération par en-têtes d'enregistrements**, permettant de récupérer les données via les marqueurs d'en-têtes d'enregistrements :



Nouvelle option de récupération

Cette option de réparation de bas niveau est à utiliser uniquement dans le cas où le fichier de données a été fortement endommagé et une fois que toutes les autres solutions (restitution de sauvegarde, réparation standard) se sont avérées inefficaces.

Qu'est-ce que la récupération par marqueur d'en-tête ?

Les enregistrements de 4D sont de taille variable : il est donc nécessaire, pour les retrouver, de conserver dans une table l'endroit où ils sont stockés sur votre disque. Le programme accède donc à l'adresse de l'enregistrement par l'intermédiaire d'un index et d'une table d'adresses.

Si seuls des enregistrements ou des index sont endommagés, l'option de réparation standard suffira généralement pour résoudre le problème. C'est lorsque la table d'adresses est touchée qu'il faudra en venir à une récupération plus sophistiquée, puisqu'il faut la reconstituer. Pour réaliser cette opération, le CSM utilise le marqueur qui se trouve en en-tête de chaque enregistrement. Les marqueurs peuvent être comparés à des résumés des enregistrements, comportant l'essentiel de leurs informations, et à partir desquels une reconstitution de la table d'adresses est possible.

Note Si vous avez désélectionné l'option **Enregistrements définitivement supprimés** dans les propriétés d'une table dans la structure de la base, la réparation par marqueurs d'en-êtes pourra faire réapparaître des enregistrements normalement supprimés.

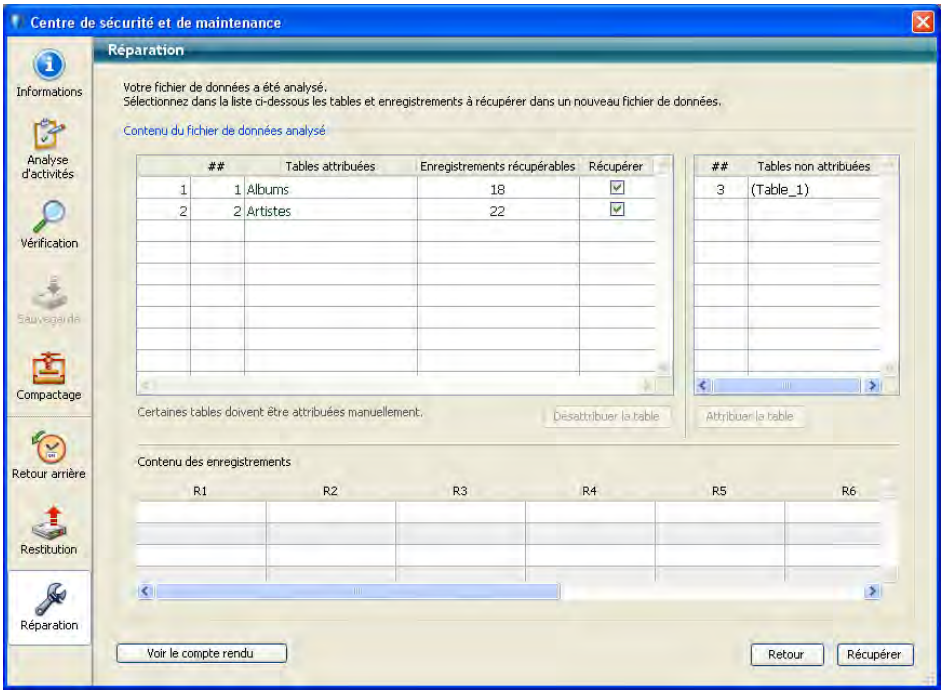
La récupération par en-têtes ne tient pas compte des éventuelles contraintes d'intégrité. En particulier, à l'issue de cette opération, vous pouvez obtenir des valeurs dupliquées avec des champs uniques ou des valeurs NULL avec des champs déclarés non NULL.

Procédure de récupération

Pour effectuer une récupération par en-têtes, affichez la page "Réparation" du CSM, cochez l'option **Récupération par en-têtes d'enregistrements** puis cliquez sur le bouton **Analyser et réparer...**

Note Cette opération nécessite que la base soit ouverte en mode Maintenance. Si ce n'est pas le cas, une boîte de dialogue d'alerte vous informe que la base va être fermée puis rouverte dans ce mode.

Dans la première phase de la procédure de récupération, 4D effectue une analyse complète du fichier de données. A l'issue de cette analyse, le résultat est affiché dans la fenêtre suivante :



Note Si tous les enregistrements et toutes les tables ont été attribués, seule la zone principale est affichée.

La zone "Contenu du fichier de données analysé" comporte deux tableaux synthétisant les informations issues de l'analyse du fichier de données.

Le premier tableau liste les informations issues de l'analyse du fichier de données. Chaque ligne représente un groupe d'enregistrements récupérables dans le fichier de données :

- La première colonne indique l'ordre de récupération des groupes d'enregistrements.
- La colonne **##** indique le numéro de la table attribuée au groupe, si une table a bien été attribuée.
- La colonne **Tables attribuées** indique le nom des tables ayant pu être automatiquement associées aux groupes d'enregistrements identifiés.

Les noms des tables attribuées automatiquement sont affichés en caractères verts.

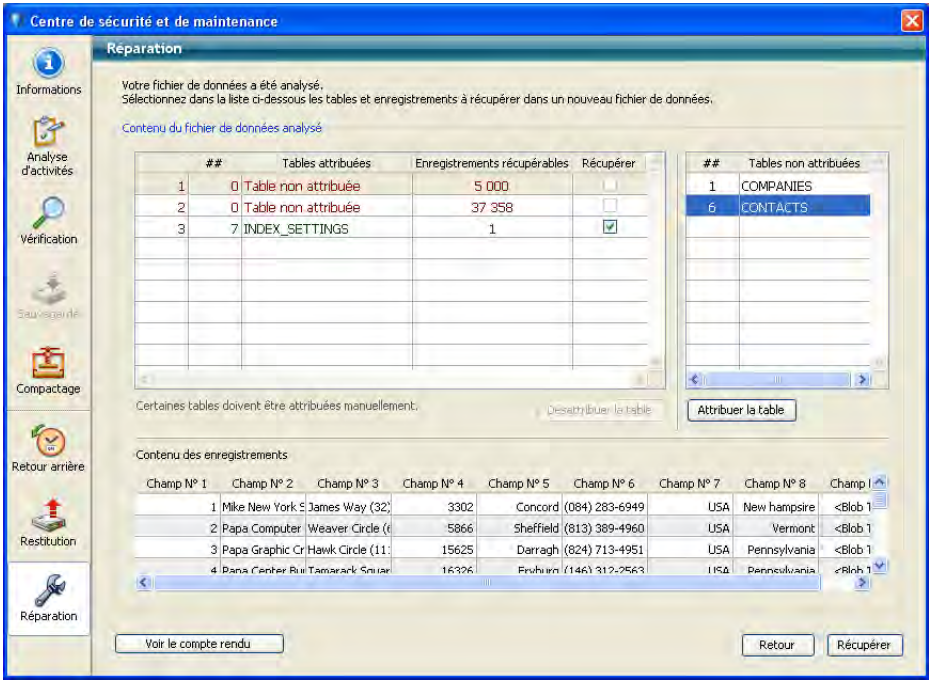
- La colonne **Enregistrements récupérables** indique le nombre d'enregistrements récupérables dans le groupe.
- La colonne **Récupérer** permet vous permet d'indiquer pour chaque groupe si vous souhaitez récupérer les enregistrements. Par défaut, l'option est cochée pour tous les groupes.

Le deuxième tableau liste les tables non atttribuées, c'est-à-dire les tables auxquelles aucun enregistrement n'a pu être rattaché.

Attribution manuelle

Si, du fait de l'endommagement de la table d'adressses, un ou plusieurs groupes d'enregistrements n'ont pas pu être attribués à des tables, vous pouvez les attribuer manuellement.

Pour attribuer une table à un groupe non identifié, sélectionnez le groupe dans le premier tableau. Lorsque vous sélectionnez des enregistrements non identifiés, la zone "Contenu des enregistrements" affiche une prévisualisation du contenu des premiers enregistrements du groupe afin de vous permettre de les attribuer plus facilement :



Sélectionnez ensuite la table à attribuer dans le tableau des "Tables non identifiées" puis cliquez sur le bouton **Attribuer la table**. Vous pouvez également attribuer une table par glisser-déposer.

Le groupe d'enregistrements est alors associé à la table, il sera récupéré dans cette table. Les noms des tables attribuées manuellement sont affichés en caractères noirs.

Le bouton **Désattribuer la table** permet de supprimer l'association effectuée manuellement entre une table et un groupe d'enregistrements.

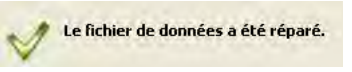
Lancer la récupération

Une fois que vous avez configuré les éléments à récupérer, cliquez sur le bouton **Récupérer** afin de lancer la réparation.

Note Vous pouvez cliquer sur le bouton **Retour** pour annuler la procédure et retourner à l'état précédent.

4D crée un nouveau fichier de données vide à l'emplacement du fichier d'origine. Le fichier d'origine est déplacé dans le dossier nommé "\Replaced Files (Repairing) *date heure*" dont l'emplacement a été défini dans la page "Réparation" du CSM (dossier de la base par défaut). Le fichier vide est rempli avec les données récupérées.

A l'issue de la procédure, la page "Réparation" du CSM est affichée. Un message indique si la réparation a été effectuée avec succès :



Dans ce cas, vous pouvez immédiatement ouvrir la base.

Comme pour toutes les fonctions d'analyse du CSM, le bouton **Voir le compte-rendu** affiche dans votre navigateur une page Web détaillant le résultat de l'opération. La page liste l'ensemble des vérifications ou réparations effectuées et signale chaque erreur rencontrée, le cas échéant ([OK] est affiché lorsque la vérification est correcte). Le fichier est généré dans le dossier **Logs** de la base. Il est créé au format xml et html et est nommé "NomBase_Repair_log".

4 Formulaires et objets

4D v12 propose de nombreuses nouveautés relatives aux formulaires et aux objets qu'ils contiennent. Ces nouveautés facilitent et accélèrent le développement des interfaces de vos applications.

Extension des capacités des sous-formulaires

4D v12 augmente les capacités des sous-formulaires (aussi appelés formulaires inclus). Plus souples d'utilisation, programmables, utilisables en composant, ces objets de nouvelle génération sont désormais de puissants outils pour le développement de fonctions avancées et facilement portables.

Terminologie

Afin de bien définir les notions mises en oeuvre avec les sous-formulaires, voici quelques définitions relatives aux termes employés :

- **sous-formulaire** : formulaire destiné à être inclus dans un autre formulaire, lui-même nommé formulaire parent.
- **formulaire parent** : formulaire contenant un ou plusieurs sous-formulaire(s).
- **conteneur de sous-formulaire** : objet inclus dans le formulaire parent, contenant une instance du sous-formulaire.
- **instance de sous-formulaire** : la représentation d'un sous-formulaire dans un formulaire parent. Cette notion est importante car il est possible d'afficher plusieurs instances d'un même sous-formulaire dans un formulaire parent.

Nouvelles propriétés

Des propriétés supplémentaires peuvent désormais être associées via la Liste des propriétés aux objets sous-formulaires installés dans les formulaires parents :

- **Nom de la variable** : une variable peut désormais être liée à un objet sous-formulaire. Par défaut, cette variable est nommée "Sous formulaire", comme l'objet lui-même. Elle a un type (cf. paragraphe suivant) et peut être représentée sous la forme d'une variable standard dans le formulaire parent. Sa modification déclenche des événements formulaire et permet ainsi de synchroniser les valeurs du formulaire parent et du sous-formulaire :
  - L'événement formulaire Sur données modifiées permet d'indiquer au conteneur de sous-formulaire que la valeur de la variable a été modifiée côté sous-formulaire.
  - Le nouvel événement formulaire Sur modif variable liée permet d'indiquer au sous-formulaire (méthode formulaire du sous-formulaire) que la variable a été modifiée côté formulaire parent (cf. paragraphe "Mise à jour du contenu du sous-formulaire", page 77).
- **Type de la variable** : permet de définir le type de la variable associée à l'objet sous-formulaire. Par défaut, le type Alpha est utilisé. Le type de la variable détermine la nature des valeurs échangées entre le formulaire parent et le sous-formulaire via la variable associée.
  Vous pouvez définir le type "Aucun", dans ce cas 4D typera automatiquement la variable lors de l'exécution (cf. paragraphe "Variables dynamiques", page 256).
- **Source** : dans les versions précédentes de 4D, cette propriété était libellée "Table source". Elle permet désormais de sélectionner différents types de sources :
  - **<Aucun>** : choisissez ce type de source si vous souhaitez utiliser un formulaire projet ou un formulaire de composant comme sous-formulaire. Ces sous-formulaires sont utilisables en mode page uniquement : l'option "Sous-formulaire liste" doit être désélectionnée pour qu'ils puissent fonctionner.
    Pour qu'un formulaire de composant apparaisse dans la liste "Formulaire détaillé", il doit avoir été publié dans le composant (cf. paragraphe "Publier un sous-formulaire (composant)", page 81).
  - ***Nom de table*** : choisissez ce type de source si vous souhaitez utiliser des formulaires de la table (équivaut au fonctionnement précédent de 4D).

Note La propriété d'apparence "Plate-forme" disparaît, l'apparence est définie par le sous-formulaire lui-même et ne peut pas être modifiée.

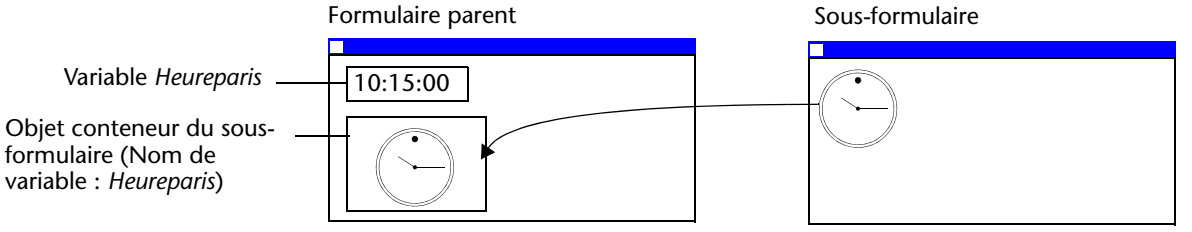
- **Méthode** : un objet sous-formulaire peut désormais comporter une méthode objet, permettant de contrôler son fonctionnement, notamment lors du déclenchement d'événements (cf. paragraphe suivant). A noter que l'interaction entre le contenu de l'objet sous-formulaire et le formulaire parent doit être gérée par des mécanismes spécifiques (cf. paragraphe "Programmation inter-formulaires avancée", page 78).
- **Evénements formulaire** : un objet sous-formulaire accepte les événements suivants :
  - Sur chargement et Sur libération : respectivement ouverture et fermeture du sous-formulaire (ces événements doivent également avoir été activés au niveau du formulaire parent pour être pris en compte). A noter que ces événements sont générés avant ceux du formulaire parent.
    A noter également que, conformément aux principes de fonctionnement des événements formulaire, si le sous-formulaire est placé sur une page autre que la page 0 ou 1, ces événements ne sont générés qu'au moment de l'affichage/la fermeture de la page (et non du formulaire).
  - Sur validation : validation de la saisie dans le sous-formulaire.
  - Sur données modifiées : la valeur de la variable de l'objet sous-formulaire a été modifiée.
  - Sur gain focus et Sur perte focus : le conteneur de sous-formulaire vient de recevoir ou de perdre le focus. Ces événements sont générés dans la méthode de l'objet sous-formulaire lorsqu'ils sont cochés. Ils sont envoyés à la méthode formulaire du sous-formulaire, ce qui permet par exemple de gérer l'affichage de boutons de navigation dans le sous-formulaire en fonction du focus.
    A noter que les objets du sous-formulaire peuvent eux-mêmes recevoir le focus.

Notes - Il est possible de définir tout type d'événement personnalisé pouvant être généré dans un sous-formulaire via la nouvelle commande APPELER CONTENEUR SOUS FORMULAIRE. Cette commande permet d'appeler la méthode de l'objet conteneur et de lui passer un code d'événement.

- Un nouvel événement formulaire, Sur modif variable liée, peut être géré au niveau du sous-formulaire (cf. paragraphe “Mise à jour du contenu du sous-formulaire”, page 77).
- Lorsque la commande FIXER MINUTEUR est exécutée dans le contexte d’un sous-formulaire (méthode formulaire du sous-formulaire), l’événement Sur minuteur est désormais généré dans le sous-formulaire et non au niveau du formulaire parent (fonctionnement des versions précédentes de 4D). Ce nouveau principe permet notamment de générer des événements Sur minuteur avec des intervalles différents dans chaque sous-formulaire et dans le formulaire parent.

Gestion de la variable liée

La variable liée au sous-formulaire permet de relier les deux contextes (formulaire et sous-formulaire) pour mettre au point des interfaces sophistiquées. Imaginons par exemple un sous-formulaire représentant une pendule dynamique, inséré dans un formulaire parent contenant une variable saisissable de type heure :



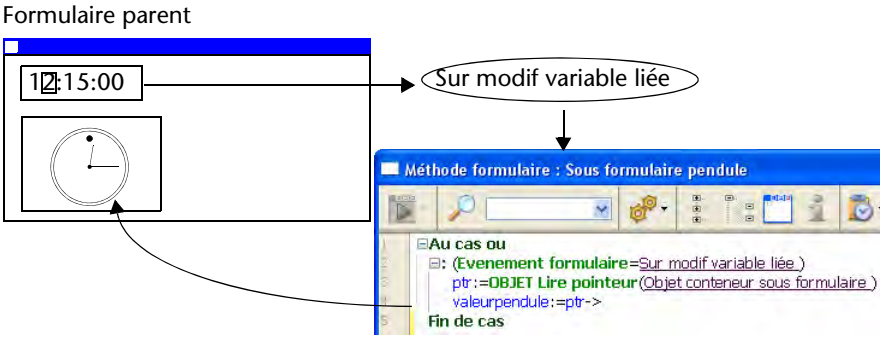
Les deux objets (variable heure et conteneur du sous-formulaire) ont le même nom de variable. Dans ce cas, à l’ouverture du formulaire parent, les deux valeurs sont automatiquement synchronisées par 4D. Si la valeur de la variable est définie à plusieurs emplacements, la valeur utilisée sera celle qui aura été chargée en dernier lieu. L’ordre de chargement suivant est appliqué :
1-Méthodes objet du sous-formulaire
2-Méthode formulaire du sous-formulaire
3-Méthodes objet du formulaire parent
4-Méthode formulaire du formulaire parent

A l’exécution du formulaire parent, la synchronisation des variables doit être effectuée par le développeur à l’aide des événements formulaires adéquats. Deux types d’interactions peuvent se produire : du formulaire vers le sous-formulaire et inversement.

Mise à jour du contenu du sous-formulaire

La valeur de la variable du formulaire parent est modifiée et cette modification doit être répercutée dans le sous-formulaire. Dans notre exemple, l'heure de *Heureparis* passe à 12:15:00, soit parce que l'utilisateur l'a saisie, soit parce qu'elle est mise à jour dynamiquement (via la commande Heure courante par exemple).

Dans ce cas, vous devez utiliser le nouvel événement formulaire **Sur modif variable liée**. Cet événement doit être coché dans les propriétés du sous-formulaire, il sera généré dans la méthode formulaire du sous-formulaire.



L'événement formulaire **Sur modif variable liée** est généré :

- dès qu'une valeur est affectée à la variable du formulaire parent, même si la même valeur est réaffectée,
- si le sous-formulaire appartient à la page formulaire courante ou à la page 0.

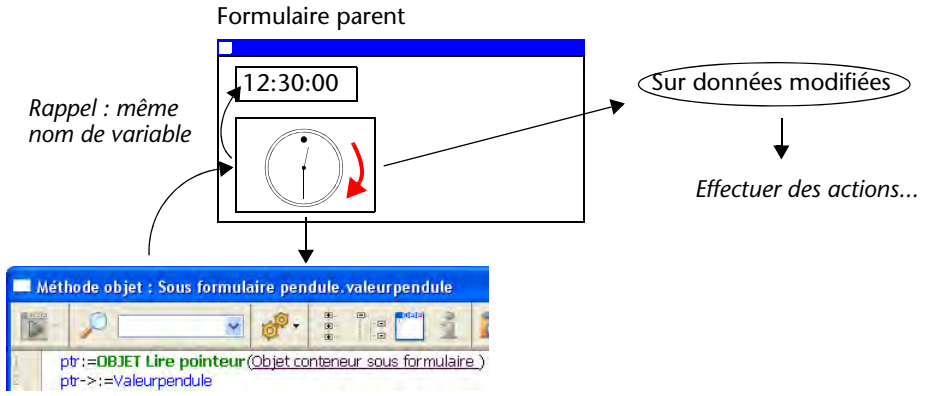
A noter que, comme dans l'exemple ci-dessus, il est préférable d'utiliser la commande OBJET Lire pointeur qui retourne un pointeur vers le conteneur de sous-formulaire plutôt que sa variable car il est possible d'insérer plusieurs sous-formulaires dans un même formulaire parent (par exemple, une fenêtre affichant des fuseaux horaires contiendrait plusieurs pendules). Dans ce cas, seul un pointeur permet de connaître le conteneur de sous-formulaire à l'origine de l'événement.

Mise à jour du contenu du formulaire parent

Le contenu du sous-formulaire est modifié et cette modification doit être répercutée dans le formulaire parent. Dans notre exemple, imaginons que l'interface du sous-formulaire permette à l'utilisateur de déplacer "manuellement" les aiguilles.

Dans ce cas, vous devez affecter la valeur de l'objet à la variable du conteneur du sous-formulaire parent depuis le sous-formulaire.

Comme dans l'exemple précédent, il est conseillé d'utiliser pour cela la commande OBJET Lire pointeur avec le sélecteur Objet conteneur sous formulaire qui retourne un pointeur vers le conteneur du sous-formulaire.
Cette affectation génère l'événement formulaire **Sur données modifiées** dans la méthode de l'objet conteneur du sous-formulaire parent, ce qui vous permet d'effectuer tout type d'action. L'événement doit être coché dans les propriétés du conteneur de sous-formulaire.



Méthode objet de la variable *valeurPendule* dans le sous-formulaire

Note Bien entendu, la modification "manuelle" des aiguilles de la pendule génère également l'événement formulaire **Sur données modifiées** dans la méthode de l'objet de la variable *valeurPendule*, dans le sous-formulaire.

Programmation inter-formulaires avancée

La communication entre le formulaire parent et les instances des sous-formulaires peut nécessiter d'aller au-delà de l'échange d'une valeur via la variable associée. En effet, vous pouvez souhaiter mettre à jour des variables dans les sous-formulaires en fonction d'actions effectuées dans le formulaire parent et inversement. Si l'on reprend l'exemple du sous-formulaire de type "pendule dynamique", on peut souhaiter définir une ou plusieurs heures d'alerte par pendule.

Pour répondre à ces besoins, de nouveaux mécanismes ont été mis en place dans 4D. Ces mécanismes sont les suivants :

- utilisation du paramètre "sous-formulaire" avec la commande OBJET Lire pointeur afin de désigner l'objet sous-formulaire et nouvelle commande OBJET Lire nom.

- appel de l'objet conteneur depuis le sous-formulaire via la nouvelle commande APPELER CONTENEUR SOUS FORMULAIRE,
- exécution d'une méthode dans le contexte du sous-formulaire via la nouvelle commande EXECUTER METHODE DANS SOUS FORMULAIRE.

Commandes OBJET Lire pointeur et OBJET Lire nom

Outre le sélecteur "Objet conteneur sous formulaire" (cf. paragraphe "Mise à jour du contenu du sous-formulaire", page 77), la commande OBJET Lire pointeur admet un paramètre permettant de préciser dans quel sous-formulaire chercher l'objet dont le nom est passé en deuxième paramètre. Cette syntaxe n'est utilisable que lorsque le sélecteur "Objet nommé" est passé.

Par exemple, l'instruction suivante :
$ptr:=**OBJET Lire pointeur**(Objet nommé;"MonBouton";"MonSousForm")

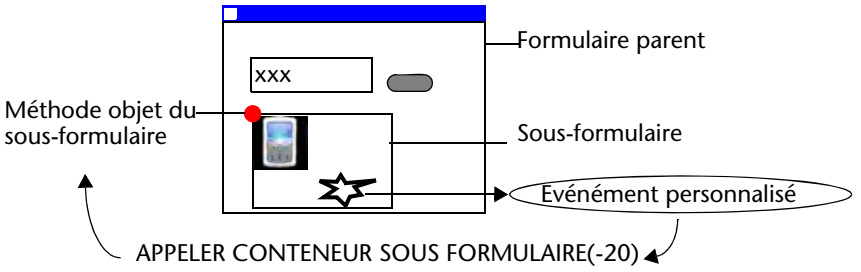
... permet de récupérer un pointeur sur la variable "MonBouton" se trouvant dans l'objet sous-formulaire "MonSousForm". Cette syntaxe permet d'accéder depuis le formulaire parent à tout objet se trouvant dans un sous-formulaire. Pour une description complète de cette commande, reportez-vous à la commande OBJET Lire pointeur, page 163.

Note A noter également la nouvelle commande OBJET Lire nom qui permet de récupérer le nom de l'objet ayant le focus.

Nouvelle commande APPELER CONTENEUR SOUS FORMULAIRE

La nouvelle commande APPELER CONTENEUR SOUS FORMULAIRE permet à une instance de sous-formulaire d'envoyer un événement à l'objet conteneur du sous-formulaire, qui peut alors le traiter dans le contexte du formulaire parent. L'événement est reçu dans la méthode de l'objet conteneur. Il peut s'agir à l'origine de tout événement détecté par le sous-formulaire (clic, glisser-déposer, etc.). Ce mécanisme est illustré dans ce schéma :

Exemple d'exécution de la commande APPELER CONTENEUR SOUS FORMULAIRE



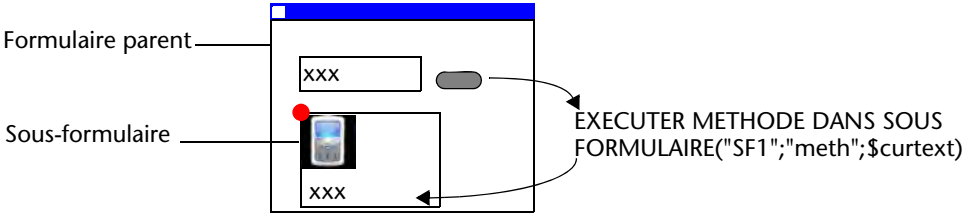
Le code de l'événement est libre (par exemple, 20000 ou -100). Il ne doit correspondre à aucun événement formulaire existant. Il est conseillé d'utiliser une valeur négative pour avoir l'assurance que 4D n'utilisera pas ce code dans les versions futures.
Vous pouvez également utiliser un code correspondant à un événement existant (par exemple, 3 pour Sur validation), mais l'événement doit être autorisé et coché pour le conteneur de sous-formulaire.

Pour une description complète de cette commande, reportez-vous au paragraphe "APPELER CONTENEUR SOUS FORMULAIRE", page 141.

Nouvelle commande EXECUTER METHODE DANS SOUS FORMULAIRE

Cette nouvelle commande permet à un formulaire ou à l'un de ses objets de demander l'exécution d'une méthode dans le contexte de l'instance du sous-formulaire, ce qui lui donne accès aux variables, objets, etc., du sous-formulaire. Cette méthode peut en outre recevoir des paramètres. Ce mécanisme est illustré dans ce schéma :

Exemple d'exécution de la commande EXECUTER METHODE DANS SOUS FORMULAIRE



Note Si le sous-formulaire provient d'un composant, la méthode doit disposer de la propriété "Partagée entre les composants et la base hôte".

Pour une description complète de cette commande, reportez-vous au paragraphe "EXECUTER METHODE DANS SOUS FORMULAIRE", page 170.

Commande ALLER A OBJET (anciennement ALLER A CHAMP)

La commande ALLER A CHAMP a été renommée ALLER A OBJET et peut désormais rechercher l'objet de destination dans le formulaire parent même si elle exécutée depuis un sous-formulaire (cf. paragraphe "ALLER A OBJET", page 144).

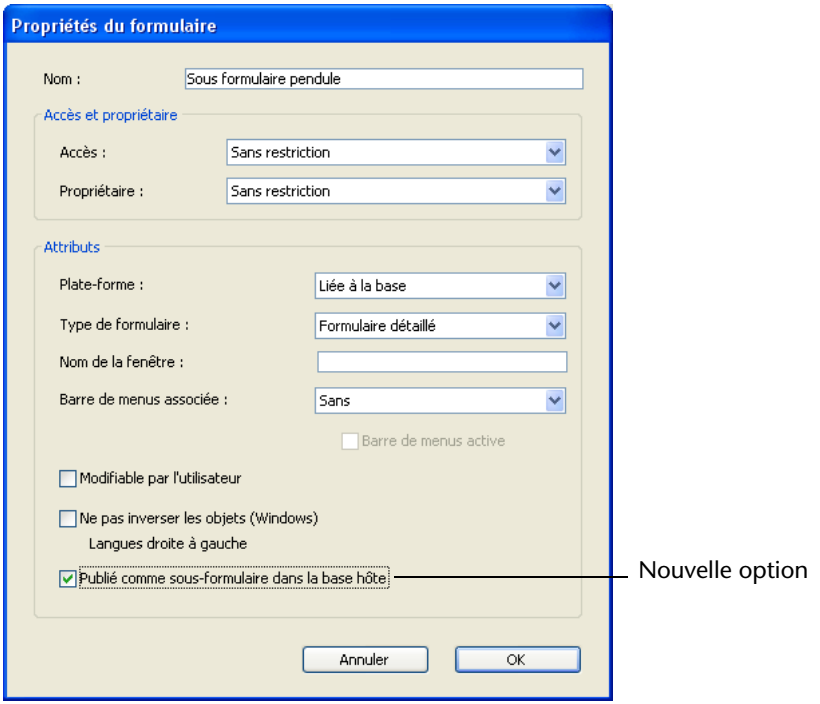
Sous-formulaires en composants

4D v12 permet de publier des formulaires de composants comme sous-formulaires dans les bases hôtes. Avec ce principe, vous pouvez notamment développer des composants proposant des objets graphiques. A noter que les nouveaux objets "widgets" proposés par 4D sont basés sur l'emploi de sous-formulaires en composants (cf. paragraphe “Widgets”, page 84).

Publier un sous-formulaire (composant)

Côté composant (base matrice), seuls les sous-formulaires projet peuvent être désignés comme sous-formulaires publiés.

Pour qu'un formulaire de composant puisse être sélectionné comme sous-formulaire dans une base hôte, il doit avoir été explicitement désigné comme "formulaire publié" dans la boîte de dialogue des propriétés du formulaire via la nouvelle option **Publié comme sous-formulaire dans la base hôte** :



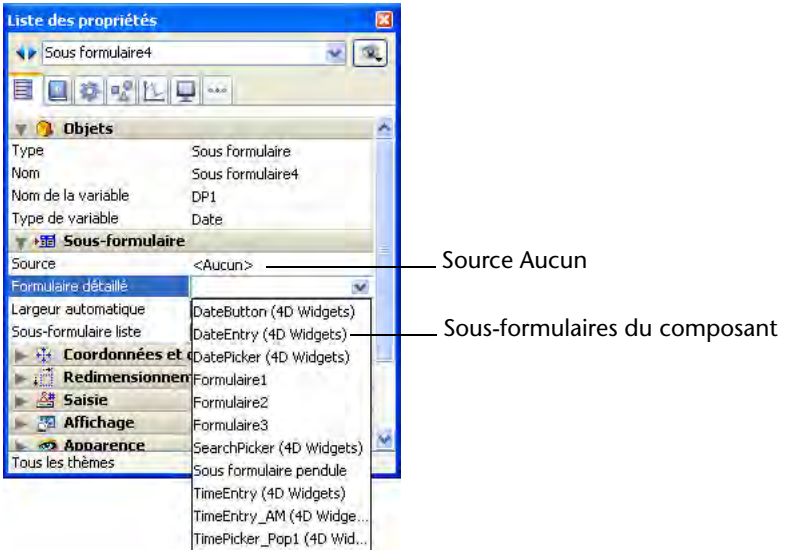
Note Cette boîte de dialogue est accessible via la commande **Propriétés du formulaire** du menu contextuel ou du menu d'action de l'Explorateur.

Vous devez gérer les interactions entre le sous-formulaire et les formulaires parents de la base hôte à l'aide des mécanismes et outils décrits dans les paragraphes précédents.

Utilisation d'un sous-formulaire de composant (base hôte)

Côté base hôte, les sous-formulaires provenant de composants doivent impérativement être utilisés en mode page : dans l'éditeur de formulaires, lorsque l'objet sous-formulaire est sélectionné dans le formulaire parent, désélectionnez l'option **Sous-formulaire liste** dans le thème "Sous-formulaire" de la Liste des propriétés.

Ensuite, choisissez **<Aucun>** dans le menu "Table". Les formulaires publiés par les composants sont alors listés dans le menu "Formulaire détaillé". Leur nom est suivi de celui du composant entre parenthèses. Il vous suffit alors de choisir dans cette liste le formulaire à utiliser dans la liste "Formulaire détaillé".



Une nouvelle instance de l'objet sélectionné est alors immédiatement créée dans le formulaire.

Bibliothèque d'objets préconfigurés

La bibliothèque d'objets préconfigurés de 4D v12 est un nouvel outil conçu pour faciliter l'ajout d'objets dans les formulaires 4D. Elle propose une collection d'objets préconfigurés utilisables dans vos formulaires par simple glisser-déposer ou copier-coller.

Objets de la bibliothèque

Cette bibliothèque utilise exclusivement des objets 4D standard (boutons, zones de texte, etc.) dont certaines propriétés ont été prédéfinies afin d'accélérer et faciliter leur mise en oeuvre.

Par exemple, l'objet "zone de saisie de mot de passe" est une variable texte associée à une feuille de style spécifique. La bibliothèque propose également des objets de haut niveau tels que les *widgets* datepicker et timepicker (cf. paragraphe "Widgets", page 84).

Note A la différence des bibliothèques d'objets utilisateur, la bibliothèque d'objets préconfigurés de 4D v12 n'est pas modifiable : vous ne pouvez ni ajouter ni supprimer d'objets dans cette bibliothèque.

Utilisation de la bibliothèque

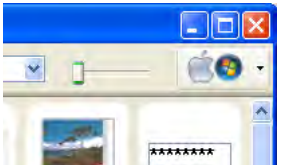
La bibliothèque d'objets préconfigurés se présente sous la forme d'une fenêtre indépendante. L'insertion des objets dans l'éditeur de formulaires s'effectue par simple glisser-déposer.

Pour afficher la fenêtre de la bibliothèque d'objets préconfigurés, il suffit de cliquer sur le dernier bouton de la barre d'outils de l'éditeur de formulaires de 4D :

La bibliothèque d'objets préconfigurés s'affiche alors dans une nouvelle fenêtre. Elle comporte un pop up menu, une zone de prévisualisation et une zone de commentaires :

Les objets sont classés en différentes catégories (boutons, zones de saisie, etc.). Pour restreindre la sélection à une catégorie d'objets, sélectionnez-la dans le pop up menu ou choisissez la ligne **Toutes les catégories** afin d'afficher tous les objets.

Certains objets sont liés à une plate-forme (Windows ou Macintosh). Vous pouvez filtrer les objets affichés en fonction de la plate-forme à l'aide du menu situé en haut de la fenêtre.



La zone centrale affiche une prévisualisation ainsi que le nom de chaque objet. Pour obtenir des informations sur un objet, cliquez dessus : la description de l'objet s'affiche dans la zone inférieure de la fenêtre.

L'insertion d'un objet dans un formulaire s'effectue par simple glisser-déposer ou copier-coller depuis la zone centrale de la fenêtre vers le formulaire. L'objet est inséré avec ses propriétés prédéfinies. Vous pouvez alors les modifier afin d'adapter l'objet à vos besoins.

Widgets

4D v12 vous permet de bénéficier d'objets composés prédéfinis. Utilisables avec ou sans programmation, ces *widgets* donnent accès à des fonctionnalités standard très simples à mettre en oeuvre. Les nouveaux objets disponibles sont les suivants :

- **SearchPicker** : zone de recherche d'apparence standard.
- **DatePicker** : sélecteur de date.
- **TimePicker** : sélecteur d'heure.

Note Les widgets sont également accessibles via la nouvelle bibliothèque d'objets intégrée de 4D (cf. paragraphe "Bibliothèque d'objets préconfigurés", page 82).

SearchPicker

Le widget *SearchPicker* vous permet de créer facilement des zones standard de recherche, semblables à celles que l'on peut trouver dans les navigateurs ou les barres d'outils. L'apparence de la zone dépend de la plate-forme.

Windows

Mac Os

Le texte affiché par défaut dans la zone peut être contrôlé par programmation, à l'aide d'une méthode composant décrite dans le paragraphe "SearchPicker", page 268.

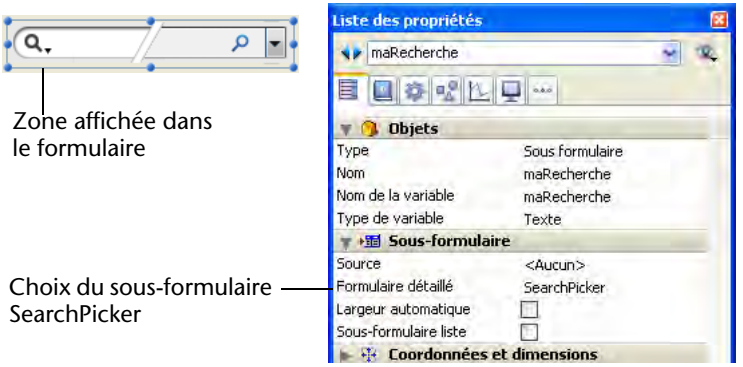
Fonctionnement

Hormis son apparence, une zone de recherche *SearchPicker* se caractérise par les éléments suivants : un texte grisé, une zone de saisie et une icône d'effacement.

- La zone de saisie permet à l'utilisateur de saisir la valeur à rechercher. Cette valeur est automatiquement et dynamiquement affectée à la variable que vous avez associée à la zone dans la Liste des propriétés (propriété **Nom de la variable**). Vous utilisez cette variable pour fournir la valeur recherchée à la méthode de recherche.
- Le texte grisé est une aide indiquant à l'utilisateur le ou les champ(s) où la recherche sera effectuée. Il disparaît dès que la zone a le focus. Ce texte peut être défini via la commande SearchPicker SET HELP TEXT.
- Le bouton d'effacement permet d'effacer le contenu de la zone. Son fonctionnement est automatique.
- En exécution, vous pouvez lancer votre méthode de recherche via un clic sur un bouton ou dans un événement formulaire. La zone génère notamment les événements Sur données modifiées et Sur perte focus. Vous pouvez proposer une recherche dynamique réévaluée à chaque caractère saisi par l'utilisateur en appelant la méthode de recherche dans l'événement Sur données modifiées. Vous pouvez également lancer la recherche lorsque l'utilisateur appuie sur la touche **Entrée** : dans ce cas il suffit d'appeler la méthode de recherche dans l'événement Sur perte focus.

Création

Vous pouvez insérer une zone SearchPicker dans un formulaire à l'aide de la nouvelle bibliothèque d'objets intégrée de 4D (cf. paragraphe "Bibliothèque d'objets préconfigurés", page 82).
Vous pouvez également créer une zone de sous-formulaire et lui affecter le formulaire détaillé de composant *SearchPicker*, comme décrit dans le paragraphe "Utilisation d'un sous-formulaire de composant (base hôte)", page 82.



Définissez ensuite le nom de la variable associée au sous-formulaire (propriété **Nom de la variable** dans la Liste des propriétés). A l'exécution du formulaire, cette variable contiendra automatiquement la valeur recherchée par l'utilisateur. Il vous suffit alors de passer cette valeur à votre méthode de recherche personnalisée.

DatePicker

Le *widget* DatePicker est un objet intuitif et très simple d'emploi que vous pouvez utiliser pour valoriser tout champ nécessitant la saisie d'une date ou simplement pour représenter une date.

Ce widget est proposé sous deux formes :

- **Calendrier DatePicker** : cet objet est utilisable soit dans un sous-formulaire, soit en tant que calendrier déroulant affiché par un clic sur un bouton.
- **Zone DateEntry** : zone de date associée à des boutons de contrôle. Cet objet est utilisable dans un sous-formulaire uniquement.

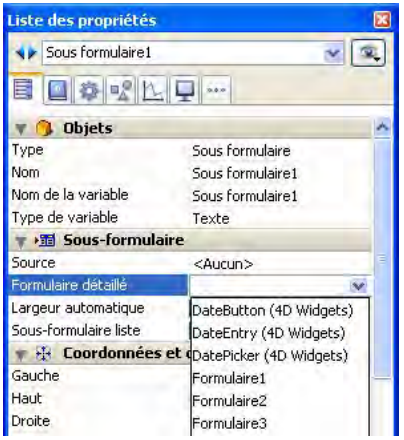
Calendrier DatePicker

En exécution, l'utilisateur peut faire défiler les mois du calendrier vers l'avant ou vers l'arrière en cliquant sur des boutons en forme de flèche. Il peut également utiliser les touches fléchées du clavier.

Lorsqu'il est inséré dans un sous-formulaire, un objet DatePicker peut être utilisé sans programmation grâce au mécanisme fourni par la variable associée (cf. ci-dessous). Toutefois, si vous souhaitez personnaliser le fonctionnement du Datepicker ou l'afficher sous forme de pop up menu, vous devez utiliser l'ensemble de méthodes composant qui est fourni. Ces méthodes composant sont décrites dans le paragraphe "DatePicker", page 259.

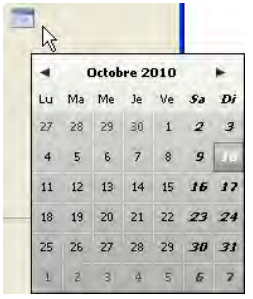
- **Utilisation en sous-formulaire**

Vous pouvez insérer un calendrier DatePicker dans un formulaire à l'aide de la nouvelle bibliothèque d'objets intégrée de 4D (cf. paragraphe "Bibliothèque d'objets préconfigurés", page 82) ou en créant une zone de sous-formulaire et en lui affectant le formulaire de composant *DatePicker*, comme décrit dans le paragraphe "Utilisation d'un sous-formulaire de composant (base hôte)", page 82.



Définissez ensuite le nom de la variable associée au sous-formulaire (propriété **Nom de la variable** dans la Liste des propriétés). A l'exécution du formulaire, cette variable date contiendra automatiquement la date sélectionnée par l'utilisateur. A l'inverse, si vous modifiez par programmation la valeur de cette variable, elle sera automatiquement représentée dans le sous-formulaire.

- **Utilisation dans un pop up**
  Vous pouvez utiliser un calendrier DatePicker sous forme de fenêtre pop up. Pour cela, il vous suffit d'appeler la méthode composant DatePicker Display Dialog. La méthode retourne la date sélectionnée par l'utilisateur.



Les méthodes du composant 4DDatePicker sont décrites dans le paragraphe "DatePicker", page 259.

Zone DateEntry

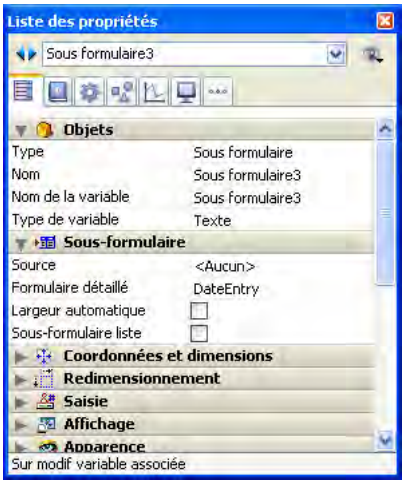
Une zone de type DateEntry facilite la saisie d'une date sous la forme définie dans les préférences système (par exemple JJ/MM/AA).

La zone se présente sous la forme d'une date de type associée à des boutons :

En exécution, les boutons situés à droite de la zone de saisie ne s'affichent que lorsque l'objet a le focus. L'utilisateur sélectionne individuellement chaque élément de la date (jour, mois ou année) via un clic ou la touche Tabulation et peut le faire défiler en utilisant le *stepper* numérique ou les touches fléchées du clavier. L'icône calendrier situé à droite permet de sélectionner une date par l'intermédiaire d'un pop up calendrier DatePicker.

Un objet DateEntry peut être utilisé sans programmation grâce au mécanisme fourni par la variable associée (cf. ci-dessous). Toutefois, si vous souhaitez en personnaliser le fonctionnement, vous pouvez utiliser l'ensemble de méthodes composant qui est fourni. Ces méthodes sont les mêmes que celles de l'objet DatePicker. Pour plus d'informations, reportez-vous au paragraphe "DatePicker", page 259.

- **Utilisation en sous-formulaire**
  Vous pouvez insérer une zone DateEntry dans un formulaire à l'aide de la nouvelle bibliothèque d'objets intégrée de 4D (cf. paragraphe "Bibliothèque d'objets préconfigurés", page 82). Vous pouvez également créer une zone de sous-formulaire et lui affecter le formulaire de composant *DateEntry*, comme décrit dans le paragraphe "Utilisation d'un sous-formulaire de composant (base hôte)", page 82.

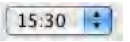


Définissez ensuite le nom de la variable associée au sous-formulaire (propriété **Nom de la variable** dans la Liste des propriétés).
A l'exécution du formulaire, cette variable contiendra automatiquement la date sélectionnée par l'utilisateur. A l'inverse, si vous modifiez par programmation la valeur de cette variable, elle sera automatiquement représentée dans le sous-formulaire.

TimePicker

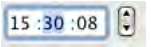
Le *widget* TimePicker propose des objets simples d'emploi que vous pouvez utiliser pour valoriser des champs nécessitant la saisie d'heures ou afficher des heures.

Il est utilisable soit sous la forme de pop up menus (simples ou doubles), soit sous la forme de zones de saisie d'heures au format "hh:mm:ss" associées à un stepper numérique permettant d'augmenter ou de réduire les valeurs d'heures, minutes ou secondes.



Pop up simple

Pop up double



Zone de saisie d'heure

En outre, chaque type de TimePicker peut afficher l'heure sur 12 heures (AM-PM) ou 24 heures.

Un objet TimePicker peut être utilisé sans programmation grâce au mécanisme fourni par la variable associée (cf. ci-dessous). Toutefois, si vous souhaitez personnaliser le fonctionnement des objets TimePicker, vous devez utiliser l'ensemble de méthodes composant qui est fourni. Ces méthodes composant sont décrites dans le paragraphe "TimePicker", page 269.

- **Utilisation**

  Une zone TimePicker est un sous-formulaire que vous pouvez insérer dans un formulaire à l'aide de la nouvelle bibliothèque d'objets intégrée de 4D (cf. paragraphe "Bibliothèque d'objets préconfigurés", page 82). Vous pouvez également créer une zone de sous-formulaire et lui affecter le formulaire de composant *TimePicker*, comme décrit dans le paragraphe "Utilisation d'un sous-formulaire de composant (base hôte)", page 82. Définissez ensuite le nom de la variable heure associée au sous-formulaire (propriété **Nom de la variable** dans la Liste des propriétés). A l'exécution du formulaire, cette variable contiendra automatiquement l'heure définie par l'utilisateur. A l'inverse, si vous modifiez par programmation la valeur de cette variable, elle sera automatiquement représentée dans le sous-formulaire.

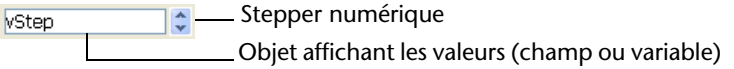
  Les méthodes du composant 4DTimePicker sont décrites dans le paragraphe "TimePicker", page 269.

Nouveaux objets de formulaire

Cette section liste les nouveautés relatives aux objets standard utilisables dans le formulaires de 4D.

Nouvel objet stepper

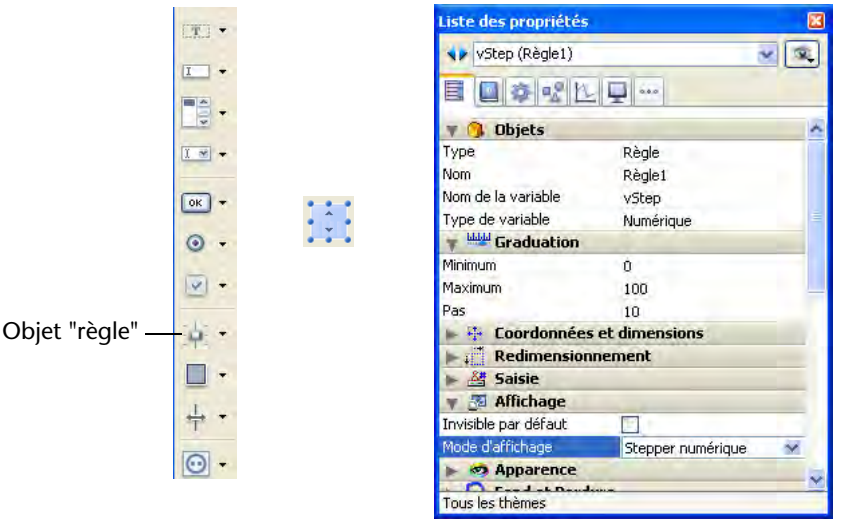
4D v12 propose un nouvel objet de formulaire : le "stepper". Cet objet standard permet à l'utilisateur de faire défiler des valeurs numériques, des durées (heures) ou des dates par paliers prédéfinis en cliquant sur des boutons en forme de flèches :



Le fonctionnement de cet objet est proche de celui de l'objet existant règle. La variable associée à l'objet peut être affectée à une zone saisissable (champ ou variable) afin de stocker ou modifier la valeur courante de l'objet.

L'objet stepper est une variante d'affichage de l'objet règle.

Pour ajouter un stepper dans un formulaire, vous devez dessiner un objet règle puis sélectionner **Stepper numérique** dans le menu "Mode d'affichage" de la Liste des propriétés :



Vous pouvez définir les propriétés Minimum, Maximum et Pas pour les steppers. Les autres propriétés, en particulier les événements formulaire, fonctionnent de la même manière que pour tous les objets jauges.

Un stepper peut être directement associé à une variable numérique, heure ou date.

- Pour les valeurs de type heure, les propriétés Minimum, Maximum et Pas représentent des secondes. Par exemple, pour définir un stepper de 8h00 à 18h00 avec des pas de 10 minutes :
  - Minimum = 28 800 (8*60*60)
  - Maximum = 64 800 (18*60*60)
  - Pas = 600 (10*60)
- Pour les valeurs de type date, la valeur saisie dans la propriété Pas représente des jours. Les propriétés Minimum et Maximum sont ignorées.

Pour que le stepper fonctionne avec une variable heure ou date, il est impératif de définir son type dans la Liste de propriétés ET de la déclarer explicitement via la commande C_HEURE ou C_DATE.

Note Il est possible d'activer l'affichage "Stepper" à l'aide de la commande OBJET FIXER FORMATAGE (nouveau nom de la commande CHOIX FORMATAGE).

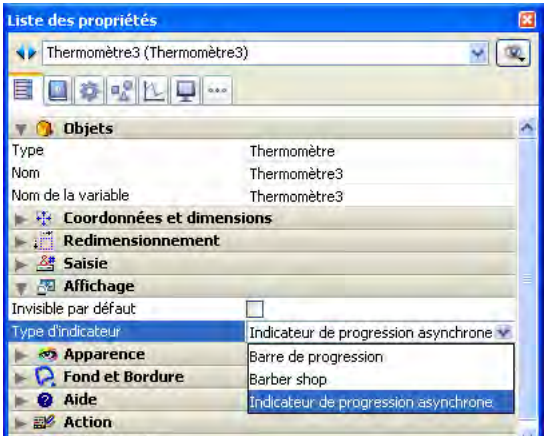
Indicateur de progression asynchrone

Note préliminaire : Dans 4D v12, pour plus de précision et de conformité avec les standards d'interface, les objets "Thermomètres" ont été renommés "Indicateurs de progression".

Plusieurs variantes d'indicateurs de progression sont désormais disponibles dans 4D v12. Ces différentes variantes permettent d'afficher des objets d'interface standard indiquant des opérations courantes telles que la recherche de connexion réseau ou le calcul.

Pour accéder à ces variantes, dessinez un indicateur de progression dans le formulaire. Dans la Liste des propriétés, le nouveau menu "Type

d'indicateur" (thème Affichage) contient les différents indicateurs possibles :



- **Barre de progression** : indicateur par défaut. Cette variante correspond à l'objet "Thermomètre" des versions précédentes de 4D. Les autres options du thème "Graduations" peuvent être définies.
- **Barber shop** : indicateur en barre affichant une animation continue de type "barber shop". Cette variante correspond à l'option "Indéterminé" dans les versions précédentes de 4D. Lorsque cette variante est sélectionnée, le thème "Graduations" est masqué.
- **Indicateur de progression asynchrone** : indicateur circulaire affichant une animation continue. Lorsque cette variante est sélectionnée, le thème "Graduations" est masqué.

Note En exécution, pour qu'un indicateur soit animé, il suffit que la variable associée à l'objet ait une valeur différente de 0 (par exemple, 1). Si la variable vaut 0, l'animation est stoppée.

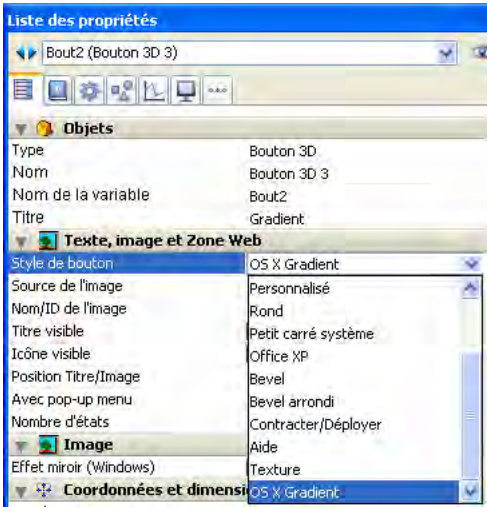
Note Ces variantes peuvent être définies à l'aide de la commande OBJET FIXER FORMATAGE (nouveau nom de la commande CHOIX FORMATAGE).

Nouveaux boutons 3D

Dans 4D v12, la famille de boutons 3D comprend quatre nouveaux styles vous permettant d'utiliser des boutons natifs dans vos interfaces:

- Contracter/Déployer
- Aide
- OS X Texture
- OS X Gradient

Ces variantes sont accessibles via le pop up menu Style de boutons de la Liste des propriétés pour les boutons 3D, cases à cocher 3D et boutons radio 3D :



Contracter/Déployer

Ce style de bouton permet d'ajouter une icône standard de contraction / déploiement. Ces boutons sont utilisés nativement dans les listes hiérarchiques. Sous Windows, le bouton prend l'apparence d'un [+] ou d'un [-] ; sous Mac OS, il s'agit d'un triangle pointant vers la droite ou vers le bas. Ce style est particulièrement destiné aux cases à cocher 3D, les deux états du bouton correspondant aux états sélectionné/déselectionné de la case :

Windows Mac OS

Aide

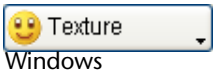
Ce style de bouton permet d'afficher un bouton d'aide standard du système. Vous pouvez utiliser ce style pour ajouter des boutons d'aide "système" dans vos formulaires :


Windows

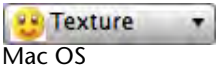

Mac OS

OS X Texture

Sous Mac OS X, un bouton "Texture" est un bouton système standard affichant un dégradé de gris. Sa hauteur est prédéfinie : il n'est pas possible de l'agrandir ou de la réduire. Ce style de bouton bénéficie de toutes les options des boutons 3D.
Sous Windows, ce style équivaut à un bouton poussoir pouvant toutefois bénéficier d'un pop up menu et a la particularité d'être transparent sous Vista :



Windows

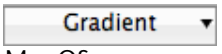


Mac OS

OS X Gradient

Sous Mac OS X, un bouton "Gradient" est un bouton système bicolore. Ce style de bouton bénéficie de toutes les options des boutons 3D.
Sous Windows, ce style équivaut à un bouton poussoir, pouvant toutefois bénéficier d'un pop up menu :

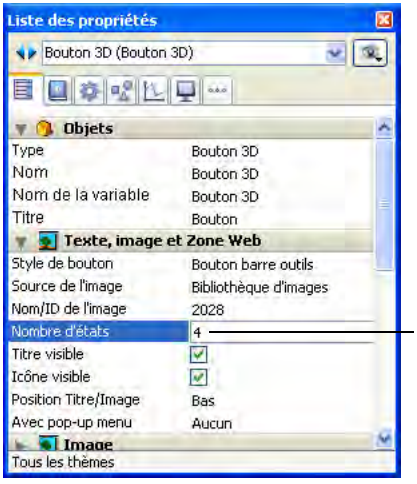
Windows



Mac OS

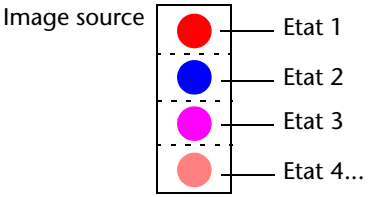
Nombre d'états

Une nouvelle propriété est disponible pour les boutons 3D : **Nombre d'états**.



Nouvelle propriété

Cette propriété permet de préciser le nombre d'états présents dans l'image utilisée comme icône pour le bouton 3D. Dans l'image source, les états doivent être superposés verticalement :

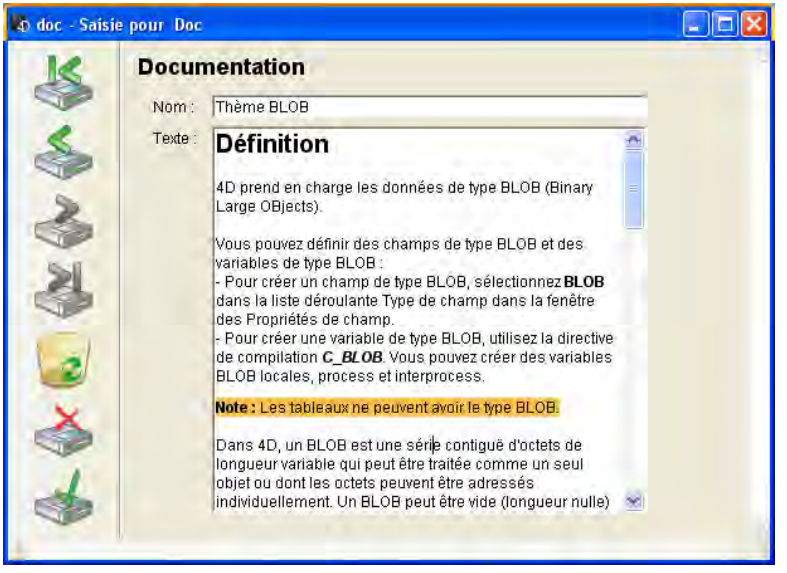


En général, une icône de bouton comporte 4 états : actif, cliqué, survolé et inactivé.

Zones de texte riche

Présentation

4D v12 permet d'utiliser des zones de "texte riche" (*rich text*) comportant des variations de style individuelles. Par exemple, il est désormais possible de mettre des mots en gras, en italique ou en couleur à l'intérieur d'une zone de texte :



Cette nouveauté s'applique aux champs et variables de type Alpha, Texte ainsi qu'aux cellules de listbox. Elle est prise en charge pour les formulaires page, liste, à l'affichage et à l'impression.

De nouvelles options de la Liste des propriétés permettent de configurer le fonctionnement des zones de texte riche.

Les attributs accessibles sont la **police**, la **taille**, le **style**, la **couleur du texte** et (Windows uniquement) la **couleur du fond**. Pour modifier des attributs de style dans une zone de texte riche, vous disposez de deux possibilités :

- en exécution, utiliser un pop up menu automatique (la disponibilité de ce menu est configurable dans la Liste des propriétés).
- par programmation, utiliser la nouvelle commande OBJET FIXER ATTRIBUT TEXTE STYLE.

Dans les zones de texte riche, les attributs de style sont stockés sous forme de balises HTML standard. Au moment de l'affichage, ces balises sont interprétées par 4D. Ce principe permet au développeur de définir et de modifier des attributs de style dans un texte par programmation. Les balises prises en charge par 4D sont décrites dans l'annexe B, "Balises de style," page 337.

La nouvelle commande OBJET Lire texte brut permet de récupérer le texte brut sans balises de style.

Note Les zones de texte riche ne sont pas utilisables dans les contextes suivants : filtres de saisie, états rapides et éditeur d'étiquettes.

Propriétés de gestion du texte riche

Dans l'éditeur de formulaires, de nouvelles propriétés permettent d'activer et de configurer la prise en charge du texte riche. Ces propriétés sont disponibles pour les variables saisissables, les champs et les cellules de listbox de type Alpha ou Texte.

Multistyle

Cette option (thème "Texte") active la possibilité d'utiliser des styles spécifiques dans la zone sélectionnée. Lorsque cette option est cochée, 4D interprète les éventuelles balises HTML de gestion de style présentes dans la zone.

Par défaut, cette option n'est pas cochée.

Stocker les balises par défaut

Cette option n'apparaît que si l'option "Multistyle" a été cochée. Elle est également située dans le thème "Texte".
Lorsque cette option est cochée, la zone stockera des balises de style avec le texte, même si aucune modification n'est effectuée. Dans ce cas, les balises correspondent au style par défaut. Lorsque cette option n'est pas cochée, seules les balises de style modifié sont stockées.

Par exemple, voici un texte comportant une modification de style :

What a **beautiful** day!

- Si l'option "Stocker les balises par défaut" n'est pas cochée, la zone ne stocke que la modification. Le contenu stocké est donc :

  What a <SPAN STYLE="font-size:13.5pt">beautiful</SPAN> day!

- Si l'option est cochée, la zone stocke toutes les informations de formatage. Une première balise générique décrit le style par défaut puis chaque variation fait l'objet d'une paire de balises imbriquées. Le contenu stocké dans la zone est alors :

  <SPAN STYLE="font-family:'Arial';font-size:9pt;text-align:left;font-weight:normal;font-style:normal;text-decoration:none;color:#000000;background-color:#FFFFFF">What a <SPAN STYLE="font-size:13.5pt">beautiful</SPAN> day!</SPAN>

Menu contextuel

Cette option (thème "Saisie") n'apparaît que si l'option "Multistyle" a été cochée.
Elle permet d'activer pour l'utilisateur la possibilité d'appeler en cours de saisie un pop up menu via un clic droit dans la zone. Ce pop up menu propose des commandes standard d'édition de texte (couper,

copier, coller) ainsi que les commandes de modification de style prises en charge : police, taille, style, couleur et (Windows) couleur de fond.

Lorsque l'utilisateur modifie un attribut de style via ce pop up menu, 4D génère l'événement formulaire Sur après modification.

Note Il est également possible de modifier les styles via la nouvelle commande OBJET FIXER ATTRIBUT TEXTE STYLE. A noter que dans ce cas, aucun événement formulaire n'est généré.

Note Le style "barré" n'est pas pris en charge sous Mac OS. La balise correspondante peut toutefois être utilisée par programmation.

Traitement du texte riche

Copier coller et glisser-déposer

Les attributs de style pris en charge (police, taille, style et couleur) sont conservés en cas de glisser-déposer ou copier-coller de texte stylé entre :

- différentes zones de texte riche à l'intérieur de 4D (variables/champs texte et listbox)
- une zone 4D Write et une zone de texte riche 4D,
- un texte externe stylé et une zone de texte riche 4D.

Dans les autres cas, les styles seront conservés suivant le contexte.

Commandes de gestion des objets texte

Les commandes permettant de manipuler des objets texte par programmation ne tiennent pas compte des éventuelles balises de style intégrées au texte. Elles agissent sur le texte affiché. Elles fonctionnent donc comme dans les versions précédentes de 4D. Il s'agit des commandes suivantes :

- Thème **Interface utilisateur**
  SELECTIONNER TEXTE
  TEXTE SELECTIONNE

  A noter que, lorsque vous utilisez ces commandes avec des commandes de manipulation des chaînes de caractères, il est nécessaire de filtrer les caractères de formatage à l'aide de la nouvelle commande OBJET Lire texte brut :

  SELECTIONNER TEXTE([Produits]Notes;1;**Longueur**(**OBJET Lire texte brut**([Produits]Notes))+1)

- Thème **Propriétés des objets**

Note Les commandes de ce thème ont été renommées dans 4D v12 et de nouvelles commandes ont été ajoutées. Pour plus d'informations, reportez-vous au paragraphe "Propriétés des objets", page 197.

Les commandes permettant de modifier le style des objets (par exemple OBJET FIXER POLICE) fonctionnent comme dans les versions précédentes de 4D : elles s'appliquent à l'objet entier et non à la sélection.
A noter que si l'objet n'a pas le focus au moment de l'exécution de la commande, la modification est appliquée simultanément à l'objet (la zone de texte) et à sa variable associée. Si l'objet a le focus, la modification est effectuée sur l'objet mais pas sur la variable associée. La modification n'est appliquée à la variable qu'au moment où l'objet perd le focus. Gardez ce principe à l'esprit lorsque vous programmez les zones de texte.

Si l'option "Stocker les balises par défaut" est cochée pour l'objet, l'utilisation de ces commandes provoquera une modification des balises enregistrées avec chaque objet.

Commande Lire texte edite

Lorsqu'elle est utilisée avec une zone de texte riche, la commande **Lire texte edite** (thème "Evénements formulaire") retourne le texte de la zone courante en incluant les éventuelles balises de style.

Pour récupérer le texte "brut" (texte sans balises) en cours d'édition, vous devez utiliser la nouvelle commande OBJET Lire texte brut :

OBJET Lire texte brut(Lire texte edite)

Note Pour plus d'informations sur cette commande, reportez-vous au paragraphe "OBJET Lire texte brut", page 218.

Commandes de recherche et de tri

Les recherches et les tris effectués parmi des objets multistyles tiennent compte des éventuelles balises de style enregistrées dans l'objet. Si une modification de style a été apportée à l'intérieur d'un mot, une recherche sur ce mot sera infructueuse.

Pour pouvoir effectuer des recherches et des tris valides, vous devez utiliser la nouvelle commande OBJET Lire texte brut. Par exemple :

CHERCHER PAR FORMULE([MaTable];**OBJET Lire texte brut**([MaTable]MonchampStyle)="très bien")

Nouvelles commandes

Plusieurs nouvelles commandes permettent de manipuler les attributs de style dans les zones de texte riche : OBJET FIXER ATTRIBUT TEXTE STYLE, OBJET LIRE ATTRIBUT TEXTE STYLE et OBJET Lire texte brut. Ces commandes sont décrites dans le paragraphe "Propriétés des objets", page 197.

Note La nouvelle propriété "Sélection toujours visible" permet de maintenir visible la sélection de texte dans un formulaire même lorsque l'objet n'a plus le focus. Ce principe permet de construire des interfaces à base de menus ou de boutons pour la modification de style. Pour plus d'informations, reportez-vous au paragraphe "Sélection toujours visible", page 106.

Nouvelles propriétés pour les champs et variables

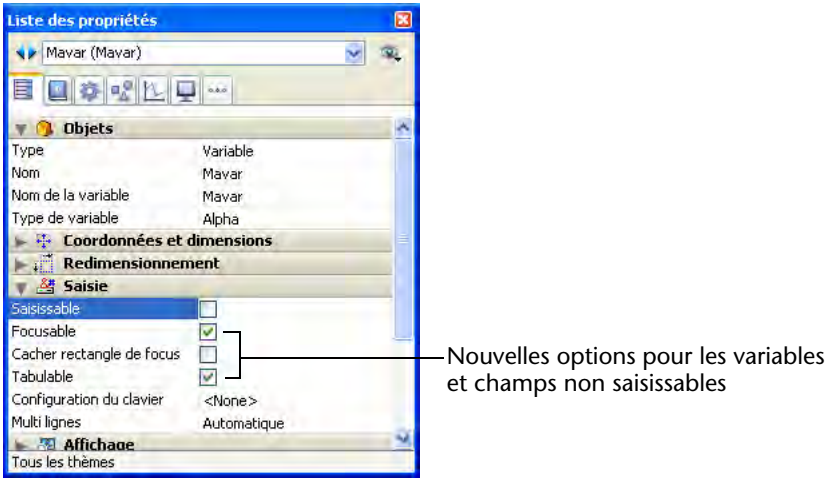
De nouvelles propriétés sont disponibles pour les champs et les variables dans l'éditeur de formulaires de 4D. Cet ensemble de nouveautés a pour objectif de fournir au développeur 4D un meilleur contrôle sur ces objets de formulaire.

Objets non saisissables

Dans les versions précédentes de 4D, le contenu des champs et des variables déclarés non saisissables (option **Saisissable** désélectionnée) ne pouvait être ni sélectionné ni copié. Désormais, de nouvelles possibilités d'interaction sont fournies via la disponibilité des propriétés **Focusable**, **Tabulable** et **Déposable**.

Focusable et Tabulable

Les options **Focusable** et **Tabulable** sont actives pour les objets non saisissables :



Lorsque l'option **Focusable** est cochée pour l'objet, l'utilisateur peut sélectionner, copier ou encore transporter par glisser-déposer le contenu de la zone.

L'option **Tabulable** est disponible une fois que l'option **Focusable** est cochée. Elle permet de sélectionner l'objet via la touche **Tabulation**. A noter qu'il n'est toutefois pas possible d'inclure l'objet dans l'ordre de saisie.

Déposable

La propriété **Déposable** peut désormais être affectée à un objet non saisissable. Ce principe permet au développeur de programmer tout type d'action en réponse au déposer d'un objet sur un champ ou une variable non saisissable.

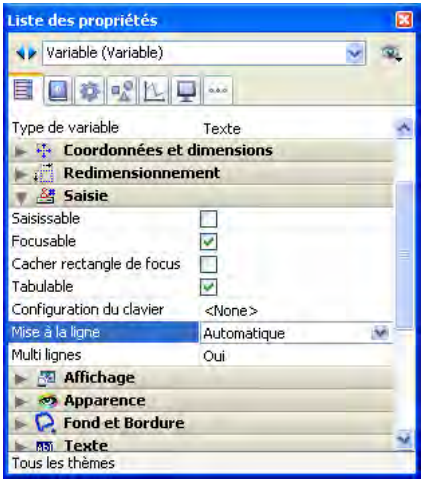
Objets multilignes

Dans les versions précédentes de 4D, l'affichage et l'impression du texte dans les champs et les variables de formulaire étaient difficilement contrôlables. Dans les objets de type texte, l'affichage pouvait dépendre de la taille de la zone et de la plate-forme.

Note Les retours chariot ne sont pas utilisables dans les objets de type alpha.

Dans 4D v12, l'affichage et l'impression des zones de texte ont été harmonisés entre les plates-formes une combinaison de deux nouvelles options permet de contrôler les retours à la ligne :

- **Multilignes** : Option disponible pour les variables et champs de type alpha et texte, saisissables ou non. Elle peut prendre trois valeurs : **Oui**, **Non**, **Automatique**.
- **Retour à la ligne** : Option disponible uniquement si l'option **Multilignes** est à **Oui**. Elle peut prendre trois valeurs : **Oui**, **Non**, **Automatique**.

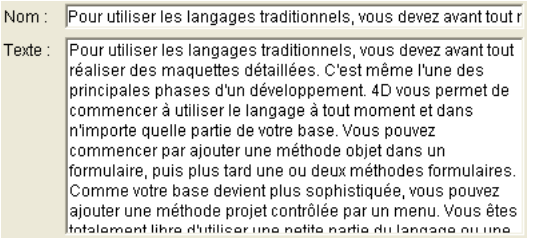


Ces options permettent de modifier deux paramètres d'affichage des zones de texte :

- l'affichage des mots situés en fin de ligne dans les zones mono-lignes,
- l'insertion automatique de retours à la ligne dans les zones de texte.

Multilignes = Automatique

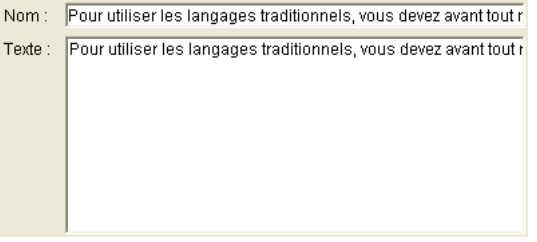
Dans les zones mono-lignes, les mots situés en fin de ligne sont tronqués et il n'y a pas de retours à la ligne.
Dans les zones multi-lignes, 4D effectue des retours à la ligne automatiques :



Ce fonctionnement correspond au fonctionnement des versions précédentes de 4D.

Multilignes = Non

Dans les zones mono-lignes, les mots situés en fin de ligne sont tronqués et il n'y a pas de retours à la ligne.
Il n'y a aucun retour à la ligne : le texte est toujours affiché sur une seule ligne. Si le champ ou la variable alpha ou texte contient des retour chariots, le texte situé après le premier retour chariot est effacé dès que la zone est modifié :

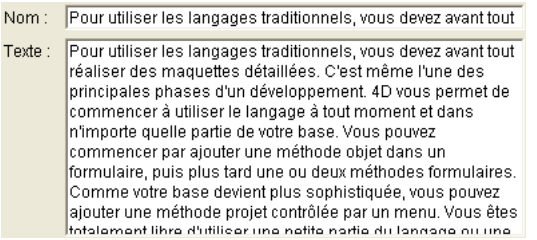


Multilignes = Oui

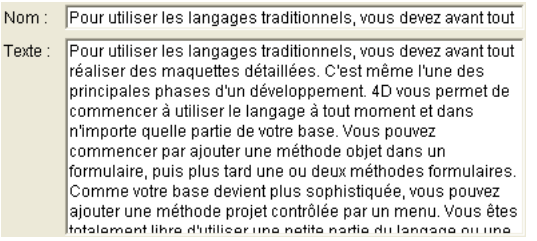
Dans ce cas, une option supplémentaire, Retour à la ligne, doit être définie.

- **Retour à la ligne = Automatique**
  Dans les zones mono-lignes, le texte est affiché jusqu'au premier retour chariot ou au dernier mot affichable en entier. 4D insère des retours à la ligne, il est possible de faire défiler le contenu de la zone en appuyant sur la touche flèche basse.

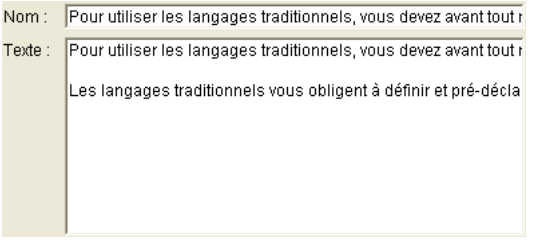
Dans les zones multi-lignes, 4D effectue des retours à la ligne automatiques :



- **Retour à la ligne = Oui**
  Dans les zones mono-lignes, seul le dernier mot affichable en entier est affiché. 4D insère des retours à la ligne, il est possible de faire défiler le contenu de la zone en appuyant sur la touche flèche basse.
  Dans les zones multi-lignes, 4D effectue des retours à la ligne automatiques :



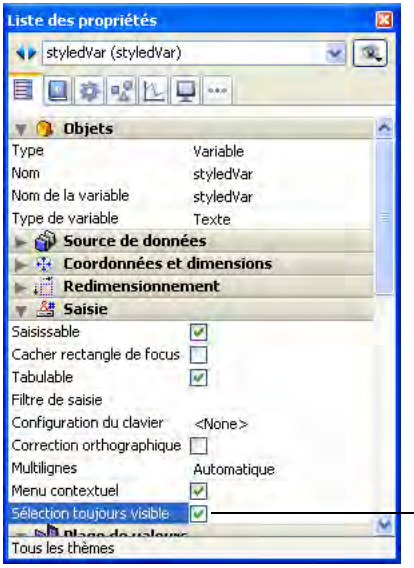
- **Retour à la ligne = Non**
  4D n'effectue aucun retour à la ligne automatique et le dernier mot affichable est éventuellement tronqué. Dans les zones de type texte, les retours chariot sont pris en charge :



Note Les objets texte ne doivent pas comporter de barres de défilement pour que ces propriétés soient correctement prises en compte.

Sélection toujours visible

Une nouvelle propriété est disponible pour les champs ou variables de type Alpha ou Texte dans les formulaires : **Sélection toujours visible** :



Cette propriété permet de conserver la visibilité de la sélection à l'intérieur de l'objet après qu'il ait perdu le focus. Ce principe facilite notamment la mise en place d'interfaces permettant de modifier le style du texte (cf. paragraphe "Zones de texte riche", page 96).

List box

4D v12 apporte plusieurs nouveautés relatives aux list box :

- la possibilité de définir des list box hiérarchiques,
- la possibilité d'imprimer des list box,
- l'accès aux données des colonnes ajoutées suite à un SELECT.

List box hiérarchiques

4D v12 vous permet de définir et d'utiliser des list box hiérarchiques. Une list box hiérarchique est une list box dans laquelle **le contenu de la première colonne** apparaît sous forme hiérarchique. Ce type de représentation est adapté à la présentation d'informations comportant des valeurs répétées et/ou hiérarchiquement dépendantes (pays/région/ville...).

Seules les list box **de type tableau** peuvent être hiérarchiques.

Les list box hiérarchiques constituent un mode de représentation particulier des données, mais ne modifient pas la structure de ces données (les tableaux). Les list box hiérarchiques sont gérées exactement de la même manière que les list box non hiérarchiques. Pour plus d'informations, reportez-vous au paragraphe "Fonctionnement des list box hiérarchiques", page 112

Pour définir une list box hiérarchique, vous disposez de trois possibilités :

- configurer manuellement les éléments hiérarchiques via la liste des propriétés dans l'éditeur de formulaires. Ce point est décrit dans le paragraphe "Nouvelles propriétés de list box", page 107.
- générer visuellement la hiérarchie à l'aide du pop up menu de gestion des list box, dans l'éditeur de formulaires. Ce point est décrit dans le paragraphe "Gérer la hiérarchie avec le menu contextuel", page 110.
- utiliser les nouvelles commandes du langage LISTBOX FIXER HIERARCHIE et LISTBOX LIRE HIERARCHIE. Pour plus d'informations, reportez-vous au paragraphe "List Box", page 173.

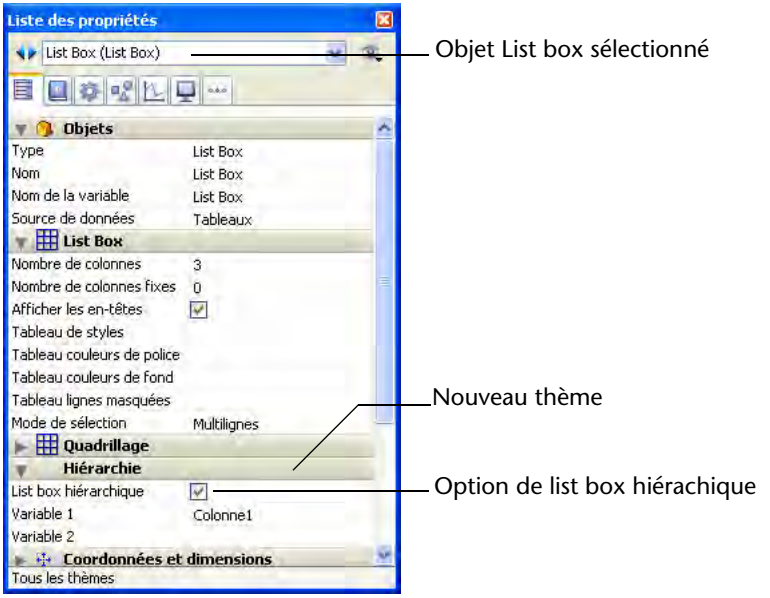
Nouvelles propriétés de list box

De nouvelles propriétés permettent de gérer les list box hiérarchiques. A noter que ces propriétés sont modifiées automatiquement si vous définissez la hiérarchie via le pop up menu de l'objet list box (cf. paragraphe "Gérer la hiérarchie avec le menu contextuel", page 110).

- **List box hiérarchique**

  Pour définir une list box hiérarchique, il vous suffit de cocher la nouvelle option **List box hiérarchique**.

Cette option est placée dans le nouveau thème "Hiérarchie" de la liste des propriétés, accessible lorsque l'objet list box est sélectionné :

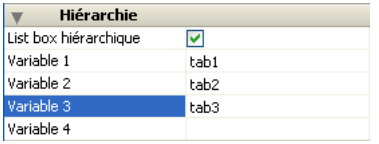


Cette option n'est présente que pour les list box dont la source de données est Tableaux.

- **Variable 1 ... 10**

Lorsque l'option **List box hiérarchique** est cochée, des champs saisissables supplémentaires sont ajoutés dans le thème : "Variable 1", "Variable 2", etc. A chaque saisie d'une valeur dans un champ, une nouvelle ligne est ajoutée. Jusqu'à 10 variables peuvent être définies.

Ces variables définissent les niveaux hiérarchique à afficher dans la première colonne.



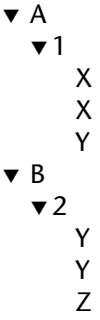
La première variable correspond toujours au nom de variable de la première colonne de la list box (les deux valeurs sont automatiquement liées). Cette première variable est toujours visible et saisissable. Par exemple : pays.

La seconde variable est également toujours visible et saisissable, elle définit le deuxième niveau hiérarchique. Par exemple : régions.

A partir du troisième champ, chaque variable dépend de celle qui la précède. Par exemple : départements, villes, etc. Un maximum de dix niveaux hiérarchiques peut être défini.

Si vous effacez une valeur, l'ensemble de la liste hiérarchique définie remonte d'un niveau.

La dernière variable n'est jamais hiérarchique même si plusieurs valeurs identiques existent à ce niveau. Par exemple, en reprenant la configuration illustrée ci-dessus, imaginons que *tab1* contienne les valeurs A A A B B B, *tab2* les valeurs 1 1 1 2 2 2 et *tab3* les valeurs X X Y Y Y Z. Dans ce cas, A, B, 1 et 2 pourront apparaître sous forme contractée, mais pas X et Y :



Ce principe n'est pas appliqué lorsqu'une seule variable est définie dans la hiérarchie : dans ce cas, les valeurs identiques pourront être groupées.

Note Si vous définissez une hiérarchie basée sur les premières colonnes d'une list box existante, vous devez ensuite supprimer ou masquer ces colonnes (à l'exception de la première) sinon elles appraîtront en double dans la list box. Si vous définissez la hiérarchie via le pop up menu de l'éditeur (cf. paragraphe "Gérer la hiérarchie avec le menu contextuel", page 110), les colonnes superflues sont automatiquement supprimées de la list box.

- **Propriétés modifiées**

  Lorsque le mode hiérarchique est activé (option "List box hiérarchique" cochée), plusieurs propriétés sont désactivées ou supprimées :

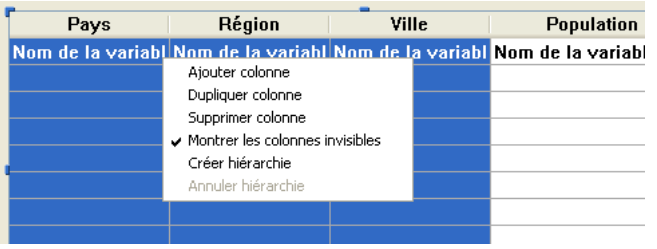
  - **Nombre de colonnes fixes** : pour les list box hiérarchiques, cette option est toujours au minimum à 1.
  - **Lignes déplaçables** : cette option n'est pas disponible pour les list box hiérarchiques.
  - Toutes les propriétés du thème "Affichage" sont désactivées pour la première colonne.

Gérer la hiérarchie avec le menu contextuel

Le menu contexuel de l'éditeur de formulaires propose deux nouvelles commandes lorsque vous cliquez sur un objet list box : **Créer hiérarchie** et **Annuler hiérarchie**.

- **Créer hiérarchie**

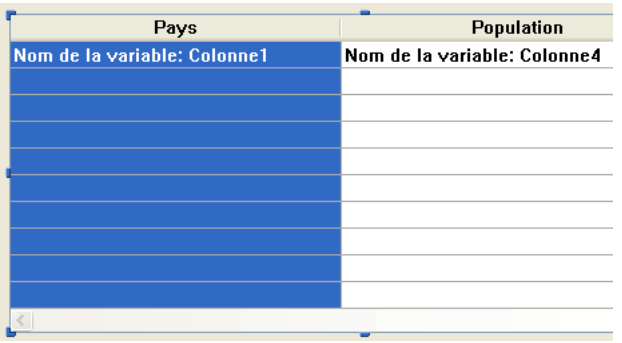
  Lorsque vous sélectionnez au moins une colonne en plus de la première dans un objet list box (de type tableau) dans l'éditeur de formulaires, la commande **Créer hiérarchie** est disponible dans le menu contextuel :

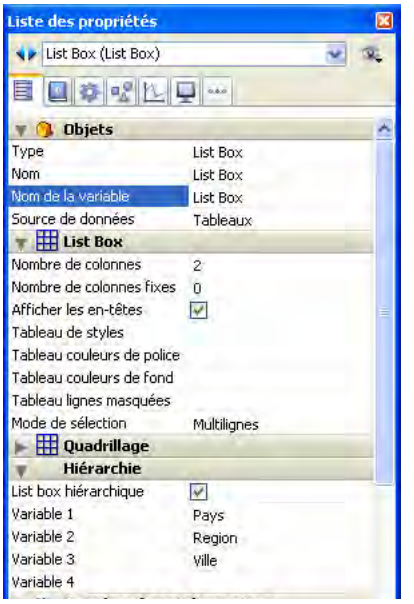


  Lorsque vous choisissez cette commande, les actions suivantes sont effectuées :

  - l'option "List box hiérarchique" est cochée pour l'objet (cf. paragraphe "List box hiérarchique", page 107).
  - Les variables des colonnes sont utilisées pour définir la hiérarchie (cf. paragraphe "Variable 1 ... 10", page 108). Elles remplacent les variables éventuellement déjà définies.
  - Les colonnes sélectionnées n'apparaissent plus dans la list box (à l'exception du titre de la première).

Exemple : soit une list box dont les premières colonnes contiennent Pays, Région, Ville et Population. Lorsque Pays, Région et Ville sont sélectionnées (cf. illustration ci-dessus), si vous choisissez **Créer hiérarchie** dans le menu contextuel, une hiérarchie à trois niveaux est créée dans la première colonne, les colonnes 2 et 3 sont supprimées et la colonne Population la deuxième :





- **Annuler hiérarchie**

  Lorsque la première colonne est sélectionnée et déjà définie comme hiérarchique, vous pouvez utiliser la commande **Annuler hiérarchie**. Lorsque vous choisissez cette commande, les actions suivantes sont effectuées :

  - l'option "List box hiérarchique" est désélectionnée pour l'objet,
  - les niveaux hiérarchiques 2 à n sont supprimés et transformés en colonnes ajoutées dans la list box.

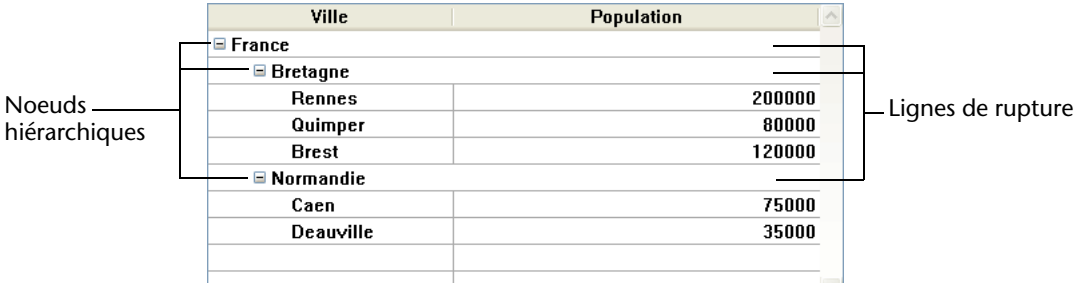
Fonctionnement des list box hiérarchiques

A la première ouverture d'un formulaire contenant une list box hiérarchique, par défaut toutes les lignes sont déployées.

Une ligne de rupture et un "noeud" hiérarchique sont automatiquement ajoutés dans la list box lorsque des valeurs sont répétées dans les tableaux. Par exemple, imaginons une list box contenant quatre tableaux définissant des villes, chaque ville étant caractérisée par un pays, une région, un nom et un nombre d'habitants :

Pays	Région	Ville	Population
France	Bretagne	Rennes	200000
France	Bretagne	Quimper	80000
France	Bretagne	Brest	120000
France	Normandie	Caen	75000
France	Normandie	Deauville	35000

Si cette list box est affichée sous forme hiérarchique (les trois premiers tableaux étant inclus dans la hiérarchie), vous obtenez :



Les tableaux ne sont pas triés avant la construction de la hiérarchie. Si par exemple un tableau contient les données AAABBAACC, la hiérarchie obtenue sera :

- A
- B
- A
- C

Pour déployer ou contracter un "noeud" hiérarchique, cliquez dessus. Si vous effectuez **Alt+clic** (Windows) ou **Option+clic** (Mac OS) sur le noeud, tous ses sous-éléments seront déployés ou contractés.

Gestion des sélections et des positions

Une list box hiérarchique affiche un nombre variable de lignes à l'écran en fonction de l'état déployé/contracté des noeuds hiérachiques. Cela ne signifie pas pour autant que le nombre de lignes des tableaux varie. Seul l'affichage est modifié, pas les données.

Il est important de comprendre ce principe car la gestion programmée des list box hiérarchiques se base toujours sur les données des tableaux, pas sur les données affichées. En particulier, les lignes de rupture ajoutées automatiquement ne sont pas prises en compte dans les tableaux d'options d'affichage (cf. paragraphe "Gestion des lignes de rupture", page 115).

Examinons par exemple les tableaux suivants :

France	Bretagne	Brest
France	Bretagne	Quimper
France	Bretagne	Rennes

Si ces tableaux sont représentés hiérarchiquement, la ligne "Quimper" ne sera pas affichée sur la deuxième ligne mais sur la quatrième, à cause des deux lignes de rupture ajoutées :

France
Bretagne
Brest
Quimper
Rennes

Quelle que soit la manière dont les données sont affichées dans la list box (hiérarchique ou non-hiérarchique), si vous souhaitez passer la ligne contenant "Quimper" en gras, vous devrez utiliser l'instruction Style{2} = gras. Seule la position de la ligne dans les tableaux est prise en compte.

Ce principe est mis en oeuvre pour les tableaux internes permettant de gérer :

- les couleurs
- les couleurs de fond
- les styles
- les lignes masquées
- les sélections

Par exemple, si vous voulez sélectionner la ligne contenant Rennes, vous devez passer :

-> MaListbox{3}:=**Vrai**

Représentation non hiérarchique :

France	Bretagne	Brest
France	Bretagne	Quimper
France	Bretagne	Rennes

Représentation hiérarchique :

France
Bretagne
Brest
Quimper
Rennes

Note Si une ou plusieurs lignes sont masquées du fait que leurs parents ont été contractés, elles ne sont plus sélectionnées. Seules les lignes visibles (directement ou suite à un défilement) sont sélectionnables. Autrement dit, les lignes ne peuvent pas être à la fois sélectionnées et cachées.

Tout comme pour les sélections, la commande LISTBOX LIRE POSITION CELLULE retournera les mêmes valeurs pour une list box hiérarchique et une list box non hiérarchique. Cela signifie que dans les deux exemples ci-dessous, LISTBOX LIRE POSITION CELLULE retournera la même position : (3;2)

Représentation non hiérarchique :

France	Bretagne	Brest	120000
France	Bretagne	Quimper	80000
France	Bretagne	Rennes	200000
France	Normandie	Caen	75000

Représentation hiérarchique :

France		
Bretagne		
	Brest	120000
	Quimper	80000
	Rennes	200000
Normandie		
	Caen	75000

Gestion des lignes de rupture

Si l'utilisateur sélectionne une ligne de rupture, LISTBOX LIRE POSITION CELLULE retourne la première occurrence de la ligne dans le tableau correspondant. Dans le cas suivant :

France	
Bretagne	
Brest	120000
Quimper	80000
Rennes	200000
Normandie	
Caen	75000

... LISTBOX LIRE POSITION CELLULE retourne (2;4). Pour sélectionner une ligne de rupture par programmation, vous devez utiliser la nouvelle commande LISTBOX SELECTIONNER RUPTURE, page 180.

Les lignes de rupture ne sont pas prises en compte dans les tableaux internes permettant de gérer l'apparence graphique des list box (styles et couleurs). Il est toutefois possible de modifier ces caractéristiques pour les lignes de rupture via les commandes de gestion graphique des objets (thème "Propriétés des objets"). Il suffit pour cela d'exécuter ces commandes appropriées sur les tableaux constituant la hiérarchie. Soit par exemple la list box suivante (les noms des tableaux associés sont précisés entre parenthèses) :

Représentation non hiérarchique :

(T1)	(T2)	(T3)	(T4)	(tStyle)	(tCouleur)
France	Bretagne	Brest	120000	*Normal*	*0*
France	Bretagne	Quimper	80000	*Souligné*	*0*
France	Bretagne	Rennes	200000	*Normal*	*0xFF0000*
France	Normandie	Caen	220000	*Normal*	*0*
France	Normandie	Deauville	4000	*Normal*	*0*

Représentation hiérarchique :

France	
Bretagne	
Brest	120000
Quimper	80000
Rennes	200000
Normandie	
Caen	75000
Deauville	4000

En mode hiérarchique, les niveaux de rupture ne sont pas pris en compte par les tableaux de modification de style nommés *tStyle* et *tCouleurs*. Pour modifier la couleur ou le style des niveaux de rupture, vous devez exécuter les instructions suivantes :

OBJET FIXER COULEURS RVB(T1;0x0000FF;0xB0B0B0)
OBJET FIXER STYLE(T2;Gras)

Dans ce contexte, seule la syntaxe utilisant la variable tableau peut fonctionner avec les commandes de propriété d'objet car les tableaux n'ont alors pas d'objet associé.

Résultat :

France	
Bretagne	
Brest	120000
Quimper	80000
Rennes	200000
Normandie	
Caen	75000
Deauville	4000

Gestion des tris

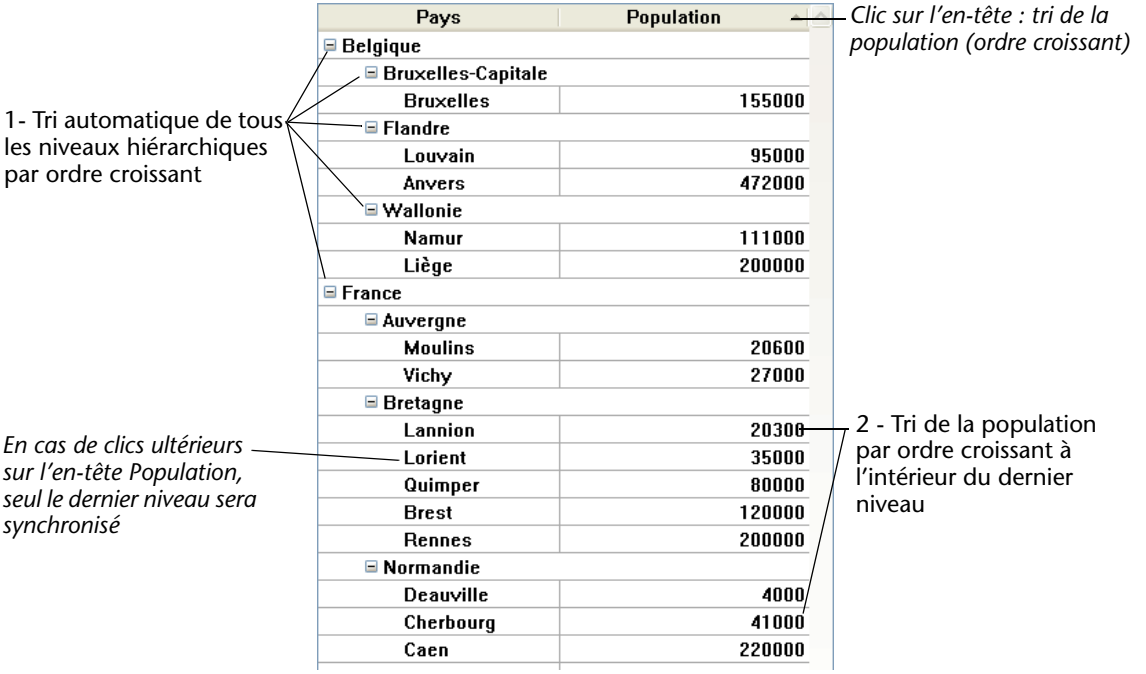
Dans une list box en mode hiérarchique, un tri standard (effectué suite à un clic dans un en-tête de colonne de la list box) est toujours construit de la manière suivante :

- En premier lieu, tous les niveaux de la colonne hiérarchique (première colonne) sont automatiquement triés par ordre croissant.
- Le tri est ensuite effectué par ordre croissant ou décroissant (suivant l'action utilisateur) sur les valeurs de la colonne où le clic a eu lieu.
- Toutes les colonnes sont synchronisées.
- Lors des tris ultérieurs des colonnes non hiérarchiques de la list box, seul le dernier niveau de la première colonne est trié. Il est possible de modifier le tri de cette colonne en cliquant sur son en-tête.

Soit par exemple la list box suivante, dans laquelle aucun tri spécifique n'est défini :

Pays	Population
France	
Bretagne	
Rennes	200000
Quimper	80000
Brest	120000
Lannion	20300
Lorient	35000
Normandie	
Caen	220000
Deauville	4000
Cherbourg	41000
Auvergne	
Vichy	27000
Moulins	20600
Belgique	
Wallonie	
Namur	111000
Liège	200000
Flandre	
Anvers	472000
Louvain	95000
Bruxelles-Capitale	
Bruxelles	155000

Si vous cliquez sur l'en-tête "Population" afin de trier les populations par ordre croissant (ou alternativement décroissant), les données apparaissent ainsi :



Comme pour toutes les list box, vous pouvez désactiver le mécanisme de tri standard en désélectionnant l'option "Triable" pour la list box et gérer le tri par programmation.

Affichage des dates et des heures

Lorsque des valeurs de type date ou heure sont incluses dans une list box hiérarchique, elles sont affichées dans un format normalisé :

- les dates sont affichées au format système court (par exemple, pour le 30 mai 2009, "05/30/09" sur un système américain et "30/05/09" sur un système européen).
- les heures sont également affichées au format système court ("12:15:30" ou "12:15" en fonction des paramètres système).

Lignes masquées

Lorsque toutes les lignes d'une sous-hiérarchie sont masquées, la ligne de rupture est automatiquement masquée. Dans l'exemple ci-dessus, si les lignes 1 à 3 sont masquées, la ligne de rupture "Bretagne" n'apparaîtra pas.

Défilement et édition

Les commandes OBJET FIXER DEFILEMENT (anciennement DEFILER LIGNES) et EDITER ELEMENT ont été adaptées pour les list box affichées en mode hiérarchique : si la ligne visée appartient à un niveau hiérarchique contracté, le niveau ainsi que les éventuels niveaux parents sont automatiquement déployés afin que la ligne soit visible.

Impression des list box

Il est désormais possible d'imprimer des list box dans 4D v12. Deux modes d'impression sont proposés : le mode **prévisualisation**, permettant d'imprimer une list box comme un objet de formulaire et le mode **avancé**, permettant de contrôler l'impression de l'objet list box lui-même au sein du formulaire. Une nouvelle propriété permet de définir l'apparence "Impression" pour les objets list box.

Mode prévisualisation

L'impression d'une list box en mode **prévisualisation** consiste à imprimer directement la list box avec le formulaire qui la contient via les commandes d'impression standard ou la commande de menu **Imprimer**. La list box est imprimée dans l'état où elle se trouve dans le formulaire.

Ce mode ne permet pas de contrôler précisément l'impression de l'objet ; en particulier, il ne permet pas d'imprimer toutes les lignes d'une list box contenant plus de lignes qu'elle ne peut en afficher.

Mode avancé

Dans ce mode, l'impression des list box s'effectue par programmation, via la nouvelle commande Imprimer objet. Par conséquent, seules des list box présentes dans des formulaires projet peuvent être imprimées en mode avancé. La nouvelle commande LISTBOX LIRE INFORMATIONS IMPRESSION permet de contrôler l'impression de l'objet.

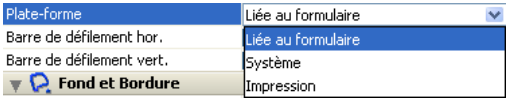
Dans ce mode :

- la hauteur de l'objet list box est automatiquement réduite lorsque le nombre de lignes à imprimer est inférieur à la hauteur d'origine de l'objet (il n'y a pas de lignes "vides" imprimées). En revanche, la hauteur n'augmente pas automatiquement en fonction du contenu de l'objet. La taille de l'objet effectivement imprimé peut être obtenue via la commande LISTBOX LIRE INFORMATIONS IMPRESSION.

- l'objet list box est imprimé "tel quel", c'est-à-dire en tenant compte de ses paramètres d'affichage courants : visibilité des en-têtes et des grilles, lignes affichées et masquées, etc.
  Ces paramètres incluent également la première ligne à imprimer : si vous appelez la commande DEFILER LIGNES avant de lancer l'impression, la première ligne imprimée dans la list box sera celle désignée par la commande.
- un mécanisme automatique facilite l'impression des list box contenant plus de lignes qu'il est possible d'en afficher : des appels successifs à Imprimer objet permettent d'imprimer à chaque fois un nouvel ensemble de lignes. La commande LISTBOX LIRE INFORMATIONS IMPRESSION permet de contrôler le statut de l'impression durant l'impression. Pour plus d'informations, reportez-vous à la description de ces commandes.

Propriété Impression

Dans la Liste des propriétés, une nouvelle option de plate-forme est disponible pour les list box (thème "Apparence") : **Impression**.



Si vous sélectionnez cette option, l'apparence de la list box est adaptée pour l'impression : en-têtes en noir et blanc, cases à cocher imprimées sous forme de [x], etc. L'apparence Impression ne dépend pas de la plate-forme d'exécution.
Si vous ne cochez pas cette option, la list box sera imprimée avec l'apparence de la plate-forme courante.

Note En mode avancé, les barres de défilement des list box ne sont jamais imprimées, quelle que soit l'apparence choisie.

Accès aux données des colonnes SELECT ajoutées

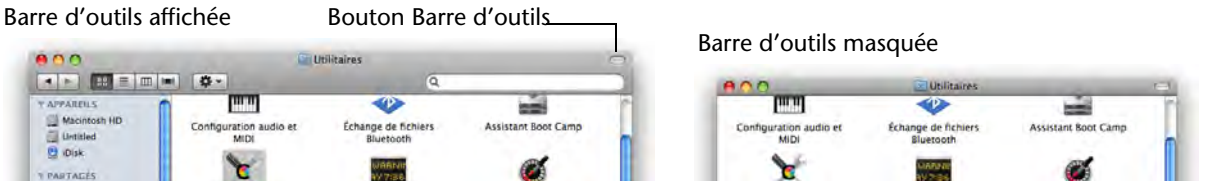
Il est désormais possible de récupérer le contenu des colonnes ajoutées automatiquement par 4D dans une list box suite à une requête SELECT dont le résultat produit un nombre de colonnes supérieur à celui de la list box.

Dans les versions précédentes, ces colonnes ajoutées étaient liées à des tableaux internes inaccessibles au langage de 4D. Dans 4D v12, ces colonnes sont liées à des variables de type *tableau temporaire*. La durée de vie de ces tableaux temporaires est celle du formulaire. Une variable temporaire est également créée pour chaque entête. Lorsque la commande LIRE TABLEAUX LISTBOX est appelée, le paramètre *tabVarCols* contient des pointeurs vers les tableaux temporaires et le paramètre *tabVarEntêtes* contient des pointeurs vers les variables d'entête temporaires.

A noter que la saisie et le tri sont maintenant possibles dans ces colonnes.

Bouton barre d'outils (Mac OS)

Sous Mac OS, 4D v12 permet d'afficher le bouton de gestion de la barre d'outils. Ce bouton standard permet alternativement d'afficher et de masquer la barre d'outils de la fenêtre :



La prise en charge par 4D de cette fonctionnalité s'effectue à deux niveaux :

- une nouvelle propriété permet de dessiner le bouton dans les fenêtres,
- un nouvel événement formulaire est généré lorsque l'utilisateur clique sur le bouton.

Il appartient au développeur de l'interface de définir les actions à effectuer en réponse au clic sur le bouton.

Propriété de barre d'outils

Pour que 4D affiche le bouton de gestion de barre d'outils dans vos fenêtres, vous disposez de deux solutions :

- cocher la nouvelle propriété **Bouton barre outils (Mac OS)** disponible dans le thème "Affichage" de la Liste des propriétés :

  Cette option est à utiliser si vous créez la fenêtre avec la commande Creer fenetre formulaire ou DIALOGUE par exemple.

- utiliser la nouvelle constante Avec bouton barre outils Mac OS. Cette option est à utiliser avec la commande Creer fenetre ou Creer fenetre formulaire (cf. paragraphe "Creer fenetre", page 257).

Evénement formulaire Sur bouton barre outils Mac

Le nouvel événement formulaire Sur bouton barre outils Mac est généré dans la méthode formulaire lorsque l'utilisateur clique sur le bouton de gestion de la barre d'outils de la fenêtre sous Mac OS. Bien entendu, la propriété correspondante doit avoir été cochée dans les propriétés d'événement du formulaire.

Seul l'événement est généré, 4D n'effectue aucune autre action dans la fenêtre. Il revient au développeur de modifier la taille de la fenêtre et d'afficher ou masquer ses éléments d'interface.

5 Langage

Ce chapitre regroupe les nouveautés et modifications apportées au langage de programmation de 4D v12.

Chaînes de caractères

Convertir vers texte

Convertir vers texte(blob ; jeuCaractères) → Texte

La commande Convertir vers texte prend désormais en charge les BOM (*Byte Order Mark*).

Si le jeu de caractères spécifié est de type Unicode (UTF-8, UTF-16 ou UTF-32), 4D tente d'identifier une BOM parmi les premiers octets reçus. Si elle est détectée, elle est filtrée du résultat et 4D utilise le jeu de caractères qu'elle définit au lieu du jeu de caractères spécifié.

Communications

RECEVOIR PAQUET

RECEVOIR PAQUET({docRef ;} réceptVar ; stopCar | nbOctets)

La commande RECEVOIR PAQUET prend désormais en charge les BOM (*Byte Order Mark*) lorsque *réceptVar* est une variable de type chaîne.

Dans ce cas, si le jeu de caractères courant est de type Unicode (UTF-8, UTF-16 ou UTF-32), 4D tente d'identifier une BOM parmi les premiers octets reçus. Si elle est détectée, elle est filtrée de la variable de réception et 4D utilise le jeu de caractères qu'elle définit au lieu du jeu de caractères courant.

Conteneur de données

FIXER FICHIER DANS CONTENEUR

FIXER FICHIER DANS CONTENEUR (fichier*{; *}*)

Paramètres	Type		Description
fichier	Texte	→	*Nom du fichier ou* Chemin d'accès complet au fichier
*	*	→	Si passé = ajouter Si omis = remplacer

Pour plus de souplesse, vous pouvez passer directement un nom de fichier à la commande FIXER FICHIER DANS CONTENEUR (sans chemin complet).

En outre, la commande admet l'étoile * en paramètre optionnel. Par défaut, lorsque ce paramètre est omis, la commande remplace le contenu du conteneur de données par le dernier chemin d'accès défini par *fichier* (fonctionnement précédent de la commande).
Si vous passez ce paramètre, la commande ajoute le *fichier* au conteneur de données. Il peut ainsi contenir une "pile" de chemins d'accès de fichiers.

Dans les deux cas, si des données autres que des chemins d'accès étaient présentes dans le conteneur, elles sont effacées.

Définition structure

Deux nouvelles commandes utilitaires ont été ajoutées dans ce thème, permettant de récupérer les éventuelles données "fantômes" de la base.

LIRE NOMS TABLES MANQUANTES

LIRE NOMS TABLES MANQUANTES(tabManquantes)

Paramètres	Type		Description
tabManquantes	Tableau texte	←	Noms des tables manquantes dans la base

La nouvelle commande LIRE NOMS TABLES MANQUANTES retourne dans le tableau *tabManquantes* les noms de toutes les tables manquantes de la base courante.

Les tables manquantes sont des tables dont les données sont présentes dans le fichier de données mais qui n'existent pas au niveau de la structure courante. Ce cas se produit lorsqu'un fichier de données est ouvert avec des versions différentes de la structure.

Typiquement, le scénario est le suivant :

- le développeur fournit une structure contenant les tables A, B et C,
- l'utilisateur ajoute des tables personnalisées D et E à l'aide, par exemple, des commandes SQL intégrées de 4D, et stocke des données dans ces tables,
- le développeur fournit une nouvelle version de la structure. Elle ne contient pas les tables D et E.

Dans ce cas, la version utilisateur de la base contient toujours les données des tables D et E, mais elles ne sont plus accessibles. La commande LIRE NOMS TABLES MANQUANTES retournera les noms "D" et "E".

Une fois que vous avez identifié les tables manquantes de la base, vous pouvez les réactiver via la nouvelle commande REGENERER TABLE MANQUANTE.

Note Les données des tables manquantes sont effacées en cas de compactage du fichier de données (si les tables n'ont pas été regénérées entre-temps).

REGENERER TABLE MANQUANTE

REGENERER TABLE MANQUANTE(nomTable)

Paramètres	Type		Description
nomTable	Texte	→	Nom de table manquante à regénérer

La nouvelle commande REGENERER TABLE MANQUANTE reconstruit la table manquante dont vous avez passé le nom dans le paramètre *nomTable*. Lorsqu'une table manquante est reconstruite, elle devient visible dans l'éditeur de Structure et ses données sont de nouveau accessibles.

Les tables manquantes sont des tables dont les données sont présentes dans le fichier de données mais qui n'existent pas au niveau de la structure. Vous pouvez identifier les tables manquantes éventuellement présentes dans l'application à l'aide de la nouvelle commande LIRE NOMS TABLES MANQUANTES.

Si la table désignée par le paramètre *nomTable* n'est pas une table manquante de la base, la commande ne fait rien.

- Cette méthode regénère toutes les tables manquantes éventuellement présentes dans la base :

```
TABLEAU TEXTE($tMissingTables;0)
LIRE NOMS TABLES MANQUANTES($tMissingTables)
$SizeArray:=Taille tableau($tMissingTables)
Si ($SizeArray#0)
  // Remplir le tableau avec les noms de toutes les tables de la base
  TABLEAU TEXTE(tabTables;Lire numero derniere table)
  Si (Lire numero derniere table>0)    //S'il y a bien des tables
    Boucle ($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
  Fin de si
  Boucle ($i;1;$SizeArray)
    Si (Chercher dans tableau(tabTables;$tMissingTables{$i})=-1)
      CONFIRMER("Regénérer la table"+$tMissingTables{$i}+" ?")
      Si (OK=1)
        REGENERER TABLE MANQUANTE($tMissingTables{$i})
      Fin de si
    Sinon
      ALERTE("Impossible de régénérer la table "+$tMissingTables{$i}+" car
il y a déjà une table de ce nom dans la base.")
    Fin de si
  Fin de boucle
Sinon
  ALERTE("Pas de tables à regénérer.")
Fin de si
```

Référence : LIRE NOMS TABLES MANQUANTES

Documents système

La gestion des documents dans le cadre d'une interface multi-langue basée sur l'architecture xliff a été étendue dans 4D v12. Une nouvelle commande permet de récupérer le chemin d'accès de tout document intégré à cette architecture en fonction de la langue courante. Par ailleurs, deux nouvelles commandes facilitent la conversion des chemins d'accès système et posix. Un nouveau paramètre a été ajouté à la commande Selectionner dossier afin de vous faire bénéficier d'options supplémentaires.

Note Il est désormais possible de définir la langue courante de la base à l'aide de la nouvelle commande FIXER LANGUE BASE, placée dans le thème "Environnement 4D".

Lire chemin document localise

Lire chemin document localise (cheminRelatif) → Texte

Paramètres	Type		Description
cheminRelatif	Texte	→	Chemin d'accès relatif du document dont on veut obtenir la version localisée
Résultat	Texte	←	Chemin d'accès absolu du document localisé

La commande Lire chemin document localise retourne le chemin d'accès complet (absolu) d'un document désigné par *cheminRelatif* et situé dans un dossier xxx.lproj.

Cette commande doit être utilisée dans le cadre d'une architecture d'application multi-langue basée sur la présence d'un dossier Resources et de sous-dossiers xxx.lproj (xxx représentant une langue). Avec cette architecture, 4D prend automatiquement en charge les fichiers localisés de type .xliff ainsi que les images, mais vous pouvez avoir besoin d'utiliser le même mécanisme pour d'autres types de fichiers.

Passez dans *cheminRelatif* le chemin d'accès relatif du document recherché. Le chemin saisi doit être relatif au premier niveau d'un dossier "xxx.lproj" de la base. La commande retournera un chemin d'accès complet en utilisant le dossier "xxx.lproj" correspondant à la langue courante de la base.

Note La langue courante est définie soit automatiquement par 4D en fonction du contenu du dossier Resources (cf. commande Lire langue base), soit via la nouvelle commande FIXER LANGUE BASE.

Vous pouvez exprimer le contenu du paramètre *cheminRelatif* à l'aide d'une syntaxe posix ou système. Par exemple :

- xsl/log.xsl (syntaxe posix : utilisable sous Mac OS ou Windows)
- xsl\log.xsl (Windows uniquement)
- xsl:log.xsl (Mac OS uniquement)

Le chemin d'accès absolu retourné par la commande est toujours exprimé en syntaxe système.

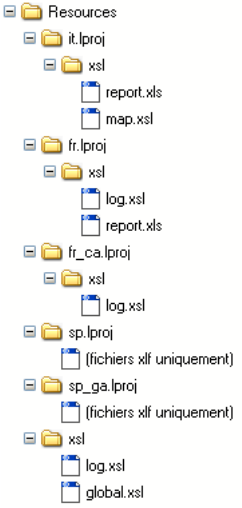
4D Server En mode distant, la commande retourne le chemin du dossier Resources sur le poste client si la commande est appelée depuis un process client.

4D recherche le fichier en respectant une séquence permettant de traiter tous les cas d'applications multi-langues. A chaque étape, 4D teste la présence de *cheminRelatif* dans le dossier correspondant à la langue et retourne le chemin complet en cas de succès. Si *cheminRelatif* n'est pas trouvé ou si le dossier n'existe pas, 4D passe à l'étape suivante. Voici les dossiers des étapes de recherche :

Langue courante (ex : fr-ca)
  Langue courante sans la région (ex : fr)
    Langue chargée par défaut au démarrage (ex : es-ga)
      Langue chargée par défaut au démarrage sans la région (ex : es)
        Premier dossier .lproj trouvé (ex : it.lproj)
          Premier niveau du dossier Resources

Si *cheminRelatif* n'est trouvé à aucun de ces emplacements, la commande retourne une chaîne vide.

- Dans le but de transformer un fichier xml en html, vous souhaitez utiliser un fichier de transformation "log.xsl". Ce fichier diffère suivant la langue courante. Vous souhaitez donc connaître le chemin du fichier "log.xsl" à utiliser. Voici le contenu du dossier Resources :



Pour utiliser un fichier .xsl adapté à la langue courante, il vous suffit d'écrire :

$monxsl:=**Lire chemin document localise** ("xsl/log.xsl")

Si la langue courante est, par exemple, le français canadien (fr-ca), la commande retourne :

- sous Windows : C:\users\…\…\…\resources\fr_ca.lproj\xsl\log.xsl"
- sous Mac OS : "HardDisk:users:…:…:…:resources:fr_ca.lproj:xsl:log.xsl"

Convertir chemin systeme vers POSIX

Convertir chemin systeme vers POSIX(cheminSystème{; *}) → Texte

Paramètres	Type		Description
cheminSystème	Texte	→	Chemin d'accès relatif ou absolu exprimé en syntaxe système
*	*	→	Option d'encodage
Résultat	Texte	←	Chemin d'accès absolu exprimé en syntaxe POSIX

La nouvelle commande Convertir chemin systeme vers POSIX convertit un chemin d'accès exprimé avec la syntaxe système en chemin d'accès exprimé avec la syntaxe POSIX (Unix).

Passez dans le paramètre *cheminSystème* le chemin d'accès à un fichier ou un dossier, exprimé avec la syntaxe système (Mac OS ou Windows). Ce chemin peut être absolu ou relatif au dossier de la base (dossier contenant le fichier de structure de la base). Il n'est pas obligatoire que les éléments du chemin existent réellement sur le disque au moment de l'exécution de la commande (la commande ne teste pas la validité du chemin d'accès).

La commande retourne le chemin d'accès complet du fichier ou du dossier exprimé dans la syntaxe POSIX. La commande retourne toujours un chemin d'accès absolu, quel que soit le type de chemin passé dans *cheminSystème*. Si vous avez passé un chemin relatif dans *cheminSystème*, 4D complète la valeur retournée en ajoutant le chemin d'accès au dossier de la base.

Le paramètre optionnel * permet de définir l'encodage du chemin POSIX. Par défaut, Convertir chemin systeme vers POSIX n'encode pas les caractères spéciaux du chemin POSIX. Si vous passez le paramètre *, les caractères seront traduits (par exemple, "Mon dossier" devient "Mon%20dossier").

- Exemple sous Mac OS

```
$chemin:=Convertir chemin systeme vers POSIX("machd:file 2.txt")
   `retourne "/Volumes/machd/file 2.txt" (même si machd est le disque de
   `démarrage)

$chemin:=Convertir chemin systeme vers POSIX("machd:file 2.txt";*)
   `retourne "/Volumes/machd/file%202.txt"

$chemin:=Convertir chemin systeme vers POSIX("resources:images")
   `retourne "/Volumes/machd/bases/basevideo/resources/images"
```

- Exemple sous Windows

```
$chemin:=Convertir chemin systeme vers POSIX("c:\docs\file 2.txt")
   `retourne "c:/docs/file 2.txt"

$chemin:=Convertir chemin systeme vers POSIX("\\srv\tempo\file.txt")
   `retourne "//srv/tempo/file.txt"
```

Référence : Convertir chemin POSIX vers systeme

Convertir chemin POSIX vers systeme

Convertir chemin POSIX vers systeme(cheminPosix{; *}) → Texte

Paramètres	Type		Description
cheminPosix	Texte	→	Chemin d'accès POSIX
*	*	→	Option d'encodage
Résultat	Texte	←	Chemin d'accès exprimé en syntaxe système

La nouvelle commande Convertir chemin POSIX vers systeme convertit un chemin d'accès exprimé avec la syntaxe POSIX (Unix) en chemin d'accès exprimé avec la syntaxe système.

Passez dans le paramètre *cheminPosix* le chemin d'accès complet à un fichier ou un dossier, exprimé avec la syntaxe POSIX. Ce chemin doit être absolu (il doit débuter par le caractère "/"). Vous devez passer un chemin disque, il n'est pas possible de passer de chemin réseau (débutant par exemple par ftp://ftp.monsite.fr).

La commande retourne le chemin d'accès complet du fichier ou du dossier exprimé dans la syntaxe du système courant.

Le paramètre optionnel * vous permet d'indiquer si le paramètre *cheminPosix* est encodé. Si c'est le cas, vous devez passer ce paramètre, sinon la conversion sera invalide. La commande retourne le chemin d'accès sans encodage.

▸ Exemples sous Mac OS

```
$chemin:=Convertir chemin POSIX vers systeme("/Volumes/machd/
            file 2.txt")
  `retourne "machd:file 2.txt"

$chemin:=Convertir chemin POSIX vers systeme("/Volumes/machd/
            file%202.txt";*)
  `retourne "machd:file 2.txt"

$chemin:=Convertir chemin POSIX vers systeme("/file 2.txt")
  `retourne "machd:file 2.txt" si machd est le disque de démarrage
```

▸ Exemples sous Windows

```
$chemin:=Convertir chemin POSIX vers systeme("c:/docs/file 2.txt")
  `retourne "c:\docs\truc 2.txt"

$chemin:=Convertir chemin POSIX vers systeme("c:/docs/
            file%202.txt";*)
  `retourne "c:\docs\truc 2.txt"
```

Référence : Convertir chemin systeme vers POSIX

Selectionner dossier

Selectionner dossier {(message{; répertoire*{; options}*})} → Texte

Paramètres	Type		Description
message	Texte	→	Titre de la fenêtre de sélection
répertoire	Texte | Entier long	→	Chemin d'accès ou Numéro de chemin d'accès mémorisé
options	*Entier long*	→	*Option(s) de sélection*
Résultat	Texte	←	Chemin d'accès au dossier sélectionné

La commande Selectionner dossier admet désormais le paramètre *options*. Ce paramètre vous permet de bénéficier de fonctions supplémentaires sous Mac OS.
Vous pouvez passer dans *options* une des constantes prédéfinies suivantes, placées dans le thème "Documents système" :

- Ouverture progiciel (2) : autoriser l'ouverture des progiciels (packages) en tant que dossiers et donc la visualisation/sélection de leur contenu. Par défaut, si cette constante n'est pas utilisée, la commande n'autorise pas l'ouverture des progiciels.
- Utiliser fenêtre feuille (16) : afficher la boîte de dialogue de sélection sous forme de fenêtre feuille. Les fenêtres feuilles sont des fenêtres spécifiques de l'interface Mac OS X, bénéficiant notamment d'une animation graphique. Par défaut, si cette constante n'est pas utilisée, la commande affiche une boîte de dialogue standard.

  Vous pouvez passer une constante ou une combinaison des deux constantes.

  Ces options sont prises en compte sous Mac OS uniquement. Sous Windows, le paramètre *options* est ignoré s'il est passé.

Environnement 4D

La nouvelle commande FIXER LANGUE BASE permet de modifier la langue courante de la base, utilisée notamment pour l'interface de l'application (architecture XLIFF). La commande Type version accepte de nouvelles valeurs permettant d'identifier 4D Server 64 bits. Par ailleurs, de nouveaux sélecteurs sont disponibles pour les commandes FIXER PARAMETRE BASE et Lire parametre base.

FIXER LANGUE BASE

FIXER LANGUE BASE(langue{; *})

Paramètres	Type		Description
langue	Texte	→	Sélecteur de langue
*	*	→	Portée de la commande

La nouvelle commande FIXER LANGUE BASE permet de modifier la langue courante de la base pour la session courante.

La langue courante de la base permet de définir le dossier .lproj dans lequel le programme va chercher les éléments localisés de l'application (textes et images). Par défaut, 4D détermine automatiquement la langue courante en fonction du contenu du dossier Resources et de l'environnement système (cf. description de la commande Lire langue base). FIXER LANGUE BASE vous permet de modifier la langue courante par défaut.

La commande ne modifie pas la langue des formulaires déjà chargés, seuls les éléments affichés postérieurement à l'appel de la commande tiendront compte du nouveau paramétrage.

Passez dans *langue* la langue à utiliser pour l'application, exprimée dans la norme définie par la RFC 3066, ISO639 et ISO3166. Typiquement, vous devez passer "fr" pour le français, "es" pour l'espagnol, "en-us" pour l'anglais américain, etc. Pour plus d'informations sur cette norme, reportez-vous au manuel *Mode Développement*.

Par défaut, la commande s'applique à toutes les bases et composants ouverts, pour tous les process. Le paramètre optionnel *, s'il est passé, spécifie que la commande doit s'appliquer uniquement à la base qui l'a appelée. Ce paramètre permet en particulier de définir séparément la langue de la base et celle d'un composant.

Si la commande est correctement exécutée, la variable système *OK* prend la valeur 1, sinon elle prend la valeur 0.

Note Conformément à la RFC, la commande utilise le "-" (tiret) pour séparer le code langue et le code région, par exemple "fr-ca" ou "en-us". En revanche, les dossiers "lproj" du dossier Resources utilisent le "_" (soulignement), par exemple "fr_ca.lproj" ou "en_us.lproj".

4D Server Avec 4D Server, les langues disponibles sont celles situées sur le poste distant ayant appelé la commande. Vous devez donc veiller à la synchronisation des dossiers Resources.

- Nous souhaitons définir la langue de l'interface en français :

  FIXER LANGUE BASE("fr")

- L'interface de votre application utilise la chaîne statique ":xliff:shopping". Les fichiers xliff contiennent notamment les informations suivantes :
  - Dossier FR :

    ```
    <trans-unit id="15" resname="Shopping">
        <source>Shopping</source>
        <target>Faire les courses</target>
    </trans-unit>
    ```

  - Dossier FR_CA :

    ```
    <trans-unit id="15" resname="Shopping">
        <source>Shopping</source>
        <target>Magasiner</target>
    </trans-unit>
    ```

  FIXER LANGUE BASE("fr")
  `la chaîne ":xliff:shopping" affiche "Faire les courses"

  FIXER LANGUE BASE("fr-ca")
  `la chaîne ":xliff:shopping" affiche "Magasiner"

  Référence : Lire langue base

Lire fragmentation table

Lire fragmentation table (laTable) → Réel

Paramètres	Type		Description
laTable	Table	→	Table de laquelle connaître le taux de fragmentation
Résultat	Réel	←	Pourcentage de fragmentation

La nouvelle commande Lire fragmentation table retourne le pourcentage de fragmentation logique des enregistrements de la table désignée par le paramètre *laTable*.

Le taux de fragmentation logique des enregistrements indique si les enregistrements sont stockés de manière ordonnée dans le fichier de données. Une fragmentation trop élevée peut ralentir sensiblement les tris et les recherches séquentiels sur la table. Un pourcentage de fragmentation de 0 correspond à une fragmentation nulle. Au-delà de 20 %, il peut être intéressant de procéder au compactage du fichier de données.

- Cette méthode de maintenance permet de demander le compactage du fichier de données en cas de fragmentation importante d'au moins une table de la base :

```
ACompacter :=Faux
Boucle ($i ;1;Lire numero derniere table)
  Si(Est un numero de table valide($i))
    Si(Lire fragmentation table(Table($i)->)>20)
      ACompacter :=Vrai
    Fin de si
  Fin de si
Fin de boucle
Si(ACompacter)
  // Poser un marqueur de demande de compactage
Fin de si
```

Lire langue base

Lire langue base *{(typeLangue)}* → Texte

Paramètres	Type		Description
typeLangue	Entier long	→	Type de langue
Résultat	Texte	←	Code de la langue utilisée

Note de compatibilité La commande Lire langue base était nommée Lire langue courante base dans les versions précédentes de 4D.

La commande Lire langue base accepte désormais un nouveau paramètre permettant d'indiquer le type de langue à obtenir. En effet, plusieurs paramétrages de langues différents peuvent être utilisés simultanément dans l'application.

Passez dans *typeLangue* une des constantes suivantes, placées dans le thème "Environnement 4D" :

***typeLangue* (valeur)**	**Description**
Langue par défaut (0)	Langue définie automatiquement par 4D au démarrage en fonction du dossier Resources et de l'environnement système (non modifiable)
Langue courante (1)	Langue courante de l'application : langue par défaut ou langue définie via la commande FIXER LANGUE BASE
Langue système (2)	Langue définie par l'utilisateur courant du système

Langue 4D interne (3)	Langue utilisée par 4D pour les tris et les comparaisons de textes (définie dans les Préférences de l'application)

Référence : FIXER LANGUE BASE

Type version

Type version → Entier long

Paramètres	Type		Description
Résultat	Entier long	←	0 = Version standard *32 bits* 1 = Version de démonstration *32 bits* *2 = Version 64 bits*

La commande Type version peut retourner une valeur supplémentaire, permettant d'identifier la version 64 bits de 4D Server. Cette valeur est accessible via une nouvelle constante du thème "Environnement 4D" :

Constante	Type	Valeur
Version 64 bits	Entier long	2

FIXER PARAMETRE BASE, Lire parametre base

FIXER PARAMETRE BASE ({table; }sélecteur; valeur)

Paramètres	Type		Description
table	Table	→	Table à paramétrer
sélecteur	Entier long	→	Code du paramètre de la base à modifier
valeur	*Réel* | Chaîne	→	Valeur du paramètre

Lire parametre base ({table; }sélecteur{; valeurAlpha}) → *Réel*

Paramètres	Type		Description
table	Table	→	Table du paramètre
sélecteur	Entier long	→	Code du paramètre de la base
valeurAlpha	Alpha	←	Valeur texte du paramètre
Résultat	*Réel*	←	Valeur du paramètre

- Le type du paramètre *valeur* et du retour de la fonction Lire parametre base ont été modifiés : il s'agit désormais de réels. Cette modification a été rendue nécessaire pour la prise en charge des nouveaux espaces de valeurs utilisables avec 4D Server 64 bits relatives à l'adressage de la mémoire (cf. paragraphe "4D Server Windows 64-bits", page 11).

- Le **sélecteur 34** (Enreg événements debogage) admet une nouvelle *valeur* : ce sélecteur admet désormais la valeur 3, activant le "mode détaillé avec valeurs". Dans ce mode avancé, les valeurs des paramètres passés aux commandes, méthodes projet et commandes de plug-ins sont également inscrites dans le fichier de débogage.
- Les nouveaux sélecteurs suivants sont disponibles.

  Sélecteur = 55 (PHP Adresse IP interpréteur)

  - *Valeurs* : Chaîne formatée du type "nnn.nnn.nnn.nnn" (par exemple "127.0.0.1").
  - *Description* : Adresse IP utilisée localement par 4D pour communiquer avec l'interpéteur PHP via fastcgi. Par défaut, la valeur est "127.0.0.1". Cette adresse doit correspondre à la machine sur laquelle se trouve 4D. Ce paramètre peut également être défini globalement pour tous les postes via les Propriétés de la base (cf. paragraphe “PHP”, page 41).
    Pour plus d'informations sur l'interpréteur PHP de 4D, reportez-vous au paragraphe “Exécuter des scripts PHP dans 4D”, page 65.

  Sélecteur = 56 (PHP Port interpréteur)

  - *Valeurs* : Valeur de type entier long positif.
  - *Description* : Numéro du port TCP utilisé par l'interpréteur PHP de 4D. Par défaut, la valeur est 8002.

  Sélecteur = 57 (PHP Nombre enfants)

  - *Valeurs* : Valeur de type entier long positif.
  - *Description* : Nombre de process enfants à créer et à maintenir localement par l'interpréteur PHP de 4D. Par défaut, la valeur est 5. Pour des raisons d'optimisation, l'interpréteur PHP crée et utilise un ensemble (pool) de process système appelés "process enfants" pour traiter les demandes d'exécution de scripts. Vous pouvez faire varier le nombre de process enfants en fonction des besoins de votre application. Pour plus d'informations, reportez-vous au paragraphe “Architecture”, page 65.

Note Sous Mac OS, tous les process enfants partagent le même port. Sous Windows, chaque process enfant utilise un numéro de port spécifique. Le premier numéro est celui défini pour l'interpréteur PHP, les autres process enfants l'incrémentent. Par exemple, si le port par défaut est le 8002 et que vous lancez 5 process enfants, ils utiliseront les ports 8002 à 8006.

Note Ce paramètre peut également être défini globalement pour tous les postes via les Propriétés de la base (cf. paragraphe “PHP”, page 41).

Sélecteur = 58 (PHP Nombre requêtes max)

- *Valeurs* : Valeur de type entier long positif.
- *Description* : Nombre maximum de requêtes acceptées par l’interpréteur PHP. Par défaut, la valeur est 500. Lorsque ce nombre maximum est atteint, l’interpréteur retourne des erreurs du type "serveur occupé". Pour des raisons de sécurité ou de performance, vous pouvez modifier cette valeur. Pour plus d’informations sur ce paramètre, reportez-vous à la documentation de fastcgi-php.

Note Côté 4D, ces paramètres sont appliqués dynamiquement, il n’est pas nécessaire de quitter le programme pour leur prise en compte. En revanche, si l’interpréteur PHP était déjà lancé, il est nécessaire de le quitter et de le relancer pour qu’il prenne en compte ces modifications.

Note Ce paramètre peut également être défini globalement pour tous les postes via les Propriétés de la base (cf. paragraphe “PHP”, page 41).

Sélecteur = 60 (PHP Utiliser interpréteur externe)

- *Valeurs* : 0 = utiliser interpréteur interne, 1 = utiliser interpréteur externe
- *Description* : Valeur indiquant si les requêtes PHP de 4D sont adressées à l’interpréteur interne fourni par 4D ou un interpréteur externe. Par défaut, la valeur est 0 (utilisation de l’interpréteur fourni par 4D). Si vous souhaitez utiliser votre propre interpréteur PHP, par exemple pour bénéficier de modules supplémentaires ou d’une configuration spécifique, passez 1 dans *valeur*. Dans ce cas, 4D ne démarrera pas son interpréteur en cas de requête PHP. L’interpréteur PHP personnalisé doit avoir été compilé en fastcgi et se trouver sur la même machine que le moteur 4D. A noter que dans ce cas, vous devez entièrement gérer l’interpréteur, il n’est ni démarré ni stoppé par 4D.

Note Ce paramètre peut également être défini globalement pour tous les postes via les Réglages de la base (cf. paragraphe “PHP”, page 41).

Sélecteur = 66 (Taille minimum libération cache)

- *Valeurs* : Valeur de type entier long positif > 1.
- *Description* : Taille minimum de mémoire à libérer du cache de la base de données lorsque le moteur a besoin d'y faire de la place pour y allouer un objet (valeur en octets).
  Ce sélecteur a pour but de permettre de réduire le nombre de libérations de données du cache afin d'obtenir des gains de performances. Vous pouvez faire varier ce paramétrage en fonction de la taille du cache et de celle des blocs de données manipulées dans votre base.
  Par défaut, si ce sélecteur n'est pas utilisé, 4D décharge au minimum 10 % du cache en cas de besoin de place.

OUVRIR PREFERENCES 4D

OUVRIR PREFERENCES 4D (sélecteur*{; accès}*)

Paramètres	Type		Description
sélecteur	Chaîne	→	Clé désignant un thème ou une page de paramètres de la boîte de dialogue des *Préférences ou des Propriétés de la base*
accès	*Booléen*	→	*Vrai=Verrouiller les autres pages de la boîte de dialogue* *Faux ou omis=Laisser actives les autres pages de la boîte de dialogue*

Le fonctionnement des Préférences ayant été entièrement remodelé dans 4D v12 (cf. paragraphe "Réorganisation des préférences", page 31), la liste des clés à passer dans le paramètre *sélecteur* a été modifiée. Par ailleurs, la commande OUVRIR PREFERENCES 4D accepte désormais le paramètre supplémentaire *accès*.

Désormais, deux boîtes de dialogue distinctes permettent de définir séparément les paramètres relatifs à l'application 4D (Préférences) et les paramètres relatifs à la base ouverte (Propriétés de la base). La commande OUVRIR PREFERENCES 4D vous permet d'accéder à ces deux boîtes de dialogue. A noter que lorsque ces boîtes de dialogues sont appelées via cette commande, elles sont modales.

Voici la liste des clés désormais utilisables dans le paramètre *sélecteur*. La première partie de la clé indique la boîte de dialogue concernée :
/4D
/4D/General
/4D/Structure

/4D/Form editor
/4D/Method editor
/4D/Client-Server
/4D/Shortcuts
/Database
/Database/General
/Database/Mover
/Database/Interface
/Database/Interface/Developper
/Database/Interface/User
/Database/Interface/Shortcuts
/Database/Compiler
/Database/Database
/Database/Database/Data storage
/Database/Database/Memory and cpu
/Database/Database/International
/Database/Backup
/Database/Backup/Scheduler
/Database/Backup/Configuration
/Database/Backup/Backup and restore
/Database/Client-Server
/Database/Client-Server/Network
/Database/Client-Server/IP configuration
/Database/Web
/Database/Web/Config
/Database/Web/Options 1
/Database/Web/Options 2
/Database/Web/Log format
/Database/Web/Log scheduler
/Database/Web/Webservices
/Database/SQL
/Database/php
/Database/Compatibility
/Database/Security

Si vous passez une clé invalide ou uniquement une barre oblique ("/"), la commande affiche la boîte de dialogue des Propriétés de la base, ouverte sur la dernière page consultée.

Note de compatibilité La commande continue de fonctionner avec les clés des versions précédentes, la correspondance est établie automatiquement par 4D. Il est toutefois conseillé de remplacer les anciens appels par des clés v12.

Le nouveau paramètre *accès* vous permet de contrôler les actions de l'utilisateur dans la boîte de dialogue des Préférences ou des Propriétés de la base en verrouillant les autres pages. Typiquement, vous pouvez souhaiter laisser l'utilisateur personnaliser certains paramètres, mais éviter que les autres puissent être modifiés. Dans ce cas, passez Vrai dans le paramètre *accès* : seule la page désignée par le paramètre *sélecteur* sera active et modifiable, l'accès à toutes les autres pages sera verrouillé (les clics sur les boutons de la barre de navigation seront sans effet). Si vous passez Faux ou omettez le paramètre *accès*, toutes les pages de la boîte de dialogue seront accessibles sans restriction.

Evenements formulaire

APPELER CONTENEUR SOUS FORMULAIRE

APPELER CONTENEUR SOUS FORMULAIRE (événement)

Paramètres	Type	Description
événement	Entier long →	Evénement à transmettre

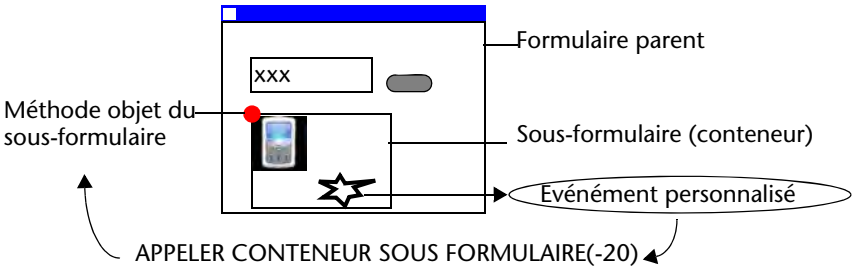
La nouvelle commande APPELER CONTENEUR SOUS FORMULAIRE permet à une instance de sous-formulaire d'envoyer l'*événement* à l'objet conteneur de sous-formulaire qui la contient. L'objet sous-formulaire peut alors traiter l'*événement* dans le contexte du formulaire parent.

Cette commande doit être placée dans la méthode formulaire du sous-formulaire ou dans la méthode objet d'un des objets du sous-formulaire. L'événement sera reçu uniquement dans la méthode objet du conteneur du sous-formulaire.

Vous pouvez passer dans *événement* tout événement formulaire prédéfini de 4D disponible pour un objet conteneur de sous-formulaire (vous pouvez utiliser les constantes du thème "Evenements formulaires") ou toute valeur correspondant à un événement personnalisé. Dans le premier cas, l'événement doit être coché pour l'objet conteneur du sous-formulaire.

Dans le cas d'un événement personnalisé, il est conseillé de passer une valeur négative dans *événement* afin de ne pas risquer d'interférer avec des numéros d'événements existants ou à venir de 4D.

Exemple d'exécution de la commande APPELER CONTENEUR SOUS FORMULAIRE



Pour plus d'informations sur les mécanismes des sous-formulaires dans 4D v12, reportez-vous au paragraphe "Extension des capacités des sous-formulaires", page 73.

Formulaires

Pour faciliter leur utilisation, les commandes de ce thème ont été renommées et deux commandes complémentaires ont été ajoutées.

Commandes renommées

Pour des raisons d'harmonisation et de standardisation, les commandes du thème "Formulaires" existantes ont été renommées dans 4D v12. Le préfixe "FORM" a été systématiquement ajouté. Le fonctionnement de ces commandes est inchangé.

Nouveau nom 4D v12	Ancien nom
FORM ALLER A PAGE	*ALLER A PAGE*
FORM DERNIERE PAGE	*DERNIERE PAGE*
FORM FIXER ENTREE	*FORMULAIRE ENTREE*
FORM FIXER SORTIE	*FORMULAIRE SORTIE*
FORM FIXER REDIMENSIONNEMENT HORIZONTAL	*FIXER REDIMENSIONNEMENT FORMULAIRE HORIZONTAL*
FORM FIXER REDIMENSIONNEMENT VERTICAL	*FIXER REDIMENSIONNEMENT FORMULAIRE VERTICAL*
FORM FIXER TAILLE	*FIXER TAILLE FORMULAIRE*
FORM LIRE OBJETS	*LIRE OBJETS FORMULAIRE*
FORM Lire page courante	*Page formulaire courante*
FORM LIRE PARAMETRE	*LIRE PARAMETRE FORMULAIRE*
FORM LIRE PROPRIETES	*LIRE PROPRIETES FORMULAIRE*

FORM PAGE PRECEDENTE	*PAGE PRECEDENTE*
FORM PAGE SUIVANTE	*PAGE SUIVANTE*
FORM PREMIERE PAGE	*PREMIERE PAGE*

FORM LIRE REDIMENSIONNEMENT HORIZONTAL

FORM LIRE REDIMENSIONNEMENT HORIZONTAL (redimension{; largeurMini{; largeurMaxi}})

Paramètres	Type		Description
redimension	Booléen	←	Vrai : le formulaire est redimensionnable horizontalement Faux : le formulaire n'est pas redimensionnable horizontalement
largeurMini	Entier long	←	Largeur minimale du formulaire (pixels)
largeurMaxi	Entier long	←	Largeur maximale du formulaire (pixels)

La nouvelle commande FORM LIRE REDIMENSIONNEMENT HORIZONTAL retourne dans les variables *redimension*, *largeurMini* et *largeurMaxi* les propriétés de redimensionnement horizontal du formulaire courant. Ces propriétés peuvent avoir été définies pour le formulaire dans l'éditeur de formulaires en mode Développement ou pour le process courant via la commande FORM FIXER REDIMENSIONNEMENT HORIZONTAL.

FORM LIRE REDIMENSIONNEMENT VERTICAL

FORM LIRE REDIMENSIONNEMENT VERTICAL (redimension{; hauteurMini{; hauteurMaxi}})

Paramètres	Type		Description
redimension	Booléen	←	Vrai : le formulaire est redimensionnable verticalement Faux : le formulaire n'est pas redimensionnable verticalement
hauteurMini	Entier long	←	Hauteur minimale du formulaire (pixels)
hauteurMaxi	Entier long	←	Hauteur maximale du formulaire (pixels)

La nouvelle commande FORM LIRE REDIMENSIONNEMENT VERTICAL retourne dans les variables *redimension*, *hauteurMini* et *hauteurMaxi* les propriétés de redimensionnement vertical du formulaire courant. Ces propriétés peuvent avoir été définies pour le formulaire dans l'éditeur de formulaires en mode Développement ou pour le process courant via la commande FORM FIXER REDIMENSIONNEMENT VERTICAL.

Gestion de la saisie

ALLER A OBJET

ALLER A OBJET({*;}objet)

Note de compatibilité La commande ALLER A OBJET était nommée ALLER A CHAMP dans les versions précédentes de 4D.

La commande ALLER A OBJET peut désormais être utilisée dans le contexte d'un sous-formulaire. Lorsqu'elle est appelée depuis un sous-formulaire, elle recherche en premier lieu *objet* dans le sous-formulaire puis, si la recherche n'aboutit pas, elle étend la recherche aux objets du formulaire parent.

Pour plus d'informations sur les nouveautés des sous-formulaires dans 4D v12, reportez-vous au paragraphe "Extension des capacités des sous-formulaires", page 73.

Glisser-déposer

Position deposer

Position deposer {(numColonne | *posXImage*)}→ Numérique

Paramètres	Type		Description
numColonne | *posXImage*	Entier long	←	Numéro de colonne de list box ou -1 *Ou* *Position coordonnée X dans l'image*
Résultat	Numérique	←	• Numéro (tableau/list box) ou • Position (liste hiérarchique) ou • Position dans la chaîne (texte/combo box) de l'élément de destination ou -1 si le déposer a lieu après le dernier élément de tableau ou de liste ou • *Position coordonnée Y dans l'image*

La commande Position deposer fonctionne désormais dans le contexte d'un événement de glisser-déposer se produisant dans un champ image ou une variable image.

Dans ce cas, la commande retourne l'emplacement vertical du clic et, dans le paramètre facultatif *posXImage*, l'emplacement horizontal du clic. Les valeurs retournées sont exprimées en pixels et relativement au système de coordonnées locales.

Référence : OBJET LIRE DEFILEMENT, OBJET FIXER DEFILEMENT

Images

Ce thème contient plusieurs nouvelles commandes permettant de tester les fichiers image (Est un fichier image) et de manipuler les métadonnées (FIXER METADONNEES IMAGE et LIRE METADONNEES IMAGE). Les commandes CONVERTIR IMAGE et LISTE CODECS IMAGES admettent des paramètres supplémentaires. Trois commandes obsolètes ont été renommées.

Nouvelles API pour l'encodage et le décodage des images

4D v12 utilise de nouvelles API natives pour encoder et décoder les images (champs et variables) sous Windows et Mac OS. Ces implémentations donnent accès à des formats natifs supplémentaires, dont le format RAW, couramment utilisé par les appareils photo numériques.

- sous Windows, 4D utilise désormais WIC (*Windows Imaging Component*). WIC prend en charge nativement les formats BMP, PNG, ICO (décodage seulement), JPEG, GIF, TIFF et WDP (Microsoft Windows Digital Photo).
  Il est possible d'utiliser des formats supplémentaires tels que JPEG-2000 en installant des codecs WIC tiers.
- sous Mac OS, 4D utilise désormais ImageIO. Tous les codecs ImageIO disponibles sont donc pris en charge nativement pour le décodage (lecture) ainsi que l'encodage (écriture) :

Décodage	Encodage
public.jpeg	public.jpeg
com.compuserve.gif	com.compuserve.gif
public.png	public.png
public.jpeg-2000	public.jpeg-2000
com.nikon.raw-image	public.tiff
com.pentax.raw-image	com.adobe.photoshop.image
com.sony.arw-raw-image	com.adobe.pdf
com.adobe.raw-image	com.microsoft.bmp

public.tiff com.canon.crw-raw-image com.canon.cr2-raw-image com.canon.tif-raw-image com.sony.raw.image com.olympus.raw-image com.konicaminolta.raw-image com.panasonic.raw-image com.fuji.raw-image com.adobe.photoshop-image com.adobe.illustrator.ai-image com.adobe.pdf com.microsoft.ico com.microsoft.bmp com.truevision.tga-image com.sgi.sgi-image com.apple.quicktime-image com.apple.icns com.apple.pict com.apple.macpaint-image com.kodak.flashpix-image public.xbitmap-image com.ilm.openexr-image public.radiance	com.truevision.tga-image com.sgi.sgi-image com.apple.pict com.ilm.openexr-image

Sous Windows comme sous Mac OS, les formats pris en charge varient en fonction du système d'exploitation et des codecs personnalisés installés sur les postes. Pour connaître les codecs disponibles, vous devez utiliser la commande LISTE CODECS IMAGES.

A noter que WIC et ImageIO permettent l'utilisation de métadonnées dans les images. Deux nouvelles commandes, FIXER METADONNEES IMAGE et LIRE METADONNEES IMAGE ont été ajoutées afin que vous puissiez en bénéficier dans vos développements.

Est un fichier image

Est un fichier image (cheminFichier{; *}) → Booléen

Paramètres	Type		Description
cheminFichier	Texte	→	Chemin d'accès de fichier
*	*	→	Valider les données
Résultat	Booléen	←	Vrai = cheminFichier désigne un fichier image, sinon Faux

La nouvelle commande Est un fichier image teste le fichier désigné par le paramètre *cheminFichier* et retourne Vrai s'il s'agit d'un fichier image valide. La commande retourne Faux si le fichier n'est pas de type image ou s'il n'a pas été trouvé.

Passez dans le paramètre *cheminFichier* le chemin d'accès du fichier image à tester. Ce chemin doit être exprimé avec la syntaxe système. Vous pouvez passer un chemin d'accès absolu ou relatif au fichier de structure de la base. Si vous passez une chaîne vide (""), la commande retourne Faux.

Si vous ne passez pas le paramètre *, la commande teste le fichier en recherchant son extension parmi la liste des codecs disponibles. Si vous souhaitez pouvoir tester des fichiers sans extension ou effectuer une vérification plus complète, passez le paramètre *. Dans ce cas, la commande effectue des analyses supplémentaires : elle charge et inspecte l'en-tête du fichier et interroge les codecs afin de valider l'image. Cette syntaxe ralentit l'exécution de la commande.

Note La commande retourne Vrai pour les fichiers PDF sous Windows et les fichiers EMF sous Mac OS.

FIXER METADONNEES IMAGE

FIXER METADONNEES IMAGE (image; nomMeta1; contenu1{ ;...; nomMetaN; contenuN})

Paramètres	Type		Description
image	Image	→	Image dont vous souhaitez écrire les métadonnées
nomMeta1...N	Texte	→	Nom ou chemin du bloc à écrire
contenu1...N	Variable	→	Contenu de la métadonnée

La nouvelle commande FIXER METADONNEES IMAGE permet d'écrire ou de modifier le contenu de métadonnées (ou méta-balises) présentes dans *image* (champ ou une variable image 4D).

Les métadonnées sont des informations supplémentaires insérées dans les images. 4D permet de manipuler quatre types de métadonnées standard : EXIF, GPS, IPTC et TIFF.

Note Pour une description détaillée de ces types de metadonnées, vous pouvez consulter les documents suivants : http://www.iptc.org/std/IIM/4.1/specification/IIMV4.1.pdf (IPTC) et http://exif.org/Exif2-2.PDF (TIFF, EXIF et GPS).

Passez dans le paramètre *nomMeta* une chaîne désignant le type de métadonnée à écrire ou à modifier. Vous pouvez passer :

- une des constantes du nouveau thème "Noms des métadonnées images". Ce thème regroupe toutes les balises prises en charge par 4D. Chaque constante contient un chemin de balise (par exemple "TIFF/DateTime"),
- le nom d'un bloc complet de métadonnées ("TIFF", "EXIF", "GPS" ou "IPTC"),
- une chaîne vide ("").

Passez dans le paramètre *contenu* les nouvelles valeurs de la métadonnée :

- Si vous avez passé une constante de chemin de balise dans *nomMeta*, passez directement dans *contenu* la valeur à écrire ou l'une des constantes appropriées du nouveau thème "Valeurs des métadonnées images". La valeur peut être de type texte, entier long, réel, date ou heure, en fonction de la métadonnée désignée. Vous pouvez utiliser un tableau si la métadonnée contient plus d'une valeur. Si vous passez une chaîne, elle doit être formatée en XML (norme XMP). Passez une chaîne vide ("") pour effacer la métadonnée si elle existe.
- Si vous avez passé un nom de bloc ou une chaîne vide dans *nomMeta*, passez dans *contenu* la référence XML DOM de l'élément contenant les métadonnées à écrire. Dans le cas d'une chaîne vide, toutes les métadonnées seront modifiées.

Note Du fait de leur nombre important, les constantes des nouveaux thèmes "Noms des métadonnées images" et "Valeurs des métadonnées images" sont décrites dans l'annexe C, "Constantes de métadonnées," page 339.

Sous Windows, si une erreur se produit durant l'exécution de la commande, la variable *OK* prend la valeur 0.
A noter que sous Mac OS, pour des raisons techniques, les erreurs d'écriture des métadonnées ne sont pas détectées. La variable OK n'est pas modifiée par cette commande sous Mac OS.

- Ecriture de plusieurs valeurs de la métadonnée "Keywords" via des tableaux :

```
TABLEAU TEXTE($tTkeywords;2)
$tTkeywords{1}:="france"
$tTkeywords{2}:="europe"
FIXER METADONNEES IMAGE(vImage;IPTC Keywords;$tTkeywords)
```

- Ecriture du bloc GPS via une référence DOM :

```
C_TEXTE($domMetas)
$domMetas:=DOM Analyser source XML("metas.xml")
C_TEXTE($refGPS)
$refGPS:=DOM Chercher element XML($domMetas;"Metadatas/GPS")
Si (OK=1)
  FIXER METADONNEES IMAGE(vImage;"GPS";$refGPS)
    //$refGPS pointe bien ici sur l'élement GPS
  ...
Fin de si
DOM FERMER XML($domMetas)
```

Note Lorsque toutes les métadonnées sont manipulées via une référence d'éléments DOM, les balises sont stockées comme attributs attachés à un élément (enfant de l'élément référencé) dont le nom est le nom du bloc (TIFF, IPTC, etc.). Lorsqu'un bloc de métadonnées spécifique est manipulé, les balises du bloc sont stockées comme attributs directement attachés à l'élement référencé par la commande.

Référence : LIRE METADONNEES IMAGE

LIRE METADONNEES IMAGE

LIRE METADONNEES IMAGE (image; nomMeta1; contenu1{ ;...; nomMetaN; contenuN})

Paramètres	Type		Description
image	Image	→	Image dont vous souhaitez lire les méta-données
nomMeta1...N	Texte	→	Nom ou chemin du bloc à lire
contenu1...N	Variable	←	Contenu de la métadonnée

La nouvelle commande LIRE METADONNEES IMAGE permet de lire le contenu de métadonnées (ou méta-balises) présentes dans *image* (champ ou une variable image 4D). Pour plus d'informations sur les métadonnées, reportez-vous à la description de la commande FIXER METADONNEES IMAGE.

Passez dans le paramètre *nomMeta* une chaîne désignant le type de métadonnée à récupérer. Vous pouvez passer :

- une constante du nouveau thème "Noms métadonnées images", contenant un chemin de balise,
- le nom d'un bloc complet de métadonnées ("TIFF", "EXIF", "GPS" ou "IPTC"),
- une chaîne vide ("").

Passez dans le paramètre *contenu* la variable destinée à recevoir les métadonnées.

- Si vous avez passé un chemin de balise dans *nomMeta*, *contenu* contient directement la valeur à lire. La valeur sera convertie dans le type de la variable. Les variables de type texte seront formatées en XML (norme XMP). Vous pouvez passer un tableau lorsque la métadonnée contient plus d'une valeur (c'est le cas notamment pour les balises IPTC Keywords)
- Si vous avez passé un nom de bloc ou une chaîne vide dans *nomMeta*, *contenu* doit être une référence d'élément DOM XML valide. Dans ce cas, le contenu du bloc désigné (ou de tous les blocs si vous avez passé une chaîne vide dans *nomMeta*) est recopié dans l'élément référencé.

Note Du fait de leur nombre important, les constantes des nouveaux thèmes "Noms métadonnées images" et "Valeurs métadonnées images" sont décrites dans l'annexe C, "Constantes de métadonnées," page 339.

La variable système OK retourne 1 si la récupération des métadonnées s'est bien passée, et 0 si une erreur se produit ou si au moins une des balises n'est pas trouvée. Dans tous les cas, les valeurs lisibles sont retournées.

- Utilisation d'arbres DOM :

```
$xml:=DOM Creer ref XML("Root") \\Création d'un arbre XML DOM

  \\Réception des métadonnées TIFF
$_Xml_TIFF:=DOM Creer element XML($xml;"/Root/TIFF")
LIRE METADONNEES IMAGE(vPicture;"TIFF";$_Xml_TIFF)

  \\Réception des métadonnées GPS
$_Xml_GPS:=DOM Creer element XML($xml;"/Root/GPS")
LIRE METADONNEES IMAGE(vPicture;"GPS";$_Xml_GPS)
```

- Utilisation de variables :

```
C_DATE($dateAsDate)
LIRE METADONNEES IMAGE("TIFF/DateTime";$dateAsDate)
  //retourne uniquement la date car $dateAsDate est de type Date

C_TEXTE($dateAsText)
LIRE METADONNEES IMAGE("TIFF/DateTime";$dateAsText)
  //retourne uniquement la date mais au format XML

C_ENTIER($urgency)
LIRE METADONNEES IMAGE(vImage;"IPTC/Urgency";$urgency)
```

- Réception de balises à valeurs multiples dans des tableaux :

```
TABLEAU TEXTE($tTkeywords;0)
LIRE METADONNEES IMAGE(vImage;"IPTC/Keywords";$tTkeywords)
```

Après exécution de la commande, *tTkeywords* contient par exemple :

```
$tTkeywords{1} = "france"
$tTkeywords{2} = "europe"
```

- Réception de balises à valeurs multiples dans une variable texte :

```
C_TEXTE($vTmots;0)
LIRE METADONNEES IMAGE(vImage;"IPTC/Keywords";$vTmots)
```

Après exécution de la commande, *vTmots* contient par exemple "france;europe".

Référence : FIXER METADONNEES IMAGE

CONVERTIR IMAGE

CONVERTIR IMAGE (image; codec*{; compression}*)

Paramètres	Type		Description
image	Image	→	Image à convertir
		←	Image convertie
codec	Chaîne	→	Identifiant de codec d'image
compression	*Réel*	→	*Qualité de compression*

La commande CONVERTIR IMAGE admet le nouveau paramètre optionnel *compression*, permettant de définir la qualité de compression à appliquer à l'image résultante lorsqu'un codec compatible est utilisé.

Passez dans *compression* une valeur entre 0 et 1 définissant la qualité de compression, 0 étant la qualité la plus médiocre (compression élevée) et 1 la meilleure qualité (compression faible). Ce paramètre est pris en compte uniquement lorsque le codec supporte la compression (par exemple JPEG ou HDPhoto) et est pris en charge par les APIs WIC et ImageIO. Par conséquent, il n'est pas utilisable avec les codecs gérés par QuickTime uniquement.

Note Pour plus d'informations sur les APIs de gestion d'image dans 4D v12, reportez-vous au paragraphe "Nouvelles API pour l'encodage et le décodage des images", page 145.

Par défaut, si vous omettez ce paramètre, la meilleure qualité est appliquée (*compression* = 1).

▶ Conversion d'une image avec une qualité de 60 % :

CONVERTIR IMAGE (vPicture;".JPG";0,6)

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
*	*	→	*Retourner la liste des codecs de lecture*

La commande LISTE CODECS IMAGES permet désormais de récupérer la liste des codecs utilisables pour décoder (lire) les images. Pour cela, il suffit passer le nouveau paramètre optionnel *. Par défaut, lorsque ce paramètre est omis, la commande retourne les codecs utilisables pour encoder (écrire) les images.

Les deux listes ne sont pas exclusives, certains codecs de lecture et d'écriture sont identiques. Les codecs destinés à l'encodage des images pourront généralement être utilisés pour le décodage. En revanche, les codecs de décodage ne permettent pas forcément l'encodage. Par exemple, le codec ".jpg" sera présent dans les deux listes, tandis que le codec ".xbmp" sera présent dans la liste des codecs de lecture mais dans celle d'écriture.

ECRIRE FICHIER IMAGE

ECRIRE FICHIER IMAGE (nomFichier ; image {; codec})

Le fonctionnement de la commande ECRIRE FICHIER IMAGE a été modifié afin de faciliter la manipulation et le stockage d'images en format natif.

Désormais, si vous omettez le paramètre *codec*, la commande tentera de déterminer le codec sur la base de l'extension du nom de fichier passé dans le paramètre *nomFichier*. Par exemple, si vous passez l'instruction : ECRIRE FICHIER IMAGE ("c:\dossier\photo.jpg";maphoto)

... la commande utilisera le codec JPEG pour stocker l'image.

Si l'extension utilisée ne correspond à aucun codec disponible, le fichier n'est pas enregistré et la variable système *OK* prend la valeur 0. Si vous ne passez ni *codec* ni extension de fichier, le fichier image est enregistré au format PICT.

Commandes renommées

Dans 4D v12, trois commandes du thème "Images" ont été préfixées "QT" afin d'indiquer qu'elles sont basées sur l'extension QuickTime d'Apple et sont obsolètes :

Nouveau nom 4D v12	Ancien nom
QT COMPRESSER IMAGE	*COMPRESSER IMAGE*
QT CHARGER ET COMPRESSER IMAGE	*CHARGER ET COMPRESSER IMAGE*
QT COMPRESSER FICHIER IMAGE	*COMPRESSER FICHIER IMAGE*

Ces commandes ne doivent plus être utilisées dans 4D v12. Elles sont conservées pour des raisons de compatibilité uniquement.

Import-export

Ce thème accueille les deux commandes **IMPORTER ODBC** et **EXPORTER ODBC**. Il s'agit des commandes auparavant nommées SQL IMPORTER et SQL EXPORTER et placées dans le thème SQL. Leur fonctionnement est inchangé. Cette réorganisation et ce changement de nom permettent de clarifier l'usage des commandes de ces deux thèmes.

Impressions

Les fonctions d'impression ont été renforcées sur deux axes dans 4D v12 :

- pour plus de personnalisation, les nouvelles commandes Imprimer objet et OUVRIR FORMULAIRE IMPRESSION viennent compléter l'ensemble des outils permettant de gérer l'impression des objets de formulaires,
- l'intégration du driver OpenSource PDFCreator et sa prise en charge via les commandes FIXER OPTION IMPRESSION et LIRE OPTION IMPRESSION facilitent les impressions PDF sous Windows.

Imprimer objet

Imprimer objet({*;}objet{;posX{;posY{;largeur{;hauteur}}}}) → Booléen

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)
posX	Entier long	→	Emplacement horizontal de l'objet
posY	Entier long	→	Emplacement vertical de l'objet
largeur	Entier long	→	Largeur de l'objet (pixels)
hauteur	Entier long	→	Hauteur de l'objet (pixels)
Résultat	Booléen	←	Vrai = objet entièrement imprimé, Faux sinon

La nouvelle commande Imprimer objet vous permet d'imprimer le ou les objet(s) de formulaire désigné(s) par les paramètres *objet* et *, à l'emplacement défini par les paramètres *posX* et *posY*.

La commande Imprimer objet permet d'imprimer uniquement les objets des formulaires projet. Avant d'appeler cette commande, vous devez désigner le formulaire projet contenant les objets à imprimer à l'aide de la nouvelle commande OUVRIR FORMULAIRE IMPRESSION.

Si vous passez le paramètre optionnel *****, vous indiquez que le paramètre *objet* est un nom d'objet (une chaîne de caractères). Si vous ne passez pas le paramètre *****, vous indiquez que *objet* est une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de variable (de type objet uniquement).

Les paramètres *posX* et *posY* définissent le point de départ de l'impression du ou des objet(s). Ces valeurs doivent être exprimées en pixels. Si ces paramètres sont omis, l'objet sera imprimé en fonction de son emplacement dans le formulaire.

Les paramètres *largeur* et *hauteur* vous permettent de définir la largeur et la hauteur de l'objet de formulaire. La commande Imprimer objet ne gère pas les objets de taille variable. Vous devez utiliser la commande OBJET LIRE TAILLE OPTIMALE pour prendre en charge la taille des objets. Vous pouvez également utiliser la commande OBJET LIRE TAILLE OPTIMALE pour connaître la taille la plus adéquate pour les objets contenant du texte. De même, Imprimer objet ne provoque pas de sauts de page automatiques. Vous devez les gérer en fonction de vos besoins.

Vous pouvez utiliser les commandes de 4D pour modifier à la volée les propriétés des objets (couleur, taille...).

La commande retourne Vrai si l'objet a été imprimé entièrement et Faux dans le cas contraire, c'est-à-dire si l'intégralité des données associées à l'*objet* n'a pas pu être imprimée dans le cadre imposé. Typiquement, la commande retourne Faux lors de l'impression d'une list box, si toutes les lignes de la list box n'ont pas pu être imprimées. Dans ce cas, il suffit d'appeler la commande Imprimer objet de façon répétée, jusqu'à ce qu'elle retourne Vrai : un mécanisme spécifique provoque automatiquement le défilement du contenu de l'objet après chaque appel.

Notes - Dans la version actuelle de 4D v12, seuls les objets de type list box bénéficient de ce mécanisme (la commande retourne toujours Vrai pour tous les autres types d'objets). Dans les prochaines versions de 4D, ce fonctionnement sera étendu à d'autres objets au contenu variable.

- La commande LISTBOX LIRE INFORMATIONS IMPRESSION permet de contrôler le statut de l'impression durant l'opération. Pour plus d'informations, reportez-vous au paragraphe "Impression des list box", page 119.

La commande Imprimer objet peut être utilisée uniquement dans le contexte d'une tâche d'impression préalablement ouverte avec la commande OUVRIR TACHE IMPRESSION. Si elle n'est pas appelée dans ce contexte, la commande ne fait rien. Plusieurs commandes Imprimer objet peuvent être appelées dans la même tâche d'impression.

Note Les listes hiérarchiques et les zones Web ne sont pas imprimables.

- Exemple d'impression de dix objets dans un formulaire :

```
PARAMETRES IMPRESSION
Si (OK=1)
  OUVRIR TACHE IMPRESSION
  Si (OK=1)
    OUVRIR FORMULAIRE IMPRESSION("PrintForm")
    x:=100
    y:=50
    LIRE ZONE IMPRESSION(hpaper;wpaper)
    Boucle ($i;1;10)
      OBJET LIRE TAILLE OPTIMALE(*;"Obj"+Chaine($i);bestwidth;
                                                        bestheight)
      $fin:=Imprimer objet(*;"Obj"+Chaine($i))
      y:=y+bestheight+15
      Si (y>hpaper)
        SAUT DE PAGE(>)
        y:=50
      Fin de si
    Fin de boucle
  Fin de si
  FERMER TACHE IMPRESSION
Fin de si
```

- Exemple d'impression d'une list box complète :

```
Repeter
  $fin:=Imprimer objet(*;"malistbox")
Jusque ($fin)
```

Référence : OUVRIR FORMULAIRE IMPRESSION

OUVRIR FORMULAIRE IMPRESSION

OUVRIR FORMULAIRE IMPRESSION(formulaire)

Paramètres	Type		Description
formulaire	Chaîne	→	Nom du formulaire projet à ouvrir pour l'impression ou Chaîne vide pour refermer le formulaire projet courant

La nouvelle commande OUVRIR FORMULAIRE IMPRESSION vous permet de charger le formulaire projet *formulaire* pour l'impression. Une fois chargé, ce formulaire devient le formulaire d'impression courant. Toutes les commandes de gestion des objets, et en particulier la nouvelle commande Imprimer objet, travaillent avec ce formulaire.

Si un formulaire d'impression avait déjà été chargé au préalable (via un appel précédent à la commande OUVRIR FORMULAIRE IMPRESSION), il est refermé et remplacé par formulaire. Vous pouvez ouvrir et refermer plusieurs formulaires projet dans la même session d'impression. Changer de formulaire d'impression via la commande OUVRIR FORMULAIRE IMPRESSION ne génère pas de saut de page. Il revient au développeur de gérer les sauts de page. Si vous passez une chaîne vide dans *formulaire*, le formulaire projet d'impression courant est refermé.

Seul l'événement formulaire "Sur chargement" est exécuté durant l'ouverture du formulaire projet. Les autres événements formulaire sont ignorés. L'événement formulaire "Sur libération" est exécuté à l'issue de l'impression.

Pour préserver la cohérence graphique des formulaires, il est conseillé d'appliquer la propriété d'apparence "Impression" sur toutes les plates-formes.

Le formulaire d'impression courant est automatiquement refermé lorsque la commande FERMER TACHE IMPRESSION est appelée.

Référence : Imprimer objet

Intégration du pilote PDFCreator sous Windows

La prise en charge des impressions PDF sous Windows a été étendue dans 4D v12. Le programme s'appuie sur le pilote *PDFCreator* pour vous proposer des fonctions d'impression PDF simples et fonctionnelles. Les commandes FIXER OPTION IMPRESSION et LIRE OPTION IMPRESSION ont été modifiées afin de tirer parti de ce pilote.

PDFCreator est un pilote gratuit (OpenSource) régi par la licence AFPL (Aladdin Free Public License). Pour plus d'informations sur cette licence, reportez-vous à l'adresse http://en.pdfforge.org/content/license

Note Sous Mac OS, l'impression PDF est prise en charge de façon native par le système.

Pour pouvoir utiliser le pilote *PDFCreator*, vous devez le télécharger et l'installer dans votre environnement, il n'est pas installé par défaut par 4D. Attention, la version de *PDFCreator* certifiée pour 4D v12 est la **version 0.9.9**. Vous pouvez télécharger cette version par exemple à l'adresse http://filehippo.com/download_pdfcreator/6821/
Vous devez disposer d'un accès Administrateur pour pouvoir l'installer.

Au cours de l'installation, une nouvelle imprimante virtuelle nommée par défaut "PDFCreator" est installée dans votre système. Vous pouvez changer ce nom si vous le souhaitez.

FIXER OPTION IMPRESSION

FIXER OPTION IMPRESSION (option; valeur1{; valeur2})

Paramètres	Type		Description
option	Entier long | *Chaîne*	→	Numéro d'option *ou Code d'option PDF*
valeur1	Entier long | Chaîne	→	Valeur 1 de l'option
valeur2	Entier long | Chaîne	→	Valeur 2 de l'option

La commande FIXER OPTION IMPRESSION a été modifiée de manière à tirer parti du pilote (*driver*) **PDFCreator** sous Windows. Pour pouvoir bénéficier de ces nouveautés, vous devez avoir installé ce pilote dans votre environnement 4D (pour plus d'informations, reportez-vous au paragraphe "Intégration du pilote PDFCreator sous Windows", page 158).

Le paramètre *option* admet désormais une valeur de type texte, dans laquelle vous pouvez passer un code d'option PDF. Le code d'option est constitué de deux parties, *TypeOption* et *NomOption*, assemblées sous la forme "*TypeOption*:*NomOption*".

Voici la description de ce code :

- *TypeOption* : indique si vous désignez une option native de PDFCreator ou une option d'administration PDF de 4D. Deux valeurs sont acceptées :
  - **PDFOptions** = option native
  - **PDFInfo** = option interne.
- *NomOption* : désigne l'option à définir (dépend de la valeur de *TypeOption*).
  - Si *TypeOption* = **PDFOptions**, vous pouvez passer dans *NomOption* l'une des nombreuses options natives de PDFCreator. Par exemple, l'option **UseAutosave** agit sur la sauvegarde automatique. Pour pouvoir modifier cette option, passez "PDFOptions:UseAutosave" dans le paramètre *option* et la valeur à utiliser dans le paramètre *valeur1*. Pour une description complète des options natives de PDFCreator, reportez-vous à la documentation livrée avec le pilote PDFCreator.
  - Si *TypeOption* = **PDFInfo**, vous pouvez passer dans *NomOption* un des sélecteurs spécifiques suivants :
    - **Reset print** : permet de réinitialiser le statut d'attente interne afin, notamment, de sortir d'une boucle infinie. Dans ce cas, *valeur1* n'est pas utilisé.
    - **Reset standard options** : permet de rétablir toutes les options de PDFCreator à leurs valeurs par défaut. Si une impression est en cours, les paramètres par défaut sont appliqués à l'issue de l'impression. Dans ce cas, *valeur1* n'est pas utilisé.
    - **Start** : permet de démarrer ou de stopper le spouler de PDFCreator. Passez 0 dans *valeur1* pour le stopper et 1 pour le démarrer.
    - **Reset options** : permet de réinitialiser toutes les options modifiées depuis le début de la session à l'aide de la commande FIXER OPTION IMPRESSION et du sélecteur **PDFOptions**.
    - **Version** : permet de lire le numéro de version courant du pilote PDFCreator. Ce sélecteur peut être utilisé uniquement avec la commande LIRE OPTION IMPRESSION. Le numéro est retourné dans le paramètre *valeur1*.
    - **Last error** : permet de lire la dernière erreur retournée par le pilote PDFCreator. Ce sélecteur peut être utilisé uniquement avec la commande LIRE OPTION IMPRESSION. Le numéro de l'erreur est retourné dans le paramètre *valeur1*.

- **Print in progress** : permet de savoir si 4D est en attente d'une impression de PDFCreator. Ce sélecteur peut être utilisé uniquement avec la commande LIRE OPTION IMPRESSION. Le paramètre *valeur1* retourne 1 si 4D est en attente de PDFCreator et 0 dans le cas contraire.
- **Job count** : permet de connaître le nombre de tâches en attente dans la file d'impression. Ce sélecteur peut être utilisé uniquement avec la commande LIRE OPTION IMPRESSION. Le nombre de tâches est retourné dans le paramètre *valeur1*.
- **Synchronous Mode** : permet de fixer le mode de synchronisation entre les requêtes d'impression envoyées par 4D et le pilote PDFCreator. Comme 4D ne peut pas obtenir d'information concernant le statut courant d'une tâche d'impression présente dans la file d'impression, une option permet de mieux contrôler l'exécution des tâches en ne les envoyant que lorsque le statut du pilote PDFCreator est "libre". Dans ce cas, 4D est synchronisé avec le pilote. Passez 0 dans *valeur1* pour que 4D envoie immédiatement les requêtes d'impression (valeur par défaut) et 1 pour que 4D soit synchronisé et attende que le pilote ait terminé sa tâche en cours avant d'en envoyer une autre.

Note Après chaque impression, 4D rétablit automatiquement les paramètres précédents du pilote PDFCreator afin d'éviter toute interférence avec les autres programmes utilisant PDFCreator.

▸ La méthode suivante configure le pilote PDF de manière à imprimer tous les enregistrements de la table dans le fichier C:\test\Test_PDF_*N* où N est le numéro de séquence de l'enregistrement :

```
FIXER IMPRIMANTE COURANTE(PDFCreator Nom imprimante)
    // Utiliser l'imprimante virtuelle installée par PDFCreator
FIXER OPTION IMPRESSION("PDFOptions:UseAutosave";1)
FIXER OPTION IMPRESSION("PDFOptions:UseAutosaveDirectory";1)
FIXER OPTION IMPRESSION("PDFOptions:AutosaveDirectory";
                                        "C:\\test\\")

TOUT SELECTIONNER([Table_1])
Boucle($i;1;Enregistrements trouves([Table_1]))
  FIXER OPTION IMPRESSION("PDFOptions:AutosaveFilename";
                                        "Test_PDF_"+Chaine($i))
  IMPRIMER ENREGISTREMENT([Table_1];*)
  ENREGISTREMENT SUIVANT([Table_1])
Fin de boucle
```

FIXER OPTION IMPRESSION("PDFInfo:Reset standard options";0)
//Réinitialisation des options du pilote PDFCreator

LIRE OPTION IMPRESSION

LIRE OPTION IMPRESSION (option; valeur1{; valeur2})

Paramètres	Type		Description
option	Entier long | *Chaîne*	→	Numéro d'option *ou Code d'option PDF*
valeur1	Entier long | Chaîne	←	Valeur 1 de l'option
valeur2	Entier long	←	Valeur 2 de l'option

La commande LIRE OPTION IMPRESSION accepte désormais une chaîne dans le paramètre *option*. Cette modification permet à la commande de gérer les options du pilote PDFCreator. Pour plus d'informations sur cette nouveauté, reportez-vous à la description de la commande FIXER OPTION IMPRESSION.

FIXER IMPRIMANTE COURANTE

FIXER IMPRIMANTE COURANTE (nomImpr)

La commande FIXER IMPRIMANTE COURANTE permet de désigner l'imprimante virtuelle installée par le pilote PDFCreator comme destination d'impression. 4D v12 s'appuie sur le pilote PDFCreator pour faciliter l'impression de documents PDF sous Windows (cf. paragraphe "Intégration du pilote PDFCreator sous Windows", page 158).

Pour imprimer un document PDF, passez dans le paramètre *nomImpr* le nom de l'imprimante virtuelle installée par le pilote PDFCreator.

Par défaut, le nom de l'imprimante virtuelle est "PDFCreator". Toutefois, ce nom peut avoir été modifié au moment de l'installation du pilote. Pour que 4D recherche et utilise automatiquement le nom de l'imprimante virtuelle, même s'il a été personnalisé, passez la nouvelle constante PDFCreator Nom imprimante dans *nomImpr*. Cette constante est placée dans le thème "Options d'impression".

Interface utilisateur

Ce thème accueille les nouvelles commandes OBJET Lire nom et OBJET Lire pointeur. Les commandes SELECTIONNER TEXTE et TEXTE SELECTIONNE acceptent désormais une syntaxe "objet". La commande DEFILER LIGNES a été renommée OBJET FIXER DEFILEMENT et transférée dans le thème "Propriétés des objets".

OBJET Lire nom

OBJET Lire nom {(sélecteur)} → Texte

Paramètres	Type		Description
sélecteur	Entier long	→	Catégorie d'objet
Résultat	Texte	←	Nom de l'objet

La nouvelle commande OBJET Lire nom retourne le nom d'un objet de formulaire.

La commande permet de désigner deux types d'objets en fonction de la valeur du paramètre *sélecteur*. Vous pouvez passer dans ce paramètre l'une des constantes suivantes (placées dans le nouveau thème "Objets de formulaire") :

- Objet courant ou *sélecteur* omis : Si vous passez ce sélecteur ou omettez le paramètre *sélecteur*, la commande retourne le nom de l'objet à partir duquel elle a été appelée (méthode objet ou sous-méthode appelée par la méthode objet). Dans ce cas, la commande doit être appelée dans le contexte d'un objet de formulaire, sinon elle retourne une chaîne vide.
- Objet avec focus : Si vous passez ce sélecteur, la commande retourne le nom de l'objet ayant le focus dans le formulaire.

▸ Méthode objet du bouton "bValiderForm" :

```
$nomBtn:=Objet Lire nom(Objet courant)
```

Après l'exécution de cette méthode objet, la variable *$nomBtn* contient la valeur "bValiderForm".

Référence : OBJET Lire pointeur

OBJET Lire pointeur

OBJET Lire pointeur *{(sélecteur{; nomObjet{; nomSousFormulaire}})}* → Pointeur

Paramètres	Type		Description
sélecteur	Entier long	→	Catégorie d'objet
nomObjet	Texte	→	Nom d'objet
nomSousFormulaire	Texte	→	Nom d'objet sous-formulaire
Résultat	Pointeur	←	Pointeur sur la variable de l'objet

La nouvelle commande OBJET Lire pointeur retourne un pointeur vers la variable d'un objet de formulaire.

Cette commande permet de désigner différents objets en fonction de la valeur du paramètre *sélecteur*. Vous pouvez passer dans ce paramètre l'une des constantes du nouveau thème "Objets de formulaire" :

- Objet courant ou *sélecteur* omis : Si vous omettez le paramètre *sélecteur* ou passez ce sélecteur, la commande retourne un pointeur vers la variable associée à l'objet courant (objet dont la méthode formulaire est en cours d'exécution).

Note Ce fonctionnement équivaut strictement à celui de la commande historique Self. La commande Self est toutefois conservée pour des raisons de compatibilité.

- Objet avec focus : Si vous passez ce sélecteur, la commande retourne un pointeur vers la variable associée à l'objet ayant le focus dans le formulaire. Les deux derniers paramètres optionnels sont ignorés s'ils sont passés.

Note Ce fonctionnement équivaut strictement à celui de la commande Objet focus. La commande Objet focus est désormais obsolète dans 4D v12.

- Objet conteneur sous formulaire : Si vous passez ce sélecteur, la commande retourne un pointeur vers la variable liée au conteneur du sous-formulaire. Les deux derniers paramètres optionnels sont ignorés s'ils sont passés. Ce sélecteur ne peut donc être utilisé que dans le contexte d'un formulaire utilisé comme sous-formulaire, afin d'accéder à la variable liée à l'objet conteneur. Pour plus d'informations, reportez-vous au paragraphe "Mise à jour du contenu du sous-formulaire", page 77.

- Objet nommé : Si vous passez ce sélecteur, vous devez également passer le deuxième paramètre, *nomObjet*. Dans ce cas, la commande retourne un pointeur vers la variable associée à l'objet dont vous avez passé le nom dans ce paramètre.
  Si *nomObjet* correspond à un sous-formulaire et que l'option "Sous-formulaire liste" est cochée, la commande retourne un pointeur vers la table du sous-formulaire si une table source est définie et Nil dans le cas contraire.

Le paramètre optionnel *nomSousFormulaire* vous permet de récupérer un pointeur vers un objet *nomObjet* n'appartenant pas au contexte courant, c'est-à-dire au formulaire parent. Pour pouvoir utiliser ce paramètre, vous devez avoir passé le *sélecteur* Objet nommé.
Lorsque le paramètre *nomSousFormulaire* est passé, la commande OBJET Lire pointeur recherche dans un premier temps l'objet sous-formulaire nommé *nomSousFormulaire* dans le formulaire courant, puis recherche à l'intérieur de ce sous-formulaire un objet nommé *nomObjet*. Si cet objet est trouvé, elle retourne un pointeur vers la variable de cet objet.

- Soit un formulaire "SF" utilisé deux fois comme sous-formulaire dans le même formulaire parent. Les objets sous-formulaires sont nommés "SF1" et "SF2". Le formulaire "SF" contient un objet nommé *ValeurCourante*. Dans l'événement "Sur chargement" de la méthode formulaire du formulaire parent, nous souhaitons initialiser l'objet *ValeurCourante* de SF1 à "Janvier" et celui de SF2 "Février" :

```
C_POINTEUR($Ptr)
$Ptr:=OBJET Lire pointeur(Objet nommé;"ValeurCourante";"SF1")
$Ptr->:="Janvier"
$Ptr:=OBJET Lire pointeur(Objet nommé;"ValeurCourante";"SF2")
$Ptr->:="Février"
```

Référence : OBJET Lire nom

SELECTIONNER TEXTE

SELECTIONNER TEXTE({*; }*objet;* débutSél; finSél)

Paramètres	Type		Description
*	*	→	*Si spécifié, objet est un nom d'objet (chaîne)* *Si omis, objet est un champ ou une variable*
objet	Objet de formulaire	→	Nom d'objet (si * est spécifié) ou Champ ou variable (si * est omis)
débutSél	Num	→	Nouvelle position de début de sélection de texte
finSél	Num	→	Nouvelle position de fin de sélection de texte

La commande SELECTIONNER TEXTE admet désormais une syntaxe de type "objet", c'est-à-dire désignant un objet de formulaire par son nom et non par sa référence de variable.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre *objet* désigne le nom d'un objet (une chaîne). Si vous ne passez pas le paramètre *, vous indiquez que le paramètre *objet* désigne un champ ou une variable. Dans ce cas, vous ne passez pas une chaîne de caractères mais la référence du champ ou de la variable (champs ou variables de formulaire uniquement).

TEXTE SELECTIONNE

TEXTE SELECTIONNE({*; }*objet;* débutSél; finSél)

Paramètres	Type		Description
*	*	→	*Si spécifié, objet est un nom d'objet (chaîne)* *Si omis, objet est un champ ou une variable*
objet	Objet de formulaire	→	Nom d'objet (si * est spécifié) ou Champ ou variable (si * est omis)
débutSél	Num	←	Position de début de sélection de texte
finSél	Num	←	Position de fin de sélection de texte

La commande TEXTE SELECTIONNE admet désormais une syntaxe de type "objet", c'est-à-dire désignant un objet de formulaire par son nom et non par sa référence de variable.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre *objet* désigne le nom d'un objet (une chaîne). Si vous ne passez pas le paramètre *, vous indiquez que le paramètre *objet* désigne un champ ou une variable. Dans ce cas, vous ne passez pas une chaîne de caractères mais la référence du champ ou de la variable (champs ou variables de formulaire uniquement).

- Cette nouvelle syntaxe facilite l'utilisation de la nouvelle commande OBJET FIXER ATTRIBUT TEXTE STYLE :

TEXTE SELECTIONNE(*;"monTexte";$debutsel,$finsel)
OBJET FIXER ATTRIBUT STYLE TEXTE(*;"monTexte";$debutsel,$finsel;
Attribut style souligné; 1; Attribut style gras;1)

Interruptions

Quatre nouvelles commandes prennent en charge les assertions *(asserts)* dans le code 4D. Ces nouveaux outils facilitent le débogage des applications en cours de développement.

Note Deux nouvelles variables système permettent de récupérer l'emplacement précis des erreurs de programmation. Pour plus d'informations, reportez-vous au paragraphe "Nouvelles variables système", page 255.

ASSERT

ASSERT(expressionBool {; texteMessage})

Paramètres	Type		Description
expressionBool	Booléen	→	Expression booléenne
texteMessage	Chaîne	→	Texte du message d'erreur

La nouvelle commande ASSERT permet de placer une assertion dans le code d'une méthode.

Une assertion est une instruction insérée dans le code et chargée de détecter des éventuelles anomalies au cours de son exécution. Le principe consiste à vérifier qu'un expression est vraie à un instant donné et, dans le cas contraire, produire une exception.
Les assertions sont surtout utilisées pour détecter des cas qui ne devraient jamais arriver en temps normal. Elles servent donc essentiellement à détecter des bogues de programmation. Il est possible d'activer ou de désactiver globalement toutes les assertions d'une application (par exemple en fonction du type de version) via la nouvelle commande FIXER ACTIVATION ASSERTIONS.
Pour plus d'informations sur les assertions en programmation, reportez-vous à l'article (en anglais) qui leur est consacré sur Wikipedia : http://en.wikipedia.org/wiki/Assertion_(computing)

La commande ASSERT évalue l'expression booléenne passée en paramètre. Si l'expression est vraie, il ne se passe rien. Si l'expression est fausse, la commande déclenche l'erreur -10518 et affiche le texte de l'assertion précédée du message "Fausse assertion :". Vous pouvez intercepter cette erreur via une méthode installée par la commande APPELER SUR ERREUR, afin par exemple d'alimenter un fichier d'historique.

La commande accepte en deuxième paramètre optionnel un texte qui sera affiché dans le message d'erreur à la place du texte de l'expression booléenne si elle est fausse.

- Avant d'effectuer des opérations sur un enregistrement, le développeur souhaite s'assurer qu'il est bien chargé en lecture écriture :

```
LECTURE ECRITURE([Table 1])
CHARGER ENREGISTREMENT([Table 1])
ASSERT(Non(Enregistrement verrouille([Table 1])))
  // déclenche l'erreur -10518 si l'enregistrement est verrouillé
```

- Une assertion peut permettre de tester les paramètres passés à une méthode projet pour détecter des valeurs aberrantes. Dans cet exemple, un message d'alerte personnalisé est utilisé.

```
  // Méthode qui retourne le numéro d'un client en fonction de son nom passé dans $1

C_TEXTE($1) // Nom du client
ASSERT($1 # "";"Recherche d'un nom de client vide")
  // Un nom vide dans ce cas est une valeur aberrante
  // Si assertion fausse, affichera dans la boîte de dialogue d'erreur :
  // "Fausse assertion : Recherche d'un nom de client vide"
```

Référence : Asserted, FIXER ACTIVATION ASSERTIONS

Asserted

Asserted (expressionBool {; texteMessage})→ Booléen

Paramètres	Type		Description
expressionBool	Booléen	→	Expression booléenne
texteMessage	Chaîne	→	Texte du message d'erreur
Résultat	Booléen	←	Résultat de l'évaluation d'expressionBool

La nouvelle commande Asserted a un fonctionnement semblable à celui de la commande ASSERT, à la différence près qu'elle retourne une valeur issue de l'évaluation du paramètre *expressionBool*.

Elle permet donc d'utiliser une assertion lors de l'évaluation d'une condition (cf. exemple). Pour plus d'informations sur le fonctionnement des assertions et sur les paramètres de cette commande, reportez-vous à la description de la commande ASSERT.

Asserted accepte une expression booléenne en paramètre et retourne le résultat de l'évaluation de cette expression. Si l'expression est fausse et si les assertions sont activées (cf. commande FIXER ACTIVATION ASSERTIONS), l'erreur -10518 est générée, exactement comme pour la commande ASSERT. Si les assertions sont inactivées, Asserted retourne simplement le résultat de l'expression qui lui est passée sans déclencher d'erreur.

- Insertion d'une assertion dans l'évaluation d'une expression :

```
LECTURE ECRITURE([Table 1])
CHARGER ENREGISTREMENT([Table 1])
Si (Asserted(Non(Enregistrement verrouille( [Table 1]))))
  // Ce code déclenche l'erreur -10518 si l'enregistrement est verrouillé
  ...
Fin de si
```

Référence : ASSERT, FIXER ACTIVATION ASSERTIONS

FIXER ACTIVATION ASSERTIONS

FIXER ACTIVATION ASSERTIONS(asserts{; *})

Paramètres	Type		Description
asserts	Booléen	→	Vrai = activer les assertions Faux = désactiver les assertions
*	*	→	Si omis = la commande s'applique à l'ensemble des process Si passé = la commande s'applique au process courant uniquement

La nouvelle commande FIXER ACTIVATION ASSERTIONS permet de désactiver ou de réactiver les assertions éventuellement insérées dans le code 4D de l'application. Pour plus d'informations sur les assertions, reportez-vous à la description de la commande ASSERT.

Par défaut, les assertions ajoutées dans le programme sont actives. Cette commande est utile pour les désactiver car leur évaluation peut parfois être coûteuse en temps d'exécution et vous pouvez aussi souhaiter les masquer pour l'utilisateur final de l'application.

Typiquement, la commande FIXER ACTIVATION ASSERTIONS pourra être utilisée dans la méthode base Sur démarrage afin d'activer ou non les assertions suivant que l'application est en mode "Test" ou en mode "Production".

Par défaut, la commande FIXER ACTIVATION ASSERTIONS agit sur tous les process de l'application déjà créés ou créés par la suite. Pour restreindre l'effet de la commande au process courant uniquement, passez le paramètre *.

A noter que lorsque les assertions sont désactivées, les expressions passées aux commandes ASSERT ne sont plus évaluées. Les lignes de code appelant ASSERT n'ont alors plus aucun effet sur le fonctionnement de l'application, ni en termes de comportement ni en terme de performances.

- Désactivation des assertions :

```
FIXER ACTIVATION ASSERTIONS(Faux)
ASSERT(MéthodeTest) // MéthodeTest ne sera pas appelée car les asserts sont désactivés
```

Référence : Lire activation assertions

Lire activation assertions

Lire activation assertions → Booléen

Paramètres	Type		Description
Résultat	Booléen	←	Vrai = les assertions sont activées Faux = les assertions sont inactivées

La nouvelle commande Lire activation assertions retourne Vrai ou Faux suivant que les assertions sont actives ou non dans la base. Pour plus d'informations sur les assertions, reportez-vous à la description de la commande ASSERT.

Par défaut, les assertions sont actives mais elles peuvent avoir été désactivées à l'aide de la commande FIXER ACTIVATION ASSERTIONS.

Référence : FIXER ACTIVATION ASSERTIONS

Ignorer les erreurs répétées dans la fenêtre d'erreur

La fenêtre d'erreur de 4D propose un nouveau raccourci, permettant d'ignorer une erreur se produisant de façon répétitive, par exemple dans une boucle.

Pour cela, il suffit de maintenir appuyée la touche **Alt** (Windows) ou **Option** (Mac OS) lorsque vous cliquez sur le bouton **Continuer** dans la fenêtre d'erreur de 4D. Cette action permet de ne plus afficher la fenêtre si la même erreur, déclenchée par la même méthode à la même ligne, survient à nouveau. Tout se passe dans ce cas comme si l'utilisateur cliquait à chaque fois sur le bouton **Continuer**.

Langage

EXECUTER METHODE DANS SOUS FORMULAIRE

EXECUTER METHODE DANS SOUS FORMULAIRE(objetSousForm; nomMéthode{; retour{; param;...;paramN}})

Paramètres	Type		Description
objetSousForm	Chaîne	→	Nom de l'objet sous-formulaire
nomMéthode	Chaîne	→	Nom de la méthode projet à exécuter
retour	Variable | *	←	Valeur retournée par la méthode ou * si la méthode ne retourne pas de valeur
param	Variable	→	Paramètre(s) à passer à la méthode

La nouvelle commande EXECUTER METHODE DANS SOUS FORMULAIRE permet d'exécuter la méthode projet *nomMéthode* dans le contexte de l'objet de sous-formulaire *objetSousForm*.

La méthode projet appelée peut recevoir de 1 à N paramètres dans *param* et retourner une valeur dans *retour*. Passez * dans le paramètre *retour* si la méthode ne retourne pas de paramètres.

Vous pouvez passer le nom de toute méthode projet accessible depuis la base ou le composant exécutant la commande. Le contexte d'exécution est préservé dans la méthode appelée, ce qui signifie que le formulaire courant et l'événement formulaire courant restent définis. Si le sous-formulaire provient d'un composant, la méthode doit appartenir au composant et disposer de la propriété "Partagée entre les composants et la base hôte".

Cette commande doit être appelée depuis la méthode formulaire du formulaire parent (contenant l'objet *objetSousForm*).

Pour plus d'informations sur les mécanismes des sous-formulaires dans 4D v12, reportez-vous au paragraphe "Programmation inter-formulaires avancée", page 78.

- Soit le formulaire "ContactDétail" utilisé comme sous-formulaire dans le formulaire parent "Société". L'objet sous-formulaire qui contient le formulaire ContactDétail est nommé "ContactSousForm". Imaginons que nous souhaitions modifier l'apparence de certains éléments du sous-formulaire en fonction de la valeur de champ(s) de la société (par exemple, "nomcontact" doit passer en rouge lorsque [Société]Ville="New York" et en bleu lorsque [Société]Ville="San Diego"). Ce mécanisme est mis en oeuvre via la méthode *SetToColor*. Pour pouvoir obtenir ce résultat, la méthode *SetToColor* ne peut pas être appelée directement depuis le process de l'événement formulaire "Sur chargement" du formulaire parent Société car l'objet "nomcontact" n'appartient pas au formulaire courant, mais au formulaire affiché dans l'objet sous-formulaire "ContactSousForm". La méthode doit donc être exécutée à l'aide de EXECUTER METHODE DANS SOUS FORMULAIRE pour pouvoir fonctionner correctement.

```
Au cas ou
  : (Evenement formulaire= Sur chargement)
     Au cas ou
        :([Société]Ville = "New York")
            $Color:=$Red
        :([Société]Ville = "San Diego")
          $Color:=$Blue
       Sinon
          $Color:=$Black
     Fin de cas
     EXECUTER METHODE DANS SOUS FORMULAIRE("ContactSous-
Form";"SetToColor";*;$Color)
Fin de cas
```

- Vous développez une base de données qui sera utilisée comme composant. Elle comporte un formulaire projet partagé (nommé par exemple Calendrier) contenant des Variables dynamiques ainsi qu'une méthode projet publique permettant de régler le calendrier : *SetCalendarDate(varDate)*.
  Si cette méthode était utilisée directement dans la méthode du formulaire Calendrier, vous pourriez l'appeler directement dans l'événement "Sur chargement" : *SetCalendarDate(Date du jour)*. Mais, dans le contexte de la base hôte, imaginons qu'un formulaire projet contienne deux sous-formulaires "Calendrier", dans des objets sous-formulaire appelés "Cal1" et "Cal2". Vous devez régler la date de Cal1 au 01/01/10 et celle de Cal2 au 05/05/10. Vous ne pouvez pas appeler directement *SetCalendarDate* car la méthode ne "saura" pas à quels formulaire et variables elle devra s'appliquer. Vous devez donc l'appeler via le code suivant :

```
EXECUTER METHODE DANS SOUS FORMULAIRE("Cal1";
            "SetCalendarDate";*;!01/01/10!)
EXECUTER METHODE DANS SOUS FORMULAIRE("Cal2";
            "SetCalendarDate";*;!05/05/10!)
```

- Exemple avancé : dans le même contexte que précédemment, cet exemple propose une méthode générique :

```
  // Contenu de la méthode SetCalendarDate
C_DATE($1)
C_TEXTE($2)
Au cas ou
  : (Nombre de parametres=1)
    // Exécution standard de la méthode (comme si elle était exécutée
    // depuis le formulaire lui-même) ou spécifiquement pour un
    // contexte (voir cas 2)

  : (Nombre de parametres=2)
    // Appel depuis l'extérieur, a besoin d'un contexte
    // Appel récursif avec un seul paramètre
EXECUTER METHODE DANS SOUS FORMULAIRE($2;
            "SetCalendarDate";*;$1)
Fin de cas
```

List Box

Plusieurs modifications ont été effectuées dans le thème List Box de 4D v12 :

- quatre nouvelles commandes sont dédiées à la gestion des list box hiérarchiques (pour plus d'informations sur les list box hiérarchiques, reportez-vous au paragraphe "List box", page 106),
- la nouvelle commande LISTBOX LIRE INFORMATIONS IMPRESSION permet de contrôler l'impression des list box,
- de nouveaux paramètres pour les commandes LISTBOX FIXER LARGEUR COLONNE et LISTBOX Lire largeur colonne permettent de contrôler le redimensionnement des colonnes.
- pour faciliter leur utilisation notamment via la saisie prédictive, toutes les commandes ont été préfixées LISTBOX.

LISTBOX FIXER HIERARCHIE

LISTBOX FIXER HIERARCHIE ({*; }objet; hiérarchique{; hiérarchie})

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)
hiérarchique	Booléen	→	Vrai = list box hiérarchique Faux = list box non hiérarchique
hiérarchie	Tableau	→	Tableau de pointeurs

La nouvelle commande LISTBOX FIXER HIERARCHIE vous permet de configurer l'objet list box désigné par les paramètres *objet* et * en mode hiérarchique ou non.

Note 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.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre *objet* est un nom d'objet (une chaîne). Si vous ne passez pas ce paramètre, vous indiquez que le paramètre *objet* est une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de variable.

Le paramètre booléen *hiérarchique* vous permet de définir le mode de la list box :

- si vous passez Vrai, la list box est affichée en mode hiérarchique,
- si vous passez Faux, la list box est affichée en mode non hiérarchique (mode tableau standard).

Lorsque vous passez une list box en mode hiérarchique, certaines propriétés sont automatiquement restreintes. Pour plus d'informations, reportez-vous au paragraphe "List box hiérarchiques", page 106.

Le paramètre *hiérarchie* vous permet de désigner les tableaux de la list box à utiliser pour construire la hiérarchie.
Si vous omettez ce paramètre :

- si la list box est déjà en mode hiérarchique, la commande ne fait rien.
- si la list box est en mode non hiérarchique et n'a jamais été déclarée hiérarchique, le premier tableau est utilisé comme hiérarchie par défaut.
- si la list box est en mode non hiérarchique mais avait été déclarée hiérarchique précédemment, la dernière hiérarchie est rétablie.

▸ Définition des tableaux *tPays*, *tRegion* et *tVille* comme hiérarchie d'une list box :

```
TABLEAU POINTEUR($TabHierarch;3)
$TabHierarch{1}:=->tPays `Premier niveau de rupture
$TabHierarch{2}:=->tRegion `Deuxième niveau de rupture
$TabHierarch{3}:=->tVille `Troisième niveau de rupture
LISTBOX FIXER HIERARCHIE(*;"mylistbox";Vrai;$TabHierarch)
```

Référence : LISTBOX LIRE HIERARCHIE

LISTBOX LIRE HIERARCHIE

LISTBOX LIRE HIERARCHIE ({*; }objet; hiérarchique{; hiérarchie})

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)
hiérarchique	Booléen	←	Vrai = list box hiérarchique Faux = list box non hiérarchique
hiérarchie	Tableau	←	Tableau de pointeurs

La nouvelle commande LISTBOX LIRE HIERARCHIE vous permet de connaître les propriétés hiérarchiques de l'objet list box désigné par les paramètres *objet* et *.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre *objet* est un nom d'objet (une chaîne). Si vous ne passez pas ce paramètre, vous indiquez que le paramètre *objet* est une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de variable.

Le paramètre booléen *hiérarchique* vous indique si list box est en mode hiérarchique ou non :

- si le paramètre retourne Vrai, la list box est en mode hiérarchique,
- si le paramètre retourne Faux, la list box est affichée en mode non hiérarchique (mode tableau standard).

Si la list box est en mode hiérarchique, la commande remplit le tableau *hiérarchie* avec des pointeurs vers les tableaux de la list box utilisés pour définir la hiérarchie.

Note Si la list box est en mode non hiérarchique, la commande retourne dans le premier élément du tableau hiérarchie un pointeur vers le tableau de la première colonne de la list box.

Référence : LISTBOX FIXER HIERARCHIE

LISTBOX DEPLOYER

LISTBOX DEPLOYER ({*; }objet{;récursive{; sélecteur{; ligne; colonne} | {niveau}}}})

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)
récursive	Booléen	→	Vrai = déployer les sous-niveaux Faux = ne pas déployer les sous-niveaux
sélecteur	Entier long	→	Partie de la list box à déployer
ligne	Entier long	→	Numéro de ligne de la rupture à déployer
colonne	Entier long	→	Numéro de colonne de la rupture à déployer
niveau	Entier long	→	Numéro de niveau de la list box à déployer

La commande LISTBOX DEPLOYER vous permet de provoquer le déploiement des lignes de rupture de l'objet list box désigné par les paramètres *objet* et *.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre *objet* est un nom d'objet (une chaîne). Si vous ne passez pas ce paramètre, vous indiquez que le paramètre *objet* est une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de variable.

Si la list box n'est pas configurée en mode hiérarchique, la commande ne fait rien. Pour plus d'informations sur les list box hérarchiques, reportez-vous au paragraphe "List box", page 106.

Le paramètre optionnel *récursive* vous permet de paramétrer le déploiement des sous-niveaux hiérarchiques de la list box. Passez Vrai ou omettez ce paramètre pour que la commande provoque le déploiement de tous les niveaux et tous les sous-niveaux. Si vous passez Faux, seul le premier niveau désigné sera déployé.

Le paramètre optionnel *sélecteur* vous permet de définir la portée de la commande. Vous pouvez passer dans ce paramètre l'une des constantes suivantes :

- Listbox tout (valeur par défaut, utilisée si le paramètre est omis) : déploie tous les sous-niveaux.
- Listbox sélection : déploie les sous-niveaux sélectionnés.

- Listbox ligne rupture : déploie le sous-niveau auquel appartient la "cellule" désignée par les paramètres *ligne* et *colonne*. A noter que ces paramètres représentent les numéros de ligne et de colonne dans la listbox en mode standard et non dans sa représentation hiérarchique. Dans ce cas, si les paramètres *ligne* et *colonne* sont omis, la commande ne fait rien.
- Listbox niveau : déploie toutes les lignes de rupture correspondant à la colonne *niveau*. Ce paramètre désigne un numéro de colonne dans la listbox en mode standard et non dans sa représentation hiérarchique. Dans ce cas, si le paramètre *niveau* est omis, la commande ne fait rien.

La commande ne sélectionne pas les lignes de rupture.

Si la sélection ou la list box ne contient pas de ligne de rupture, ou si toutes les lignes de rupture sont déjà déployées, la commande ne fait rien.

▸ Cet exemple illustre différents modes d'utilisation de la commande. Soient les tableaux suivants représentés dans une listbox :

France	Bretagne	Brest	120000
France	Bretagne	Quimper	80000
France	Bretagne	Rennes	200000
France	Normandie	Caen	220000
France	Normandie	Deauville	4000
France	Normandie	Cherbourg	41000
Belgique	Wallonie	Namur	111000
Belgique	Wallonie	Liège	200000
Belgique	Flandre	Anvers	472000
Belgique	Flandre	Louvain	95000

`Déployer toutes les lignes et sous-lignes de rupture de la list box
LISTBOX DEPLOYER(*;"MaListbox")

V France		
V Bretagne		
	Brest	120000
	Quimper	80000
	Rennes	200000
V Normandie		
	Caen	220000
	Deauville	4000
	Cherbourg	41000

V Belgique	
V Wallonie	
Namur	111000
Liège	200000
V Flandre	
Anvers	472000
Louvain	95000

`Déployer le premier niveau de lignes de rupture de la sélection
LISTBOX DEPLOYER (*;"MaListbox";**Faux**;Listbox sélection)
`Si la ligne "Belgique" était sélectionnée

> France
V Belgique
> Wallonie
> Flandre

`Déployer la ligne de rupture Bretagne sans récursivité
LISTBOX DEPLOYER (*;"MaListbox";**Faux**;Listbox ligne rupture;1;2)

V France	
V Bretagne	
Brest	120000
Quimper	80000
Rennes	200000
> Normandie	
> Belgique	

`Déployer toutes les premières colonnes (pays) sans récursivité
LISTBOX DEPLOYER (*;"MaListbox";**Faux**;Listbox niveau;1)

V France
> Bretagne
> Normandie
V Belgique
> Wallonie
> Flandre

Référence : LISTBOX CONTRACTER

LISTBOX CONTRACTER

LISTBOX CONTRACTER ({*; }objet{;récursive{; sélecteur{; ligne; colonne} | {niveau}}}})

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)
récursive	Booléen	→	Vrai = contracter les sous-niveaux Faux = ne pas contracter les sous-niveaux
sélecteur	Entier long	→	Partie de la list box à contracter
ligne	Entier long	→	Numéro de ligne de la rupture à contracter
colonne	Entier long	→	Numéro de colonne de la rupture à contracter
niveau	Entier long	→	Numéro de niveau de la list box à contracter

La commande LISTBOX CONTRACTER vous permet de provoquer la contraction des lignes de rupture de l'objet list box désigné par les paramètres *objet* et *.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre *objet* est un nom d'objet (une chaîne). Si vous ne passez pas ce paramètre, vous indiquez que le paramètre *objet* est une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de variable.

Si la list box n'est pas configurée en mode hiérarchique, la commande ne fait rien. Pour plus d'informations sur les list box hérarchiques, reportez-vous au paragraphe "List box", page 106.

Le paramètre optionnel *récursive* vous permet de paramétrer la contraction des sous-niveaux hiérarchiques de la list box. Passez Vrai ou omettez ce paramètre pour que la commande provoque la contraction de tous les niveaux et tous les sous-niveaux. Si vous passez Faux, seul le premier niveau sera contracté.

Le paramètre optionnel *sélecteur* vous permet de définir la portée de la commande. Vous pouvez passer dans ce paramètre l'une des constantes suivantes :

- Listbox tout (valeur par défaut, utilisée si le paramètre est omis) : contracte tous les sous-niveaux.
- Listbox sélection : contracte les sous-niveaux sélectionnés.

- Listbox ligne rupture : contracte le sous-niveau auquel appartient la "cellule" désignée par les paramètres *ligne* et *colonne*. A noter que ces paramètres représentent les numéros de ligne et de colonne dans la listbox en mode standard et non dans sa représentation hiérarchique.
- Listbox niveau : contracte toutes les lignes de rupture correspondant à la colonne *niveau*. Ce paramètre désigne un numéro de colonne dans la listbox en mode standard et non dans sa représentation hiérarchique.

Si la sélection ou la list box ne contient pas de ligne de rupture, ou si toutes les lignes de rupture sont déjà contractées, la commande ne fait rien.

▸ Cet exemple contracte le premier niveau de lignes de rupture de la sélection de la list box :

LISTBOX CONTRACTER (*;"MaListbox";**Faux**;Listbox sélection)

Référence : LISTBOX DEPLOYER

LISTBOX SELECTIONNER RUPTURE

LISTBOX SELECTIONNER RUPTURE({*; }objet; ligne; colonne{; action})

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)
ligne	Entier long	→	Numéro de ligne de la rupture
colonne	Entier long	→	Numéro de colonne de la rupture
action	Entier long	→	Action de sélection

La nouvelle commande LISTBOX SELECTIONNER RUPTURE permet de sélectionner des lignes de rupture dans l'objet list box désigné par les paramètres *objet* et *. La list box doit être affichée en mode hiérarchique.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre *objet* est un nom d'objet (une chaîne). Si vous ne passez pas ce paramètre, vous indiquez que le paramètre *objet* est une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de variable.

Les lignes de rupture sont ajoutées pour représenter la hiérarchie mais ne correspondent pas à des lignes de tableaux existantes. Pour désigner une ligne de rupture à sélectionner, vous devez passer dans les paramètres *ligne* et *colonne* des numéros de ligne et de colonne correspondant à la première occurrence dans le tableau correspondant. Ces valeurs sont retournées par la commande LISTBOX LIRE POSITION CELLULE lorsque l'utilisateur a sélectionné une ligne de rupture. Ce principe est détaillé dans le paragraphe "Gestion des sélections et des positions", page 113.

Le paramètre *action*, s'il est passé, permet de définir l'action de sélection à effectuer lorsqu'une sélection de lignes de rupture existe déjà dans la list box. Vous pouvez passer une valeur ou l'une des constantes suivantes, placées dans le thème "List box" :

- Remplacer sélection listbox (0) : La ligne de rupture sélectionnée devient la nouvelle sélection de lignes de rupture et remplace la sélection existante. La commande produit le même effet qu'un clic de l'utilisateur sur une ligne de rupture. Cette action est effectuée par défaut (lorsque le paramètre *action* n'est pas passé).
- Ajouter à sélection listbox (1) : la ligne de rupture sélectionnée est ajoutée à la sélection existante. Si la ligne de rupture désignée appartient déjà à la sélection existante, la commande ne fait rien.
- Supprimer de sélection listbox (2) : la ligne de rupture sélectionnée est supprimée de la sélection existante. Si la ligne de rupture désignée n'appartient pas à la sélection existante, la commande ne fait rien.

▶ Soient les tableaux suivants représentés dans une list box :

(T1)	(T2)	(T3)	(T4)
France	Bretagne	Brest	120000
France	Bretagne	Quimper	80000
France	Bretagne	Rennes	200000
France	Normandie	Caen	220000
France	Normandie	Deauville	4000
France	Normandie	Cherbourg	41000
Belgique	Wallonie	Namur	111000
Belgique	Wallonie	Liège	200000
Belgique	Flandre	Anvers	472000
Belgique	Flandre	Louvain	95000

Nous souhaitons sélectionner la ligne de rupture "Normandie" :

```
$ligne:=Chercher dans tableau (T2;"Normandie")
$colonne:=2
LISTBOX CONTRACTER (*;"MaListbox") `contraction de tous les niveaux
LISTBOX SELECTIONNER RUPTURE (*;"MaListbox";$ligne;$colonne)
```

Voici le résultat :

V France
> Bretagne
> Normandie
> Belgique

Référence : LISTBOX LIRE POSITION CELLULE, LISTBOX DEPLOYER

LISTBOX LIRE INFORMATIONS IMPRESSION

LISTBOX LIRE INFORMATIONS IMPRESSION ({*; }objet; sélecteur; info)

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)
sélecteur	Entier long	→	Information à obtenir
info	Entier long | Booléen	←	Valeur courante

La nouvelle commande LISTBOX LIRE INFORMATIONS IMPRESSION retourne des informations courantes relatives à l'impression de l'objet list box désigné par les paramètres *objet* et *. Cette commande permet de contrôler l'impression du contenu de la list box.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre *objet* est un nom d'objet (une chaîne). Si vous ne passez pas ce paramètre, vous indiquez que le paramètre *objet* est une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de variable.

Cette commande doit être appelée dans le contexte de l'impression d'une list box via la commande Imprimer objet. Hors de ce contexte, elle ne retourne pas de valeurs significatives.

Passez dans *sélecteur* une valeur indiquant l'information à connaître et dans *info* une variable de type numérique ou BLOB. Vous pouvez passer dans *sélecteur* une des constantes suivantes, placées dans le thème "List box" :

sélecteur	Valeur retournée dans info
Listbox Num dernière ligne impr (0)	Retourne dans *info* le numéro de la dernière ligne imprimée. Permet de connaître le numéro de la prochaine ligne à imprimer. Le numéro retourné peut être supérieur au nombre de lignes effectivement imprimées si la list box contient des lignes invisibles ou si la commande OBJET FIXER DEFILEMENT a été appelée. Par exemple, si les lignes 1, 18 et 20 ont été imprimées, *info* vaut 20.
Listbox Nombre lignes imprimées (1)	Retourne dans *info* le nombre de lignes effectivement imprimées lors du dernier appel à la commande Imprimer objet. Ce nombre inclut les éventuelles lignes de ruptures ajoutées dans le cadre d'une list box hiérarchique. Par exemple, *info* vaut 10 si la list box contient 20 lignes et que les lignes impaires ont été masquées.
Listbox Impression terminée (2)	Retourne dans *info* un booléen indiquant si la dernière ligne (visible) de la list box a été effectivement imprimée. Vrai = la ligne a été imprimée, Faux sinon.
Listbox Hauteur imprimée (3)	Retourne dans *info* la hauteur en pixels de l'objet effectivement imprimé (inclut les entêtes, filets, etc.). Souvenez-vous que si le nombre de lignes à imprimer est inférieur à la "capacité" de la list box, sa hauteur est automatiquement réduite.

Pour plus d'informations sur les principes d'impression des list box, reportez-vous au paragraphe "Impression des list box", page 119.

- Impression jusqu'à ce que toutes les lignes soient imprimées :

```
OUVRIR TACHE IMPRESSION
OUVRIR FORMULAIRE IMPRESSION("SalesForm")
…
$Over:=Faux
```

```
Repeter
  IMPRIMER OBJET (*;"malistbox")
  LISTBOX LIRE INFORMATIONS IMPRESSION(*;"malistbox";
                                       Listbox Impression terminée;$Over)
Jusque ($Over)
...
FERMER TACHE IMPRESSION
```

- Impression d'au moins 500 lignes de la list box, sachant que certaines lignes sont masquées :

```
$GlobalPrinted:=0
Repeter
  IMPRIMER OBJET (*;"malistbox")
  LISTBOX LIRE INFORMATIONS IMPRESSION(*;"malistbox";
                                       Listbox Nombre lignes imprimées;$Printed)
  $GlobalPrinted:=$GlobalPrinted+$Printed
  SAUT DE PAGE
Jusque ($GlobalPrinted>=500)
```

LISTBOX FIXER LARGEUR COLONNE

LISTBOX FIXER LARGEUR COLONNE({*; }objet; largeur*{; largeurMin{; largeurMax}}*)

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)
largeur	Entier long	→	Largeur de colonne (en pixels)
largeurMin	*Entier long*	→	*Largeur minimale de colonne (en pixels)*
largeurMax	*Entier long*	→	*Largeur maximale de colonne (en pixels)*

La commande LISTBOX FIXER LARGEUR COLONNE admet désormais deux paramètres optionnels supplémentaires, permettant de fixer des limites au redimensionnement manuel de la colonne.

Vous pouvez passer dans *largeurMin* et *largeurMax* respectivement des valeurs de largeur minimale et maximale, exprimées en pixels.

Si vous souhaitez que l'utilisateur ne puisse pas redimensionner la colonne, il suffit de passer la même valeur dans *largeur*, *largeurMin* et *largeurMax*.

Référence : LISTBOX Lire largeur colonne

LISTBOX Lire largeur colonne

LISTBOX Lire largeur colonne({*; }objet*{; largeurMin{; largeurMax}}*) → Entier long

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)
largeurMin	*Entier long*	←	*Largeur minimale de colonne (en pixels)*
largeurMax	*Entier long*	←	*Largeur maximale de colonne (en pixels)*
Résultat	Entier long	←	Largeur de colonne (en pixels)

La commande LISTBOX Lire largeur colonne peut désormais retourner dans les nouveaux paramètres *largeurMin* et *largeurMax* les limites de redimensionnement de la colonne.

Ces limites peuvent être définies via la commande LISTBOX FIXER LARGEUR COLONNE.

Si aucune valeur de largeur minimale et/ou maximale n'a été fixée pour la colonne, le paramètre correspondant retourne 0.

Référence : LISTBOX FIXER LARGEUR COLONNE

Commandes d'insertion ou suppression

Les commandes existantes LISTBOX INSERER COLONNE, LISTBOX INSERER COLONNE FORMULE et LISTBOX SUPPRIMER COLONNE sont sans effet lorsqu'elles sont appliquées à la première colonne d'une list box affichée en mode hiérarchique.

Note Ces commandes sont renommées dans 4D v12 (cf. paragraphe ci-dessous).

Commandes renommées

Pour des raisons d'harmonisation et de standardisation, les commandes du thème "Listbox" existantes ont été renommées dans 4D v12. Le préfixe "LISTBOX" a été systématiquement ajouté. Le fonctionnement de ces commandes est inchangé.

Nouveau nom 4D v12	Ancien nom
LISTBOX FIXER COULEUR GRILLE	*FIXER COULEUR GRILLE LISTBOX*
LISTBOX FIXER HAUTEUR LIGNES	*FIXER HAUTEUR LIGNES LISTBOX*
LISTBOX FIXER LARGEUR COLONNE	*FIXER LARGEUR COLONNE LISTBOX*
LISTBOX FIXER TABLE SOURCE	*FIXER TABLE SOURCE LISTBOX*
LISTBOX INSERER COLONNE FORMULE	*INSERER COLONNE FORMULE LISTBOX*
LISTBOX INSERER COLONNE	*INSERER COLONNE LISTBOX*
LISTBOX INSERER LIGNE	*INSERER LIGNE LISTBOX*
LISTBOX Lire hauteur lignes	*Lire hauteur lignes listbox*
LISTBOX Lire information	*Lire information listbox*
LISTBOX Lire largeur colonne	*Lire largeur colonne listbox*
LISTBOX Lire nombre colonnes	*Lire nombre colonnes listbox*
LISTBOX Lire nombre lignes	*Lire nombre lignes listbox*
LISTBOX LIRE POSITION CELLULE	*LIRE POSITION CELLULE LISTBOX*
LISTBOX LIRE TABLE SOURCE	*LIRE TABLE SOURCE LISTBOX*
LISTBOX LIRE TABLEAUX	*LIRE TABLEAUX LISTBOX*
LISTBOX MONTRER GRILLE	*MONTRER GRILLE LISTBOX*
LISTBOX NUMERO COLONNE DEPLACEE	*NUMERO COLONNE LISTBOX DEPLACEE*
LISTBOX NUMERO LIGNE DEPLACEE	*NUMERO LIGNE LISTBOX DEPLACEE*
LISTBOX SELECTIONNER LIGNE	*SELECTIONNER LIGNE LISTBOX*
LISTBOX SUPPRIMER COLONNE	*SUPPRIMER COLONNE LISTBOX*
LISTBOX SUPPRIMER LIGNE	*SUPPRIMER LIGNE LISTBOX*
LISTBOX TRIER COLONNES	*TRIER COLONNES LISTBOX*

Outils

Plusieurs modifications ont été effectuées dans le thème "Outils" de 4D v12 :

- une nouvelle commande permettant de générer des identifiants uniques UUID a été ajoutée,
- les commandes ENCODER et DECODER ont été renommées ENCODER BASE64 et DECODER BASE64 et peuvent retourner un texte.

Generer UUID

Generer UUID → Alpha

Paramètres	Type		Description
Cette commande ne requiert pas de paramètres			
Résultat	Alpha	←	Nouvel UUID sous forme de texte non-canonique (32 caractères)

La commande Generer UUID retourne un nouvel identifiant UUID de 32 caractères sous forme non-canonique.

Pour plus d'informations sur les identifiants UUID, reportez-vous au paragraphe "Prise en charge des UUID", page 19.

- Génération d'un UUID dans une variable :

```
C_TEXTE(MonUUID)
MonUUID:=Generer UUID
```

FIXER VARIABLE ENVIRONNEMENT

La commande FIXER VARIABLE ENVIRONNEMENT fonctionne désormais avec la commande PHP Executer.

ENCODER BASE64

ENCODER BASE64 (blob*{; texteEncodé}*)

Paramètres	Type		Description
blob	BLOB	→	BLOB à encoder en base64
		←	BLOB encodé en base64 *(si* texteEncodé *omis)*
texteEncodé	*Texte*	←	*Résultat de blob encodé en base64*

Note de compatibilité La commande ENCODER BASE64 était nommée ENCODER dans les versions précédentes de 4D.

La commande ENCODER BASE64 accepte un nouveau paramètre optionnel, *texteEncodé*. Si vous passez ce paramètre, il reçoit à l'issue de l'exécution de la commande le contenu de *blob* encodé. Dans ce cas, le paramètre *blob* lui-même n'est pas modifié par la commande.

Cette modification permet d'utiliser la commande de façon plus conforme aux spécifications du XML (cf. paragraphe "Note sur l'usage de BLOBs XML dans 4D v12", page 237).

Si vous omettez le paramètre *texteEncodé*, la commande fonctionne comme dans les versions précédentes de 4D.

DECODER BASE64

DECODER BASE64 (*{texteEncodé; }* blob)

Paramètres	Type		Description
texteEncodé	*Texte*	→	*Texte contenant un BLOB encodé en base64*
blob	BLOB	→	BLOB encodé en base64 *(si* texteEncodé *omis)*
		←	BLOB décodé

Note de compatibilité La commande DECODER BASE64 était nommée DECODER dans les versions précédentes de 4D.

La commande DECODER BASE64 accepte un nouveau paramètre optionnel, *texteEncodé*. Si vous passez ce paramètre, la commande décode son contenu et le retourne dans le paramètre *blob*. Il doit contenir un BLOB encodé en base64. Dans ce cas, le contenu initial du paramètre *blob* est ignoré par la commande.

Cette modification permet d'utiliser la commande de façon plus conforme aux spécifications du XML (cf. paragraphe "Note sur l'usage de BLOBs XML dans 4D v12", page 237).

Si vous omettez le paramètre *texteEncodé*, la commande fonctionne comme dans les versions précédentes de 4D.

▸ Cet exemple permet de transférer une image via un BLOB :

```
C_BLOB($blobSource)
C_IMAGE($monimage)
$monimage:=[personnes]photo
IMAGE VERS BLOB($monimage;$blobSource;".JPG")
C_TEXTE($texteBASE64)
ENCODER BASE64($blobSource;$texteBASE64) //Encodage du texte
// le binaire est maintenant disponible sous forme de chaîne de caractères dans $texteBASE64

C_TEXTE($texteBASE64)
C_BLOB($blobCible)
DECODER BASE64($texteBASE64;$blobCible) //Décodage du texte
// le binaire encodé en base 64 est maintenant disponible sous forme de BLOB dans $blobCible
```

PHP

Ce nouveau thème regroupe les commandes dédiées à l'exécution de scripts PHP dans 4D (cf. paragraphe "Exécuter des scripts PHP dans 4D", page 65).

PHP Executer

PHP Executer (cheminScript{; nomFonction{; * | résultat{; param}{; param2 ;…; paramN}}}) → Booléen

Paramètres	Type		Description
cheminScript	Chaîne	→	Chemin d'accès au script PHP ou "" pour exécuter une fonction PHP
nomFonction	Chaîne	→	Fonction PHP à exécuter
* | résultat	* | Variable | Champ	←	Résultat d'exécution de la fonction PHP ou * pour ne pas recevoir de résultat
param		→	Paramètre(s) de la fonction PHP
Résultat	Booléen	←	Vrai = exécution correcte Faux = erreur d'exécution

La commande PHP Executer permet d'exécuter un script ou une fonction PHP.

Passez dans le paramètre *cheminScript* le chemin d'accès du fichier de script PHP à exécuter. Il peut s'agir d'un chemin d'accès relatif si le fichier est situé à côté de la structure de la base ou d'un chemin complet. Le chemin d'accès peut être exprimé en syntaxe système ou Posix.
Si vous souhaitez exécuter directement une fonction PHP standard, passez une chaîne vide ("") dans *cheminScript*. Le nom de la fonction doit être passé en deuxième paramètre.

Passez dans le paramètre *nomFonction* un nom de fonction PHP si vous souhaitez exécuter une fonction spécifique dans le script *cheminScript*. Si vous passez une chaîne vide ou omettez ce paramètre, le script est exécuté entièrement.

Note PHP tient compte de la casse des caractères dans le nom de la fonction. N'utilisez pas de parenthèses, saisissez uniquement le nom de la fonction.

Le paramètre *résultat* reçoit le résultat de l'exécution de la fonction PHP. Vous pouvez passer soit :

- une variable, un tableau ou un champ afin de recevoir le résultat,
- le caractère * si la fonction ne retourne pas de résultat ou si vous ne souhaitez pas le récupérer.

Vous pouvez passer une variable, un tableau ou un champ de type texte, entier long, réel, booléen, date ainsi que (hormis pour les tableaux) BLOB et heure. 4D effectuera la conversion des données et les ajustements nécessaires (pour une valeur réelle, choix du séparateur décimal en fonction du système par exemple).
Si la valeur retournée ne correspond pas au type du paramètre *résultat*, 4D la convertira dans la mesure du possible. Par exemple, si vous passez une variable *résultat* de type booléen et recevez un nombre, *résultat* vaudra Vrai sauf si le nombre est égal à 0. A l'inverse, si vous passez une variable *résultat* de type numérique (entier long ou réel) et recevez True ou False, *résultat* vaudra 1 ou 0.

Si le résultat est placé dans une variable ou un champ de type texte, par défaut il ne contient pas les en-têtes retournés par l'interpréteur PHP. Si vous souhaitez récupérer également les en-têtes en texte, utilisez la constante PHP Résultat brut avec la commande PHP FIXER OPTION. A noter que si le résultat retourné n'est pas exploitable en texte, 4D l'encode en base64.

Notes - Si le résultat est placé dans un BLOB, les en-têtes sont toujours conservés.
- Si le résultat est placé dans une date, le script PHP doit retourner une date au format DATE_ATOM (type "AAAA-MM-JJTHH:MM:SS") et non un format personnalisé ou localisé.

Si la fonction PHP appelée attend des arguments, utilisez le(s) paramètre(s) *param* pour passer une ou plusieurs valeurs. Les valeurs doivent être séparées par des points-virgules. Vous pouvez passer des valeurs de type alpha, texte, booléen, réel, entier, entier long, date ou heure. Les images et les BLOBs ne sont pas admis. Vous pouvez passer les valeurs d'un tableau en utilisant un pointeur sur ce tableau (cf. exemple). La commande accepte tous les types de tableaux sauf les tableaux pointeur, les tableaux image et les tableaux 2D.

Notes - Lorsque vous passez une date, elle est automatiquement convertie en texte UTC par 4D. Par exemple, le 23 avril 2010 est envoyé sous la forme "2010-04-23T00:00:00Z". De même, les heures sont automatiquement converties en texte au format HH:MM:SS. Par exemple, 1h05 est envoyée sous la forme "01:05:00".
Vous pouvez utiliser une chaîne personnalisée si vous souhaitez changer de format. En particulier, comme il n'existe pas de type "datetime" dans le langage 4D, vous pouvez construire une chaîne représentant un format ISO ("2008-12-10T19:12:28+02:00").
- Pour des raisons techniques, la taille des paramètres passés via le protocole *fast cgi* ne doit pas dépasser 64 Ko. Vous devez tenir compte de cette limitation si vous utilisez des paramètres de type Texte.

La commande retourne Vrai si l'exécution s'est déroulée correctement côté 4D, c'est-à-dire si le lancement de l'environnement d'exécution, l'ouverture du script et l'établissement de la communication avec l'interpréteur PHP ont été réussis. Dans le cas contraire, une erreur est générée, que vous pouvez intercepter avec la commande APPELER SUR ERREUR et analyser avec LIRE PILE DERNIERE ERREUR.
En outre, le script lui-même peut générer des erreurs PHP. Dans ce cas, vous devez utiliser la commande PHP LIRE REPONSE COMPLETE afin d'analyser la source de l'erreur (voir exemple 4).

Note PHP permet de configurer la gestion d'erreurs. Pour plus d'informations, reportez-vous par exemple à la page http://www.php.net/manual/en/errorfunc.configuration.php#ini.error-reporting.

Utiliser les variables d'environnement

Vous pouvez utiliser la commande FIXER VARIABLE ENVIRONNEMENT pour définir des variables d'environment utilisées par le script. Attention : après un appel à LANCER PROCESS EXTERNE ou PHP Executer, l'ensemble des variables d'environment est effacé.

Fonctions spéciales

4D propose les fonctions spéciales suivantes :

- quit_4d_php : permet de quitter l'interpréteur PHP et tous ses process enfants. Si un process enfant au moins est en train d'exécuter un script, l'interpréteur ne quitte pas et la commande PHP Executer retourne Faux.
- relaunch_4d_php permet de relancer l'interpréteur PHP.
A noter que l'interpréteur est relancé automatiquement à la première requête envoyée par PHP Executer.

Exemples

- Exemple 1 : Appel du script "myPhpFile.php" sans fonction. Voici le contenu du script :

```
<?php
echo 'Version php courante : ' . phpversion();
?>
```

Le code 4D suivant :

```
C_TEXTE($result)
C_BOOLEEN($isOK)
$isOK:=PHP Executer("C:\\php\\myPhpFile.php"; "";$result)
ALERTE ($Result)
```

... affichera "Version php courante : 5.3"

- Exemple 2 : Appel de la fonction *myPhpFunction* dans le script "myNewScript.php" avec des paramètres. Voici le contenu du script :

```
<?php
// . . . code php. . .
function myPhpFunction($p1, $p2) {
  return $p1 . ' '. $p2;
}
// . . . code php. . .
?>
```

Appel avec fonction :

```
C_TEXTE($result)
C_TEXTE($param1)
C_TEXTE($param2)
C_BOOLEEN($isOk)
$param1:= "Hello"
$param2 := "4D world !"
$isOk:=PHP Executer( "C:\\MonDossier\\myNewScript.php";"my-
PhpFunction";$result;$param1;$param2 )
ALERTE($result) // Affiche "Hello 4D world!"
```

- Exemple 3 : Faire quitter l'interpréteur PHP

```
$ifOk:=PHP Executer ("";"quit_4d_php")
```

- Exemple 4 : Gestion des erreurs

```
  // Installation de la méthode de getion d'erreurs
phpCommError :="" // Modifiée par PHPErrorHandler
$T_saveErrorHandler :=Methode appelee sur erreur
APPELER SUR ERREUR("PHPErrorHandler")

  // Exécution du script
C_TEXTE($T_result)
Si(PHP Executer("C:\\MyScripts\\MiscInfos.php";"";$T_result))
      // Pas d'erreur, $T_Result contient le résultat
Sinon
      // Une erreur a été détectée, gérée par PHPErrorHandler
   Si(phpCommError="")
     ... // erreur PHP, utilisez PHP LIRE REPONSE COMPLETE
   Sinon
     ALERTE(phpCommError)
   Fin de si
Fin de si

// Désinstallation de la méthode
APPELER SUR ERREUR ($T_saveErrorHandler)
```

La méthode *PHP_errHandler* est la suivante :

```
phpCommError:=""
LIRE PILE DERNIERE ERREUR(tabCodes; tabComps; tabLibellés)
Boucle ($i;1;Taille tableau(tabCodes))
  phpCommError:=phpCommError+tabCodes{$i}+" "+tabComps{$i}+" "+
        tabLibellés{$i}+Caractere(Retour chariot)
Fin de boucle
```

- Exemple 5 : Création dynamique par 4D d'un script avant son exécution.

```
DOCUMENT VERS BLOB("C:\\Scripts\\MonScript.php";$blobDoc)
Si (OK=1)
  $strDoc:=BLOB vers texte($blobDoc;UTF8 Texte sans longueur)

  $strPosition:=Position("function2Rename";$strDoc)

  $strDoc:=Inserer chaine($strDoc;"_v2";Longueur("function2Rename")
        +$strPosition)

  TEXT VERS BLOB($strDoc;$blobDoc;UTF8 Texte sans longueur )
  BLOB VERS DOCUMENT("C:\\Scripts\\MonScript.php";$blobDoc)
  Si (OK#1)
```

```
    ALERTE("Erreur à la création du script")
  Fin de si
Fin de si
```

Le script est ensuite exécuté :

```
$err:=PHP Executer("C:\\Scripts\\MonScript.php";
            "function2Rename_v2";*)
```

- Exemple 6 : Récupération directe d'une valeur de type date et heure. Voici le contenu du script :

```
<?php
// . . . code php. . .
echo date(DATE_ATOM, mktime(1, 2, 3, 4, 5, 2009));
// . . . code php. . .
?>
```

Réception de la date côté 4D :

```
C_DATE($phpResult_date)
$result :=PHP Executer("C:\php_scripts\ReturnDate.php";"";
                                                    $phpResult_date)
  //$phpResult_date vaut !05/04/2009 !

C_HEURE($phpResult_time)
$result :=PHP Executer("C:\php_scripts\ReturnDate.php";"";
$phpResult_time)

  //$phpResult_time vaut ?01 :02 :03 ?
```

- Exemple 7 : Répartition de données dans des tableaux :

```
TABLEAU TEXTE($arText ;0)
TABLEAU ENTIER LONG($arLong ;0)
$p1 :=","
$p2 := "11,22,33,44,55"
$phpok :=PHP Executer("" ;"explode";$arText;$p1;$p2)
$phpok :=PHP Executer("" ;"explode";$arLong;$p1;$p2)

    // $arText contient les valeurs alpha "11", "22", "33", etc.
    // $arLong contient les numériques, 11, 22, 33, etc.
```

- Exemple 8 : Initialisation d'un tableau :

```
TABLEAU TEXTE($arText ;0)
$phpok :=PHP Executer("";"array_pad";$arText;->$arText;50;"indéfini")
    // Exécute en php : $arrTest = array_pad($arrTest, 50, 'indéfini');
    // Remplit le tableau $arText avec 50 éléments "indéfini"
```

- Exemple 9 : Passage de paramètres via un tableau :

```
TABLEAU ENTIER($arInt;0)
$phpok :=PHP Executer("";"json_decode";$arInt;"[13,51,69,42,7]")
    // Exécute en php : $arInt = json_decode('[13,51,69,42,7]');
    // Remplit le tableau avec des valeurs initiales
```

PHP FIXER OPTION

PHP FIXER OPTION (option; valeur{; *})

Paramètres	Type		Description
option	Entier long	→	Option à définir
valeur	Chaîne | Booléen	→	Nouvelle valeur de l'option
*	*	→	Si passé : la modification ne s'applique qu'à l'appel suivant

La commande PHP FIXER OPTION permet de définir des options spécifiques avant un appel à la commande PHP Executer. La portée de cette commande est le process courant.

Passez dans le paramètre *option* une constante du thème "PHP" désignant l'option à modifier et dans le paramètre *valeur*, la nouvelle valeur de l'option. Voici la description des options :

Option	Description	Valeur(s)
PHP Privilèges	Définition des privilèges utilisateur spécifiques relatifs à l'exécution du script	Chaîne de la forme "Utilisateur:Mot de passe". Par exemple : "root:jd51254d"
PHP Résultat brut	Définition du mode de traitement des en-têtes HTTP renvoyés par PHP dans le résultat de l'exécution lorsque ce résultat est de type texte (lorsque le résultat est de type BLOB, les en-têtes sont toujours conservés).	Booléen *Faux* (valeur par défaut) : supprimer les en-têtes HTTP du résultat. *Vrai* : conserver les en-têtes HTTP

Par défaut, PHP FIXER OPTION définit l'option pour tous les appels à PHP Executer ultérieurs du process. Si vous souhaitez la définir pour le prochain appel uniquement, passez le paramètre étoile (*).

- Exécuter le script "myAdminScript.php" avec des droits d'accès Admin :

```
PHP FIXER OPTION(PHP privilèges; "admin:mypwd"; *)
  `Nous passons le *, les privilèges admin seront utilisés une seule fois
C_TEXTE($result)
C_BOOLEEN($isOK)
$isOK:=PHP Executer("myAdminScript.php";$result)
Si($isOK)
  ALERTE($result)
Fin de si
```

Référence : PHP LIRE OPTION

PHP LIRE OPTION

PHP LIRE OPTION (option; valeur)

Paramètres	Type		Description
option	Entier long	→	Option à lire
valeur	Chaîne | Booléen	←	Valeur courante de l'option

La commande PHP LIRE OPTION permet de connaître la valeur courante d'une option relative à l'éxécution de scripts PHP.

Passez dans le paramètre *option* une constante du thème "PHP" désignant l'option à lire. La commande retourne dans le paramètre *valeur* la valeur courante de l'option. Vous pouvez passer une des constantes suivantes :

- PHP Privilèges : retourne le compte utilisateur courant (le mot de passe n'est pas retourné).
- PHP Résultat brut : retourne le mode de traitement des en-têtes HTTP.

Pour plus d'informations sur ces options, reportez-vous à la description de la commande PHP FIXER OPTION.

- Nous souhaitons connaître le compte d'utilisateur courant :

```
C_TEXTE($userAccount)
C_BLOB($isOK)
$isOK:=PHP LIRE OPTION (PHP privilèges;$userAccount)
Si($isOK)
  ALERTE($userAccount)
Fin de si
```

Référence : PHP FIXER OPTION

PHP LIRE REPONSE COMPLETE

PHP LIRE REPONSE COMPLETE (stdOut{; libellésErr; valeursErr}{; champsEnteteHttp{; valeursEnteteHttp}})

Paramètres	Type		Description
stdOut	Variable Texte/BLOB	←	Contenu du buffer stdOut
libellésErr	Tableau texte	←	Libellés des erreurs
valeursErr	Tableau texte	←	Valeurs des erreurs
chpsEnteteHttp	Tableau texte	←	Noms des en-têtes HTTP
valeursEnteteHttp	Tableau texte	←	Valeurs des en-têtes HTTP

La commande PHP LIRE REPONSE COMPLETE vous permet d'obtenir des informations supplémentaires sur la réponse retournée par l'interpréteur PHP. Cette commande est particulièrement utile en cas d'erreur survenant au cours de l'exécution du script.

Le script PHP peut écrire des données dans le buffer stdOut (echo, print...). Ces données peuvent être récupérées dans la variable *stdOut*.

Les tableaux texte synchronisés *libellésErr* et *valeursErr* sont remplis lorsque l'exécution des scripts PHP provoque des erreurs. Ces tableaux fournissent des informations notamment sur l'origine, le script et la ligne de l'erreur. Ces deux tableaux sont indissociables : si *libellésErr* est passé, *valeursErr* doit être passé également.

Comme les échanges entre 4D et l'interpréteur PHP s'effectuent via FastCGI, l'interpréteur PHP fonctionne comme s'il était appelé par un serveur HTTP et envoie donc des en-têtes HTTP. Vous pouvez récupérer ces en-têtes et leurs valeurs dans les tableaux *champsEnteteHttp* et *valeursEnteteHttp*.

Propriétés des objets

Les commandes de ce thème ont subi de nombreuses modifications dans 4D v12 :

- De nouvelles propriétés sont accessibles via de nouvelles commandes.
- Pour plus de facilité d'utilisation, les commandes existantes ont été préfixées, renommées et complétées si nécessaire par des commandes de "lecture" et d'"écriture" des propriétés symétriques.

OBJET DUPLIQUER

OBJET DUPLIQUER ({*; }objet{;nouvNom{;nouvVar{; reliéA{; dépH{; dépV{; redimH{; redimV}}}}}}}{; *})

Paramètres	Type		Description
*	*	→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est une variable ou un champ
objet	Objet de formulaire	→	Nom d'objet (si * est spécifié) ou Variable ou champ (si * est omis)
nouvNom	Texte	→	Nom du nouvel objet
nouvVar	Pointeur	→	Pointeur vers la variable du nouvel objet
reliéA	Texte	→	Nom de l'objet saisissable (ou du bouton radio) précédent
dépH	Entier long	→	Décalage horizontal du nouvel objet (>0 = vers la droite, <0 = vers la gauche)
dépV	Entier long	→	Décalage vertical du nouvel objet (>0 = vers le bas, <0 = vers le haut)
redimH	Entier long	→	Redimensionnement horizontal du nouvel objet
redimV	Entier long	→	Redimensionnement vertical du nouvel objet
*	*	→	Si spécifié = coordonnées absolues Si omis = coordonnées relatives

La nouvelle commande OBJET DUPLIQUER permet de créer une copie de l'objet désigné par le paramètre *objet*. La copie est générée dans le contexte du formulaire en cours d'exécution (mode Application). Le formulaire d'origine, généré en mode Développement, n'est pas modifié.

Par défaut, toutes les options définies dans la Liste des propriétés pour l'objet source sont appliquées à la copie (taille, options de redimensionnement, couleur, etc.), y compris la méthode objet éventuellement associée.

Les exceptions suivantes sont toutefois à noter :

- Bouton par défaut : il ne peut y avoir qu'un seul bouton par défaut dans un formulaire. Lorsque vous dupliquez un bouton ayant la propriété "Bouton par défaut", cette propriété est attribuée à la copie et est supprimée de l'objet d'origine.
- Equivalents clavier : le raccourci clavier associé à un objet source n'est pas dupliqué. Cette propriété est laissée vide dans la copie.

- Noms d'objet : il ne peut pas y avoir plusieurs objets de même nom dans un formulaire. Si vous ne passez pas le paramètre *nouvNom*, le nom de l'objet source est automatiquement incrémenté dans le nouvel objet (cf. ci-dessous).

Si vous passez le paramètre optionnel *****, vous indiquez que le paramètre *objet* désigne le nom d'un objet (une chaîne). Si vous ne passez pas le paramètre *****, vous indiquez que le paramètre *objet* désigne un champ ou une variable. Dans ce cas, vous ne passez pas une chaîne de caractères mais la référence du champ ou de la variable (champs ou variables objets uniquement).
Si vous passez une référence de champ ou de variable et si le formulaire contient plusieurs objets utilisant la même référence, la première occurrence trouvée est utilisée. Dans ce cas, pour éviter toute ambiguïté, il est conseillé de travailler avec les noms d'objets, qui sont uniques.

Passez dans le paramètre *nouvNom* le nom attribué à la copie de l'objet. Ce nom doit être conforme aux règles de nommage des objets et être unique dans le formulaire. S'il est invalide ou est déjà utilisé par un autre objet, la commande ne fait rien et la variable *OK* retourne 0.
Si vous omettez ce paramètre ou passez une chaîne vide, le nouveau nom est automatiquement généré par incrémentation du nom de l'objet source (si ce nom n'est pas déjà utilisé). Par exemple :

Nom d'origine	Nom de la copie
Bouton	Bouton 1
Bouton20	Bouton21
Bouton21	Bouton23 si Bouton22 existe déjà

Passez dans *nouvVar* un pointeur vers la variable à associer au nouvel objet. Vous devez en principe pointer vers une variable du même type que celle de l'objet d'origine mais certains "retypages" sont possibles. La commande propose des automatismes facilitant l'écriture de code générique :

- De manière générale, toutes les variables "saisissables" peuvent être retypées ; par exemple, un objet affichant une Date ou un Entier long peut être dupliqué et utilisé avec une variable de type Texte. Les propriétés compatibles sont maintenues. La commande permet également les changements de type entre des objets Texte et des objets Image. Un objet texte dupliqué et associé à une variable ou un champ booléen aura automatiquement l'apparence d'une case à cocher.

- Il est généralement possible de transformer dynamiquement une variable en champ et inversement.

  En revanche, les objets graphiques (boutons, cases à cocher...) ne peuvent pas être tranformés en d'autres types de contrôles.

  Si le type de la variable est incompatible avec l'objet, la commande ne fait rien et la variable OK prend la valeur 0.
  Si vous omettez ce paramètre, la variable est créée dynamiquement par 4D (cf. paragraphe "Variables dynamiques", page 256).
  Si vous dupliquez un objet statique (ligne, rectangle, image statique...) ce paramètre est ignoré. Passez un pointeur Nil (->[]) si vous souhaitez pouvoir utiliser les autres paramètres.

  Vous utilisez le paramètre *reliéA* dans deux cas :

- mise à jour de l'ordre de saisie : dans ce cas, passez dans *reliéA* le nom de l'objet saisissable situé juste avant l'objet dupliqué. Si vous souhaitez que le nouvel objet devienne le premier objet dans l'ordre de saisie de la page, passez la nouvelle constante Objet Premier ordre saisie (cf. commande OBJET Lire pointeur, page 163).

- association à un groupe de boutons radio : les boutons radios fonctionnent de façon coordonnée lorsqu'ils sont groupés. Si l'objet dupliqué est un bouton radio, passez dans *reliéA* le nom d'un bouton radio du groupe auquel rattacher le nouvel objet.

  Si vous omettez ce paramètre ou passez une chaîne vide, le nouvel objet devient le dernier objet saisissable de la page du formulaire. Dans le cas d'un bouton radio, l'objet est rattaché au groupe du bouton source.

  Le nouvel objet peut être déplacé et redimensionné via les paramètres *dépH*, *dépV*, *redimH* et *redimV*. Comme pour la commande OBJET DEPLACER, le sens du déplacement ou du redimensionnement est défini par le signe des valeurs passées dans les paramètres *dépH* et *dépV* :

- Si la valeur est positive, le déplacement ou le redimensionnement s'effectue respectivement vers la droite ou vers le bas.

- Si la valeur est négative, le déplacement ou le redimensionnement s'effectue respectivement vers la gauche ou vers le haut.

Par défaut, les valeurs de *dépH*, *dépV*, *redimH* et *redimV* modifient les coordonnées de l'objet relativement à sa position précédente. Si vous souhaitez que ces paramètres définissent des coordonnées absolues, passez le dernier paramètre optionnel *.

Si vous omettez ces paramètres, le nouvel objet se superpose à l'objet d'origine.

Cette commande doit être utilisée dans le contexte de l'affichage d'un formulaire. Elle sera généralement appelée dans l'événement Sur chargement du formulaire ou suite à une action utilisateur (événement Sur clic).

Note Si l'événement Sur chargement est associé à l'objet d'origine, il est généré pour l'objet dupliqué au moment de l'exécution de la commande. Ce principe permet par exemple d'initialiser la valeur de l'objet.

Pour des raisons techniques et logiques, OBJET DUPLIQUER ne peut pas être appelée dans le cadre de certains événements formulaire, notamment :

- Evénement Sur chargement généré dans une méthode objet
- Evénement Sur libération,
- Evénement lié à un contexte d'impression (Sur entête, Sur impression corps, etc.). Pour imprimer plusieurs fois un objet, il est préférable d'utiliser la nouvelle commande Imprimer objet.

Lorsque la commande est appelée dans un contexte non pris en charge, l'objet n'est pas dupliqué et la variable *OK* prend la valeur 0. Si elle est appelée dans un contexte d'impression, l'erreur -10601 est en outre générée.

Si la commande est exécutée correctement, la variable OK prend la valeur 1. Sinon, elle prend la valeur 0.

▸ Création d'un nouveau bouton nommé "BoutonAnnul" au-dessus de l'objet existant "BoutonOK" et association à la variable *vAnnul* :

```
OBJET DUPLIQUER(*;"BoutonOK";"BoutonAnnul";vAnnul)
```

- Création d'un nouveau bouton radio "bRadio6" basé sur le bouton radio existant "bRadio5". Ce bouton sera associé à la variable <>r6, intégré au groupe du bouton "bRadio5" et placé 20 pixels au-dessous :

  OBJET DUPLIQUER(*;"bRadio5";"bRadio6";<>r6;"bRadio5";0;20)

OBJET FIXER ACTIVATION

OBJET FIXER ACTIVATION({*; }objet; actif)

Paramètres	Type		Description
*	*	→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est une variable ou un champ
objet	Objet de formulaire	→	Nom d'objet (si * est spécifié) ou Variable (si * est omis)
actif	Booléen	→	Vrai = objet(s) activé(s), Faux sinon

La nouvelle commande OBJET FIXER ACTIVATION permet d'activer ou d'inactiver l'objet ou le groupe d'objets désigné par *objet* dans le formulaire courant.

Un objet activé réagit aux clics souris et aux raccourcis clavier.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre *objet* est un nom d'objet (une chaîne). Si vous ne passez pas le paramètre, vous indiquez que le paramètre *objet* est une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de variable (variable objet uniquement).

Cette commande peut être appliquée aux types d'objets suivants :

- Bouton, Bouton par défaut, Bouton 3D, Bouton invisible, Bouton inversé
- Bouton radio, Bouton radio 3D, Bouton image
- Case à cocher, Case à cocher 3D
- Pop-up menu, Liste déroulante, Combo Box, Menu/Liste déroulante
- Thermomètre, Règle

Note Cette commande est sans effet avec un objet auquel une action standard a été assignée (4D se charge de modifier l'état de cet objet lorsque c'est nécessaire), sauf dans le cas des actions **Valider** et **Annuler**.

Référence : OBJET Lire activation, ACTIVER BOUTON, INACTIVER BOUTON

OBJET FIXER ATTRIBUT TEXTE STYLE

OBJET FIXER ATTRIBUT TEXTE STYLE({*; }objet; débutSel; finSel; nomAtt; valeurAtt{; nomAtt2; valeurAtt2;...;nomAttN; valeurAttN})

Paramètres	Type		Description
*	*	→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est une variable ou un champ
objet	Objet de formulaire	→	Nom d'objet (si * est spécifié) ou Variable ou champ (si * est omis)
débutSel	Numérique	→	Début de la nouvelle sélection de texte
finSel	Numérique	→	Fin de la nouvelle sélection de texte
nomAtt	Entier long	→	Attribut à définir
valeurAtt	Chaîne | Numérique	→	Nouvelle valeur d'attribut

La nouvelle commande OBJET FIXER ATTRIBUT TEXTE STYLE permet de modifier un ou plusieurs attribut(s) de style dans le ou les objet(s) de formulaire désigné(s) par *objet*.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre *objet* est un nom d'objet (une chaîne). Si vous ne passez pas le paramètre, vous indiquez que le paramètre *objet* est un champ ou une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de champ ou de variable.

La définition d'un attribut s'effectue via l'insertion ou la modification de balises HTML de style dans à l'intérieur du texte (cf. paragraphe "Propriétés de gestion du texte riche", page 97). A noter que OBJET FIXER ATTRIBUT TEXTE STYLE insère des balises de style dans tous les cas, même si *objet* désigne des objets texte de formulaire n'ayant pas la propriété Multistyle.

Note L'ajout de balise augmente la taille de la chaîne. Si vous modifiez les attributs de style de champs Alpha, veillez à ce que la longueur de la chaîne résultante n'excède pas la taille maximale du champ. Dans le cas contraire, les données seront tronquées.

Les paramètres *débutSel* et *finSel* permettent de désigner la sélection de texte à laquelle appliquer la ou les modification(s) de style à l'intérieur de l'*objet*. Passez dans *débutSel* la position du premier caractère à modifier et dans *finSel* la position plus un du dernier caractère à modifier.

Si les valeurs de *débutSel* et *finSel* sont égales ou si *débutSel* est supérieur à *finSel*, une erreur est retournée. Si la valeur de *finSel* est supérieure au nombre de caractères de l'*objet*, tous les caractères entre *débutSel* et la fin du texte seront modifiés.
Les valeurs *débutSel* et *finSel* ne tiennent pas compte des balises de style éventuellement déjà présentes dans la zone. Elles sont évaluées sur la base du texte brut (texte duquel les balises de style ont été filtrées).

Passez dans les paramètres *nomAtt* et *valeurAtt* respectivement le nom et la valeur de l'attribut à modifier. Vous pouvez passer autant de paires attribut/valeur que vous souhaitez.

Pour définir le paramètre *nomAtt*, utilisez les constantes prédéfinies placées dans le thème "Attributs de texte multistyle". La valeur à passer dans le paramètre *valeurAtt* dépend du paramètre *nomAtt* :

nomAtt	***valeurAtt***
Attribut style gras (1)	0 = supprimer attribut gras de la sélection 1= appliquer attribut gras à la sélection
Attribut style italique (2)	0 = supprimer attribut italique de la sélection 1= appliquer attribut italique à la sélection
Attribut style barré (3)	0 = supprimer attribut barré de la sélection 1= appliquer attribut barré à la sélection
Attribut style souligné (4)	0 = supprimer attribut souligné de la sélection 1= appliquer attribut souligné à la sélection
Attribut nom de police (5)	Nom de la police (chaîne)
Attribut taille texte (6)	Nombre de points (numérique)
Attribut couleur texte (7)	Valeurs hexadécimales ou noms de couleurs HTML (cf. paragraphe "Couleurs")
Attribut couleur fond (8)	(Windows uniquement) Valeurs hexadécimales ou noms de couleurs HTML (cf. paragraphe "Couleurs")

Couleurs
Si vous passez la constante Attribut couleur texte ou Attribut couleur fond dans *nomAtt*, vous devez passer dans *valeurAtt* une chaîne contenant soit un nom de couleur HTML soit une valeur de couleur hexadécimale :

Nom de couleur HTML	**Valeur hexa**
Aqua	#00FFFF
Black	#000000

Blue	#0000FF
Fushia	#FF00FF
Gray	#808080
Green	#008000
Lime	#00FF00
Maroon	#800000
Navy	#000080
Olive	#808000
Purple	#800080
Red	#FF0000
Silver	#C0C0C0
Teal	#008080
White	#FFFFFF
Yellow	#FFFF00

Le paramètre *valeurAtt* peut être de type texte, booléen, entier, réel, date ou heure. Si vous passez des attributs ou des valeurs invalides, 4D retourne une erreur.

- Dans cet exemple, nous modifions la taille, la couleur de texte ainsi que les attributs gras et souligné des caractères 2 à 5 du champ :

OBJET FIXER ATTRIBUT TEXTE STYLE([MaTable]MonChamp;2;5;
Attribut nom de police;"Arial";Attribut taille texte; 10;
Attribut style souligné; 1; Attribut style gras;
1;Attribut couleur texte;"Blue")

OBJET FIXER DEFILEMENT

OBJET FIXER DEFILEMENT({*; }objet{; positionV*{; positionH}*}{; *})

Paramètres	Type		Description
*	*	→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est une variable*, un champ* ou une table
objet	Objet de formulaire	→	Nom d'objet (si * est spécifié) ou Variable *ou champ* ou table (si * est omis)
positionV	Entier long	→	Numéro de ligne à afficher *ou Défilement vertical en pixels (images)*
positionH	Entier long	→	*Numéro de colonne à afficher (list box) ou Défilement horizontal en pixels (images)*
*	*	→	Afficher la ligne (*et la colonne si le paramètre positionH est passé*) en première position après défilement

Note La commande OBJET FIXER DEFILEMENT était nommée DEFILER LIGNES et était placée dans le thème "Interface utilisateur" dans les versions précédentes de 4D.

La portée de la commande OBJET FIXER DEFILEMENT a été étendue dans 4D v12. Outre le défilement vertical des sous-formulaires, formulaires liste, listes hiérarchiques et list box, cette commande peut également provoquer le défilement horizontal des list box et le défilement vertical et horizontal des images.

- Défilement horizontal d'une list box : si *objet* désigne une list box, vous pouvez passer dans le paramètre *positionH* un numéro de colonne. L'exécution de la commande provoquera dans ce cas le défilement horizontal de la list box de manière à ce que la colonne soit visible. Si la colonne est déjà visible, la commande ne fait rien.
Comme pour le défilement vertical, si vous passez le second paramètre optionnel *****, la colonne rendue visible par la commande (si la list box a effectivement défilé) sera placée en première position.

Note Gardez à l'esprit que cette commande se base toujours sur la représentation "standard" (non hiérarchique) d'une list box, même si elle est affichée en mode hiérarchique. Par conséquent, le résultat d'une instruction OBJET FIXER DEFILEMENT pourra être différent suivant que la list box est affichée en mode standard ou en mode hiérarchique.

- Défilement des images : si *objet* désigne une image affichée dans le formulaire, vous pouvez faire défiler son contenu à l'aide de la commande OBJET FIXER DEFILEMENT. L'image doit être affichée dans le format "Image tronquée (non centrée)". Passez dans *positionV* le décalage vertical et dans *positionH* le décalage horizontal à appliquer à l'image. Passez 0 dans *positionV* dans pour ne pas faire défiler l'image dans la dimension verticale. Les valeurs doivent être exprimées en pixels relativement à l'origine de l'image dans son contexte local.

Note Le défilement par programmation d'un objet reste possible même si les barres de défilement ont été masquées dans le formulaire.

- Cet exemple illustre la différence de fonctionnement de la commande avec une list box affichée en mode standard et hiérarchique :

```
OBJET FIXER DEFILEMENT (*;"malistbox";4;2;*)
  // afficher en tête la 4e ligne de la 2e colonne de la list box
```

Si cette instruction est appliquée à une list box affichée en mode standard :

France	Bretagne	Brest	120000	...
France	Bretagne	Quimper	80000	...
France	Bretagne	Rennes	200000	...
France	**Normandie**	Caen	220000	...
France	Normandie	Deauville	4000	...
France	Normandie	Cherbourg	41000	...
...	...	...	...	...

... les lignes et les colonnes de la list box défilent effectivement :

Normandie	Caen	220000	...	...
Normandie	Deauville	4000	...	...
Normandie	Cherbourg	41000	...	...
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...

En revanche, si la même instruction est appliquée à la list box affichée en mode hiérarchique, les lignes défilent mais pas les colonnes car la 2e colonne appartient à la hiérarchie :

V	**Normandie**	
	Caen	220000
	Deauville	4000
	Cherbourg	41000
	...	

Référence : OBJET LIRE DEFILEMENT, OBJET LIRE BARRES DEFILEMENT

OBJET FIXER FORMATAGE

OBJET FIXER FORMATAGE({*; }objet; formatAffich)

Note La commande OBJET FIXER FORMATAGE était nommée CHOIX FORMATAGE dans les versions précédentes de 4D.

Dans le cas des thermomètres et règles, un nouveau sous-paramètre est disponible dans la suite de sous-paramètres *formatAffich* :

min;max;unité;pas;mode{;format*{;affichage}*}

- Pour les règles, ce sous-paramètre peut prendre les valeurs suivantes :
  - *affichage* = 0 (ou est omis) : afficher une règle standard.
  - *affichage* = 1 : activer le mode "Stepper". Pour plus d'informations, reportez-vous au paragraphe “Nouvel objet stepper”, page 91.
- Pour les thermomètres (indicateurs de progression), ce sous-paramètre n’est pris en compte que si le sous-paramètre *mode* vaut 128 (indéterminé). Dans ce cas :
  - *affichage* = 0 (ou est omis) : afficher un thermomètre en animation continue de type "barber shop".
  - *affichage* = 1 : activer le mode "Progression asynchrone". Pour plus d'informations, reportez-vous au paragraphe “Indicateur de progression asynchrone”, page 92.

OBJET FIXER TEXTE STYLE

OBJET FIXER TEXTE STYLE({*; }objet; nouvTexte{; débutSel{; finSel}})

Paramètres	Type		Description
*	*	→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est une variable ou un champ
objet	Objet de formulaire	→	Nom d'objet (si * est spécifié) ou Variable ou champ (si * est omis)
nouvTexte	Chaîne	→	Texte à insérer
débutSel	Entier long	→	Début de la sélection
finSel	Entier long	→	Fin de la sélection

La nouvelle commande OBJET FIXER TEXTE STYLE insère le texte passé dans le paramètre *nouvTexte* dans le champ ou la variable de texte multistyle désigné(e) par le paramètre *objet*.

Cette commande s'applique uniquement au texte brut du paramètre *objet*, sans modifier les éventuelles balises de style qu'il contient. Elle permet de modifier par programmation du texte stylé affiché à l'écran.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre *objet* est un nom d'objet (une chaîne). Si vous ne passez pas le paramètre, vous indiquez que le paramètre *objet* est un champ ou une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de champ ou de variable.

Passez dans *nouvTexte* le texte à insérer.

Les paramètres optionnels *débutSel* et *finSel* vous permettent de désigner une sélection de texte dans *objet*. Les valeurs *débutSel* et *finSel* expriment une sélection de texte brut, sans tenir compte des balises de style éventuellement présentes dans le texte. L'action de la commande diffère en fonction des paramètres facultatifs *débutSel* et *finSel* :

- si vous omettez *débutSel* et *finSel*, OBJET FIXER TEXTE STYLE remplace la totalité du texte de *objet* par *nouvTexte*,
- si vous passez uniquement *débutSel*, OBJET FIXER TEXTE STYLE insère le texte *nouvTexte* dans *objet* à partir de *débutSel*,
- si vous passez *débutSel* et *finSel*, OBJET FIXER TEXTE STYLE remplace le texte brut défini par ces bornes avec le texte *nouvTexte*.

Si les valeurs de *débutSel* et *finSel* sont égales ou si *débutSel* est supérieur à *finSel*, une erreur est retournée.

▸ Vous souhaitez remplacer le texte stylé sélectionné par l'utilisateur avec le contenu de la variable *vtempo* :

TEXTE SELECTIONNE([Produits]Notes;vDebut;vFin)
OBJET FIXER TEXTE STYLE([Produits]Notes;vtempo;vDebut;vFin)

Référence : OBJET Lire texte style, OBJET Lire texte brut

OBJET Lire activation

OBJET Lire activation ({*; }objet) → Booléen

Paramètres	Type		Description
*	*	→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est une variable ou un champ
objet	Objet de formulaire	→	Nom d'objet (si * est spécifié) ou Variable (si * est omis)
Résultat	Booléen	←	Vrai = objet(s) activé(s), Faux sinon

La nouvelle commande OBJET Lire activation retourne Vrai si l'objet ou le groupe d'objets désigné par *objet* est activé dans le formulaire et Faux s'il est inactivé.

Un objet activé réagit aux clics souris et aux raccourcis clavier.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre *objet* est un nom d'objet (une chaîne). Si vous ne passez pas le paramètre, vous indiquez que le paramètre *objet* est une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de variable (variable objet uniquement).

Cette commande peut être appliquée aux types d'objets suivants :

- Bouton, Bouton par défaut, Bouton 3D, Bouton invisible, Bouton inversé
- Bouton radio, Bouton radio 3D, Bouton image
- Case à cocher, Case à cocher 3D
- Pop-up menu, Liste déroulante, Combo Box, Menu/Liste déroulante
- Thermomètre, Règle

Référence : OBJET FIXER ACTIVATION

OBJET LIRE ATTRIBUT TEXTE STYLE

OBJET LIRE ATTRIBUT TEXTE STYLE({*; }objet; débutSel; finSel; nomAtt; valeurAtt{; nomAtt2; valeurAtt2;...;nomAttN; valeurAttN})

Paramètres	Type		Description
*	*	→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est une variable ou un champ
objet	Objet de formulaire	→	Nom d'objet (si * est spécifié) ou Variable ou champ (si * est omis)
débutSel	Numérique	→	Début de la sélection de texte
finSel	Numérique	→	Fin de la sélection de texte
nomAtt	Entier long	→	Attribut à lire
valeurAtt	Variable	←	Valeur courante de l'attribut

La nouvelle commande OBJET LIRE ATTRIBUT TEXTE STYLE permet de récupérer la valeur courante d'un attribut de style dans une sélection de texte du ou des objet(s) de formulaire désigné(s) par *objet*.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre *objet* est un nom d'objet (une chaîne). Si vous ne passez pas le paramètre, vous indiquez que le paramètre *objet* est un champ ou une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de champ ou de variable.

Les paramètres *débutSel* et *finSel* permettent de désigner la sélection de texte de l'*objet* de laquelle lire l'attribut de style. Passez dans *débutSel* la position du premier caractère et dans *finSel* la position plus un du dernier caractère de la sélection.
Si les valeurs de *débutSel* et *finSel* sont égales ou si *débutSel* est supérieur à *finSel*, une erreur est retournée.
Les valeurs *débutSel* et *finSel* ne tiennent pas compte des balises de style éventuellement déjà présentes dans la zone. Elles sont évaluées sur la base du texte brut (texte duquel les balises de style ont été filtrées).

Passez dans le paramètre *nomAtt* le nom de l'attribut à lire. Pour cela, vous devez utiliser l'une des constantes du thème "Attributs de texte multistyle". Pour plus d'informations, reportez-vous à la description de la commande OBJET FIXER ATTRIBUT TEXTE STYLE, page 203. Passez dans le paramètre *valeurAtt* une variable devant récupérer la valeur courante de l'attribut.
Vous pouvez passer autant de paires attribut/valeur que vous souhaitez.

Si la valeur de l'attribut *nomAtt* est identique dans la totalité de la sélection, elle est retournée dans *valeurAtt*. Si cette valeur est différente ou si *objet* ne contient pas de balises SPAN, les valeurs suivantes sont retournées :

nomAtt	*valeurAtt* si attribut hétérogène dans la sélection ou pas de balises SPAN
Attribut nom de police	"" (chaîne vide)
Attribut taille texte	2
Attribut couleur texte	FFFFFFFF
Attribut couleur fond	FFFFFFFF
Attribut style gras	2
Attribut style italique	2
Attribut style souligné	2
Attribut style barré	2

OBJET LIRE BARRES DEFILEMENT

OBJET LIRE BARRES DEFILEMENT({*; }objet; horizontale; verticale)

Paramètres	Type		Description
*	*	→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est une variable ou un champ
objet	Objet de formulaire	→	Nom d'objet (si * est spécifié) ou Variable ou champ (si * est omis)
horizontale	Booléen	←	Vrai=affichée, Faux=cachée
verticale	Booléen	←	Vrai=affichée, Faux=cachée

La nouvelle commande OBJET LIRE BARRES DEFILEMENT permet de connaître le statut affiché/masqué des barres de défilement horizontale et verticale de l'objet ou du groupe d'objets désigné(s) par *objet*.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre *objet* est un nom d'objet (une chaîne). Si vous ne passez pas le paramètre, vous indiquez que le paramètre *objet* est un champ ou une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de champ ou de variable (champ ou variable objet uniquement).

Cette commande est utilisable avec les objets de formulaire suivants :

- list box,
- zones de défilement,
- listes hiérarchiques,
- sous-formulaires.

OBJET LIRE COULEURS RVB

OBJET LIRE COULEURS RVB({*; }objet; couleurAvantPlan{; couleurArrière-Plan{; couleurArrièrePlanAlt}})

Paramètres	Type		Description
*	*	→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est une variable ou un champ
objet	Objet de formulaire	→	Nom d'objet (si * est spécifié) ou Variable ou champ (si * est omis)
couleurAvantPlan	Entier long	←	Valeur de la couleur RVB d'avant-plan
couleurArrièrePlan	Entier long	←	Valeur de la couleur RVB d'arrière-plan
couleurArrièrePlanAlt	Entier long	←	Valeur de la couleur RVB d'arrière-plan alternée

La nouvelle commande OBJET LIRE COULEURS RVB retourne les couleurs d'avant-plan et d'arrière-plan de l'objet ou du groupe d'objets désigné(s) par *objet*.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre *objet* est un nom d'objet (une chaîne). Si vous ne passez pas le paramètre, vous indiquez que le paramètre *objet* est un champ ou une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de champ ou de variable (champ ou variable objet uniquement).

Lorsque la commande est appliquée à un objet de type List box, la couleur d'arrière plan alternée des lignes peut être retournée dans le paramètre *couleurArrièrePlanAlt*. Dans ce cas, la valeur de *couleurArrièrePlan* est utilisée pour le fond des lignes impaires uniquement.

Les valeurs de couleurs RVB retournées dans les paramètres *couleurAvantPlan*, *couleurArrièrePlan* et *couleurArrièrePlanAlt* sont des entiers longs de 4 octets dans format (0x00RRGGBB). Pour plus d'informations sur ce format, reportez-vous à la description de la commande OBJET FIXER COULEURS RVB.

Référence : OBJET FIXER COULEURS RVB

OBJET LIRE DEFILEMENT

OBJET LIRE DEFILEMENT({*; }objet; positionV{; positionH})

Paramètres	Type		Description
*	*	→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est une variable, un champ ou une table
objet	Objet de formulaire	→	Nom d'objet (si * est spécifié) ou Variable ou champ ou table (si * est omis)
positionV	Entier long	←	Numéro de la première ligne affichée ou Défilement vertical en pixels (images)
positionH	Entier long	←	Numéro de la première colonne affichée (list box) ou Défilement horizontal en pixels (images)

La nouvelle commande OBJET LIRE DEFILEMENT retourne dans les paramètres *positionV* et *positionH* des informations relatives à la position des barres de défilement de l'objet de formulaire désigné par les paramètres * et *objet*.

Si vous passez le premier paramètre optionnel *, vous indiquez que le paramètre *objet* est le nom d'un objet de type sous-formulaire, liste hiérarchique, zone de défilement, list box ou image (dans ce cas, passez une chaîne dans objet). Si vous ne passez pas ce paramètre, vous indiquez que le paramètre objet est une table (table du formulaire liste ou du sous-formulaire) ou une variable (*RefList* de liste hiérarchique, variable image ou list box).

- Si *objet* désigne un objet de type liste (sous-formulaire, formulaire liste, liste hiérarchique, zone de défilement ou list box), *positionV* retourne le numéro de la première ligne affichée dans l'objet. *positionH* (list box uniquement) retourne le numéro de la première colonne entièrement visible dans la partie gauche de la list box. Avec les autres types d'objets, ce paramètre retourne 0.
- Si *objet* désigne une image (variable ou champ), *positionV* retourne le décalage vertical et *positionH* le décalage horizontal de l'image. Ces valeurs sont exprimées en pixels par rapport à l'origine de l'image dans le système de coordonnées locales.

Référence : OBJET FIXER DEFILEMENT

OBJET Lire filtre saisie

OBJET Lire filtre saisie ({*; }objet) → Chaîne

Paramètres	Type		Description
*	*	→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est une variable ou un champ
objet	Objet de formulaire	→	Nom d'objet (si * est spécifié) ou Variable ou champ (si * est omis)
Résultat	Chaîne	←	Nom du filtre de saisie

La nouvelle commande OBJET Lire filtre saisie retourne le nom du filtre de saisie éventuellement associé à l'objet ou au groupe d'objets désigné par *objet*.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre *objet* est un nom d'objet (une chaîne). Si vous ne passez pas le paramètre, vous indiquez que le paramètre *objet* est un champ ou une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de champ ou de variable (champ ou variable objet uniquement).

Référence : OBJET FIXER FILTRE SAISIE

OBJET Lire nom enumeration

OBJET Lire nom enumeration ({*; }objet) → Chaîne

Paramètres	Type		Description
*	*	→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est une variable ou un champ
objet	Objet de formulaire	→	Nom d'objet (si * est spécifié) ou Variable ou champ (si * est omis)
Résultat	Chaîne	←	Nom de l'énumération (définie en mode Développement)

La nouvelle commande OBJET Lire nom enumeration retourne le nom de l'énumération associée à l'objet ou au groupe d'objets désigné par *objet*.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre *objet* est un nom d'objet (une chaîne). Si vous ne passez pas le paramètre, vous indiquez que le paramètre *objet* est un champ ou une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de champ ou de variable (champ ou variable objet uniquement).

Référence : OBJET FIXER NOM ENUMERATION

OBJET Lire police

OBJET Lire police ({*; }objet) → Chaîne

Paramètres	Type		Description
*	*	→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est une variable ou un champ
objet	Objet de formulaire	→	Nom d'objet (si * est spécifié) ou Variable ou champ (si * est omis)
Résultat	Chaîne	←	Nom de la police

La nouvelle commande OBJET Lire police retourne le nom de la police de caractères utilisée par le ou les objet(s) de formulaire désigné(s) par *objet*.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre *objet* est un nom d'objet (une chaîne). Si vous ne passez pas le paramètre, vous indiquez que le paramètre *objet* est un champ ou une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de champ ou de variable (champ ou variable objet uniquement).

Référence : OBJET FIXER POLICE

OBJET Lire saisissable

OBJET Lire saisissable ({*; }objet) → Booléen

Paramètres	Type		Description
*	*	→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est une variable ou un champ
objet	Objet de formulaire	→	Nom d'objet (si * est spécifié) ou Variable ou champ (si * est omis)
Résultat	Booléen	←	Vrai = objet(s) saisissable(s), Faux sinon

La nouvelle commande OBJET Lire saisissable retourne Vrai si l'objet ou le groupe d'objets désigné par *objet* dispose de l'attribut saisissable et Faux sinon.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre *objet* est un nom d'objet (une chaîne). Si vous ne passez pas le paramètre, vous indiquez que le paramètre *objet* est un champ ou une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de champ ou de variable (champ ou variable objet uniquement).

Référence : OBJET FIXER SAISISSABLE

OBJET Lire style police

OBJET Lire style police ({*; }objet) → Entier long

Paramètres	Type		Description
*	*	→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est une variable ou un champ
objet	Objet de formulaire	→	Nom d'objet (si * est spécifié) ou Variable ou champ (si * est omis)
Résultat	Entier long	←	Style de police

La nouvelle commande OBJET Lire style police retourne le style courant de la police de caractères utilisée par le ou les objet(s) de formulaire désigné(s) par *objet*.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre *objet* est un nom d'objet (une chaîne). Si vous ne passez pas le paramètre, vous indiquez que le paramètre *objet* est un champ ou une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de champ ou de variable (champ ou variable objet uniquement).

Vous pouvez comparer la valeur retournée à la valeur d'une ou plusieurs des constantes prédéfinies suivantes, placées dans le thème "Styles de caractères" :

Constante	Type	Valeur
Normal	Entier long	0
Gras	Entier long	1
Italique	Entier long	2
Souligné	Entier long	4

Référence : OBJET FIXER STYLE POLICE

OBJET Lire taille police

OBJET Lire taille police ({*; }objet) → Entier long

Paramètres	Type		Description
*	*	→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est une variable ou un champ
objet	Objet de formulaire	→	Nom d'objet (si * est spécifié) ou Variable ou champ (si * est omis)
Résultat	Entier long	←	Taille de la police en points

La nouvelle commande OBJET Lire taille police retourne la taille (en points) de la police de caractères utilisée par le ou les objet(s) de formulaire désigné(s) par *objet*.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre *objet* est un nom d'objet (une chaîne). Si vous ne passez pas le paramètre, vous indiquez que le paramètre *objet* est un champ ou une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de champ ou de variable (champ ou variable objet uniquement).

Référence : OBJET FIXER TAILLE POLICE

OBJET Lire texte brut

OBJET Lire texte brut ({*; }objet) → Texte

Paramètres	Type		Description
*	*	→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est une variable ou un champ
objet	Objet de formulaire	→	Nom d'objet (si * est spécifié) ou Variable ou champ (si * est omis)
Résultat	Texte	←	Texte sans balises

La nouvelle commande OBJET Lire texte brut supprime du champ ou de la variable texte désigné(e) par les paramètres * et *objet* toute balise de style et retourne le texte brut.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre *objet* est un nom d'objet (une chaîne). Si vous ne passez pas le paramètre, vous indiquez que le paramètre *objet* est un champ ou une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de champ ou de variable.

- Vous cherchez le texte "très beau" parmi les valeurs d'un champ texte multistyle. La valeur a été stockée sous la forme "Il fait très beau **aujourd'hui**".

```
CHERCHER PAR FORMULE([Commentaires];
                     OBJET Lire texte brut([Commentaires]Meteo)="@très beau@")
```

Note Dans ce contexte, l'instruction suivante ne donnera pas le résultat escompté car le texte est enregistré avec des balises de style : **CHERCHER**([Commentaires];[Commentaires]Meteo="@très beau@")

OBJET Lire texte style

OBJET Lire texte style({*; }objet{; débutSel{; finSel}}) → Texte

Paramètres	Type		Description
*	*	→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est une variable ou un champ
objet	Objet de formulaire	→	Nom d'objet (si * est spécifié) ou Variable ou champ (si * est omis)
débutSel	Entier long	→	Début de la sélection
finSel	Entier long	→	Fin de la sélection
Résultat	Texte	←	Texte incluant les balises de style

La nouvelle commande OBJET Lire texte style retourne le texte stylé présent dans le champ ou la variable de texte multistyle désigné(e) par le paramètre *objet*.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre *objet* est un nom d'objet (une chaîne). Si vous ne passez pas le paramètre, vous indiquez que le paramètre *objet* est un champ ou une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de champ ou de variable.

La commande retourne le texte avec les éventuelles balises de style qui lui sont associées, ce qui permet par exemple de copier et coller du texte en conservant les styles.

Les paramètres optionnels *débutSel* et *finSel* vous permettent de désigner une sélection de texte dans *objet*. Les valeurs *débutSel* et *finSel* expriment une sélection de texte brut, sans tenir compte des balises de style éventuellement présentes.

- si vous omettez *débutSel* et *finSel*, OBJET Lire texte style retourne la totalité du texte contenu dans *objet*,
- si vous passez *débutSel* et *finSel*, OBJET Lire texte style retourne la sélection de texte définie par ces bornes.

Si les valeurs de *débutSel* et *finSel* sont égales ou si *débutSel* est supérieur à *finSel*, une erreur est retournée.

Référence : OBJET FIXER TEXTE STYLE, OBJET Lire texte brut

OBJET Lire titre

OBJET Lire titre ({*; }objet) → Chaîne

Paramètres	Type		Description
*	*	→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est une variable ou un champ
objet	Objet de formulaire	→	Nom d'objet (si * est spécifié) ou Variable ou champ (si * est omis)
Résultat	Chaîne	←	Titre du bouton

La nouvelle commande OBJET Lire titre retourne le titre (libellé) du ou des objet(s) de formulaire désigné(s) par *objet*. Cette commande ne peut être utilisée qu'avec des objets de type "bouton" affichant du texte : boutons, cases à cocher et boutons radio.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre *objet* est un nom d'objet (une chaîne). Si vous ne passez pas le paramètre, vous indiquez que le paramètre *objet* est un champ ou une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de champ ou de variable (champ ou variable objet uniquement).

Référence : OBJET FIXER TITRE

OBJET Lire visible

OBJET Lire visible ({*; }objet) → Booléen

Paramètres	Type		Description
*	*	→	Si spécifié, objet est un nom d'objet (chaîne) Si omis, objet est une variable ou un champ
objet	Objet de formulaire	→	Nom d'objet (si * est spécifié) ou Variable ou champ (si * est omis)
Résultat	Booléen	←	Vrai = objet(s) visible(s), Faux sinon

La nouvelle commande OBJET Lire visible retourne Vrai si l'objet ou le groupe d'objets désigné(s) par *objet* dispose de l'attribut visible et Faux sinon.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre *objet* est un nom d'objet (une chaîne). Si vous ne passez pas le paramètre, vous indiquez que le paramètre *objet* est un champ ou une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de champ ou de variable (champ ou variable objet uniquement).

Référence : OBJET FIXER VISIBLE

Réorganisation du thème

Dans un soucis de standardisation et d'harmonisation, plusieurs commandes existantes du thème "Propriétés des objets" ont été renommées dans 4D v12 (le préfixe "OBJET" a notamment été systématiquement ajouté). Le fonctionnement de ces commandes est inchangé.
En outre, de nouvelles commandes ont été ajoutées afin de proposer un ensemble plus symétrique de commandes de "lecture" et d'"écriture" des propriétés.

Toutes ces modifications sont listées dans le tableau suivant. Les nouvelles commandes sont décrites dans cette section :

Nouveau nom 4D v12	Ancien nom 4D
OBJET DEPLACER	*DEPLACER OBJET*
OBJET FIXER ALIGNEMENT	*FIXER ALIGNEMENT*
OBJET FIXER BARRES DEFILEMENT	*CHOIX VISIBLE BARRES DEFILEMENT*
OBJET FIXER COULEUR	*CHOIX COULEUR*
OBJET FIXER COULEURS RVB	*FIXER COULEURS RVB*
OBJET FIXER FILTRE SAISIE	*CHOIX FILTRE SAISIE*
OBJET FIXER FORMATAGE	*CHOIX FORMATAGE*

OBJET FIXER NOM ENUMERATION	*CHOIX ENUMERATION*
OBJET FIXER POLICE	*CHANGER JEU DE CARACTERES*
OBJET FIXER SAISISSABLE	*CHOIX SAISISSABLE*
OBJET FIXER STYLE POLICE	*CHANGER STYLE*
OBJET FIXER TAILLE POLICE	*CHANGER TAILLE*
OBJET FIXER TITRE	*TITRE BOUTON*
OBJET FIXER VISIBLE	*CHOIX VISIBLE*
OBJET Lire alignement	*Lire alignement*
OBJET Lire formatage	*Lire formatage*
OBJET LIRE RECT	*LIRE RECT OBJET*
OBJET LIRE TAILLE OPTIMALE	*TAILLE OBJET OPTIMALE*
Nouvelles commandes 4D v12	
OBJET DUPLIQUER	
OBJET FIXER ACTIVATION	
OBJET Lire activation	
OBJET LIRE BARRES DEFILEMENT	
OBJET LIRE COULEURS RVB	
OBJET LIRE DEFILEMENT	
OBJET Lire filtre saisie	
OBJET Lire nom enumeration	
OBJET Lire police	
OBJET Lire saisissable	
OBJET Lire style police	
OBJET Lire taille police	
OBJET Lire titre	
OBJET Lire visible	

ACTIVER BOUTON, INACTIVER BOUTON

Les commandes ACTIVER BOUTON et INACTIVER BOUTON sont déclarées obsolètes dans 4D v12. Elles sont conservées pour des raisons de compatibilité uniquement. Leur portée globale, incluant toutes les instances de la variable désignée et non uniquement celles du formulaire courant, ne correspond pas à celle des commandes du thème "Propriétés des objets".

Dans 4D v12, ACTIVER BOUTON et INACTIVER BOUTON sont avantageusement remplacées par les nouvelles commandes OBJET FIXER ACTIVATION et OBJET Lire activation.

Sauvegarde

RESTITUER

RESTITUER *{(cheminArchive{; cheminDossierDest})}*

Paramètres	Type		Description
cheminArchive	*Texte*	→	*Chemin d'accès de l'archive à restituer*
cheminDossierDest	*Texte*	→	*Chemin d'accès du dossier de destination*

La commande RESTITUER accepte désormais deux paramètres facultatifs permettant d'automatiser le processus de restitution et met à jour les variables système *OK* et *document*.

Le paramètre *cheminArchive* vous permet d'indiquer le chemin d'accès du fichier d'archive à restituer. Ce chemin doit être exprimé avec la syntaxe système. Vous pouvez passer un chemin d'accès absolu ou relatif au fichier de structure de la base.
Dans ce cas (si le paramètre *cheminDossierDest* est omis), la boîte de dialogue de restitution standard apparaît avec l'archive pré-sélectionnée, permettant à l'utilisateur de désigner le dossier de destination. Le fonctionnement de la commande est ensuite identique à celui des versions précédentes de 4D : à l'issue de la procédure, une boîte de dialogue d'alerte apparaît et le dossier contenant les éléments restitués est affiché.

Vous pouvez également passer le paramètre *cheminDossierDest* avec le chemin d'accès du dossier de destination des éléments restitués. Ce chemin doit être exprimé avec la syntaxe système. Vous pouvez passer un chemin d'accès absolu ou relatif au fichier de structure de la base. Si vous passez ce paramètre, une boîte de dialogue de restitution préconfigurée apparaît, permettant uniquement à l'utilisateur de lancer ou d'annuler la restitution. A l'issue de la procédure, la fenêtre est simplement refermée sans affichage d'information supplémentaire.

Désormais, la commande RESTITUER modifie la valeur des variables *OK* et *Document* : si la restitution s'est déroulée correctement, *OK* prend la valeur 1 et *Document* contient le chemin du dossier de restitution.
Si l'utilisateur a annulé la boîte de dialogue de restitution, interrompu la restitution ou si une erreur s'est produite, *OK* prend la valeur 0 et *Document* contient une chaîne vide. Vous pouvez intercepter l'erreur à l'aide d'une méthode installée via la commande APPELER SUR ERREUR.

Serveur Web

TRAITER BALISES HTML

La commande TRAITER BALISES HTML a été optimisée dans 4D v12. Les traitements effectués par cette commande sont désormais totalement découplés des paramétrages du serveur Web. En particulier, lorsque vous utilisez des paramètres de type BLOB, la commande ne tient plus compte du jeu de caractères défini pour le serveur Web. Par soucis de compatibilité, elle considère que le jeu de caractères utilisé pour les BLOBs est MacRoman.
Pour plus d'efficacité, il est désormais fortement conseillé d'utiliser des paramètres de type Texte. Dans ce cas, les traitements s'effectuent automatiquement en Unicode.

SQL

Trois nouvelles commandes SQL permettent d'exporter les données de la base 4D et d'exécuter des scripts SQL.

A noter que les nouveautés relatives au moteur SQL de 4D sont décrites dans le chapitre 6, "SQL", page 297.

SQL EXPORTER BASE

SQL EXPORTER BASE(cheminDossier{; nbFichiers{; tailleLimiteFichier}})

Paramètres	Type		Description
cheminDossier	Chaîne	→	Chemin d'accès du dossier d'export ou "" pour afficher une boîte de dialogue de sélection de dossier
nbFichiers	Numérique	→	Nombre maximum de fichiers par dossier
tailleLimiteFichiers	Numérique	→	Valeur de limite de taille des fichiers d'export (en Ko)

La nouvelle commande SQL EXPORTER BASE exporte au format SQL tous les enregistrements de toutes les tables de la base ouverte. En SQL, cette opération d'exportation globale est appelée "Dump".

Note Cette commande ne peut pas être utilisée avec une connexion externe ouverte directement ou via ODBC.

La commande génère un fichier texte contenant les instructions SQL nécessaires à l'importation des données dans une autre base. Ce fichier peut notamment être utilisé directement par la nouvelle commande SQL EXECUTER SCRIPT afin d'importer les données dans une autre base 4D.

Les fichiers exportés seront placés dans un dossier nommé "SQLExport" créé dans le dossier de destination désigné par le paramètre *cheminDossier*. A noter que si un dossier "SQLExport" existe déjà à l'emplacement défini, la commande le remplace sans qu'aucun message d'alerte n'apparaisse.
Si vous passez une chaîne vide dans ce paramètre, 4D affiche une boîte de dialogue standard permettant de à l'utilisateur de désigner le dossier de destination. Par défaut, la boîte de dialogue affiche le dossier courant de l'utilisateur ayant ouvert la session ("Mes Documents" sous Windows ou "Documents" sous Mac OS).

Pour chaque table exportée, la commande effectue les actions suivantes :

- un sous-dossier du nom de la table est créé dans le dossier de destination.
- un fichier texte nommé "Export.sql" est créé dans le sous-dossier. Ce fichier est encodé en UTF-8 avec BOM (marque d'ordre des octets). Il contient des ordres SQL INSERT correspondant aux données exportées.
  Les valeurs des champs sont séparées par des caractères deux-points. Il peut y avoir moins de valeurs que de champs dans la table. Dans ce cas, les champs restants seront considérés NULL.
- si la table contient des champs BLOB, image ou texte (textes à stockage externe c'est-à-dire stockés en-dehors des enregistrements), un sous-dossier supplémentaire nommé "BLOBS" est créé à côté du fichier "Export.sql". A l'intérieur de ce sous-dossier sont créés autant de sous-dossiers que nécessaire nommés "BlobsN". Ces sous-dossiers stockeront sous forme de fichiers séparés le contenu de tous les champs BLOB, image et texte externe des enregistrements de la table. Les fichiers BLOB sont nommés "BlobNNNNN.BLOB" (où NNNNN est un nombre unique généré par l'application). Les fichiers image sont nommés PICTNNNNN.ZZZZ (où NNNNN est un nombre unique généré par l'application et ZZZZ est l'extension).

Lorsque c'est possible, les images sont exportées dans leur format natif d'origine avec l'extension correspondant au type d'image (.jpg, .png, etc.). Si l'export au format natif n'est pas possible, les images sont exportées dans le format 4D interne avec l'extension .4PCT.

Si vous passez le paramètre *nbFichiers*, la commande créera autant de sous-dossiers "BlobsN" que nécessaire afin que chacun d'eux ne contienne pas plus de *nbFichiers* fichiers BLOB, image ou textes externes. Par défaut, si le paramètre *nbFichiers* est omis, la commande limite le nombre de fichiers à 200. Si vous passez 0, chaque sous-dossier contiendra au plus un seul fichier.

Le paramètre *tailleLimiteFichiers* vous permet de définir une limite de taille (en octets) pour chaque fichier "Export.sql" créé sur le disque. Lorsque la taille du fichier d'export en cours de création atteint la valeur définie dans *tailleLimiteFichiers*, 4D stoppe l'écriture des enregistrements, referme le fichier et en crée un nouveau nommé "ExportN.sql" (où N représente le numéro de séquence) à côté du précédent. A noter qu'avec ce mécanisme, la taille effective des fichiers "ExportN.sql" dépasse la valeur définie dans *tailleLimiteFichiers* car le fichier n'est refermé qu'à l'issue de l'écriture complète de l'enregistrement en cours d'exportation. La valeur minimale acceptée est de 100 Ko et la valeur maximale est de 10 Mo.

Dans le fichier d'export, il peut y avoir moins de valeurs que de champs dans la table. Dans ce cas, les champs vides seront considérés comme NULL. Vous pouvez également passer la valeur NULL dans un champ.

Si l'export s'est déroulé correctement, la variable OK prend la valeur 1. Dans le cas contraire, elle prend la valeur 0.

Référence : SQL EXPORTER SELECTION

SQL EXPORTER SELECTION

SQL EXPORTER SELECTION(laTable; cheminDossier{; nbFichiers{; taille-MaxFichier}})

Paramètres	Type		Description
laTable	Table	→	Table de laquelle exporter la sélection
cheminDossier	Chaîne	→	Chemin d'accès du dossier d'export ou "" pour afficher une boîte de dialogue de sélection de dossier
nbFichiers	Numérique	→	Nombre maximum de fichiers par dossier
tailleLimiteFichiers	Numérique	→	Valeur de limite de taille des fichiers d'export (en Ko)

La nouvelle commande SQL EXPORTER SELECTION exporte au format SQL les enregistrements de la sélection courante de la table 4D désignée par le paramètre *laTable*.

Note Cette commande ne peut pas être utilisée avec une connexion externe ouverte directement ou via ODBC.

Cette commande est quasiment identique à la commande SQL EXPORTER BASE. La principale différence entre ces deux commandes réside dans le fait que SQL EXPORTER SELECTION exporte uniquement la sélection courante de *laTable* alors que SQL EXPORTER BASE exporte la totalité de la base. De même, à la différence de SQL EXPORTER BASE, la commande SQL EXPORTER SELECTION ne fonctionne pas avec les bases SQL externes (cf. paragraphe "Utiliser des bases externes", page 309). Elle ne peut être utilisée qu'avec la base principale.

Reportez-vous à la description de la commande SQL EXPORTER BASE pour le détail du fonctionnement et des paramètres de ces commandes.

Si la sélection courante est vide, la commande ne fait rien. A noter que dans ce cas, le dossier de destination n'est pas vidé.
Si l'export s'est déroulé correctement, la variable OK prend la valeur 1. Dans le cas contraire, elle prend la valeur 0.

Référence : SQL EXPORTER BASE

SQL EXECUTER SCRIPT

SQL EXECUTER SCRIPT(cheminScript; actionErreur{; nomAttrib1; valAttrib1;...; nomAttribN; valAttribN})

Paramètres	Type		Description
cheminScript	Chaîne	→	Chemin d'accès complet du fichier contenant le script SQL à exécuter
actionErreur	Entier long	→	Action à effectuer en cas d'erreur durant l'exécution du script
nomAttrib1...N	Chaîne	→	Nom d'attribut à utiliser
valAttrib1...N	Chaîne	→	Valeur de l'attribut

La nouvelle commande SQL EXECUTER SCRIPT vous permet d'exécuter une suite d'instructions SQL placées dans le fichier de script désigné par *cheminScript*. SQL EXECUTER SCRIPT ne peut être exécutée que sur un poste local (4D local ou procédure stockée sur 4D Server).

Note Cette commande ne peut pas être utilisée avec une connexion externe ouverte directement ou via ODBC.

Passez dans le paramètre *cheminScript* le chemin d'accès complet du fichier texte contenant les instructions SQL à exécuter. Le chemin d'accès doit être exprimé à l'aide de syntaxe du système courant. Si vous passez une chaîne vide ("") dans *cheminScript*, une boîte de dialogue standard d'ouverture de documents s'affiche, permettant à l'utilisateur de sélectionner le fichier de script à exécuter.

Note Les nouvelles commandes SQL EXPORTER BASE et SQL EXPORTER SELECTION génèrent automatiquement ce fichier de script.

Le paramètre *actionErreur* vous permet de paramétrer le fonctionnement de la commande lorsqu'elle rencontre une erreur au cours de l'exécution du script. Vous pouvez passer l'une des trois constantes ci-dessous, placées dans le thème "SQL" :

Constante	Type	Valeur
SQL Abandonner si erreur	Entier long	1
SQL Confirmer si erreur	Entier long	2
SQL Continuer si erreur	Entier long	3

- SQL Abandonner si erreur : en cas d'erreur, 4D stoppe immédiatement l'exécution du script.
- SQL Confirmer si erreur : en cas d'erreur, 4D affiche une boîte de dialogue détaillant l'erreur et permettant à l'utilisateur d'interrompre ou de poursuivre l'exécution du script.

- SQL Continuer si erreur : en cas d'erreur, 4D l'ignore et poursuit l'exécution du script.

Les paramètres *nomAttrib* et *valAttrib* doivent être passés par paires. Ces paramètres sont destinés à permettre de définir des attributs spécifiques pour l'exécution du script. Dans la version actuelle de 4D, un seul attribut peut être passé dans *nomAttrib*, disponible via la constante suivante, placée dans le thème "SQL" :

Constante	Type	Valeur
SQL Utiliser les droits d'accès	Chaîne	SQL_Use_Access_Rights

- SQL Utiliser les droits d'accès : permet de restreindre les droits d'accès à appliquer lors de l'exécution des commandes SQL du script. Lorsque vous utilisez cet attribut, vous devez passer 0 ou 1 dans *valAttrib* :
  - *valAttrib* = 1 : 4D utilise les droits d'accès de l'utilisateur 4D courant.
  - *valAttrib* = 0 (ou attribut non défini) : 4D ne restreint pas les accès, les droits du Super_Utilisateur sont utilisés.

Si le fichier d'enregistrement des requêtes de 4D est activé (via les sélecteurs 28 ou 45 de la commande FIXER PARAMETRE BASE), chaque commande SQL exécutée génèrera une entrée avec les informations suivantes :

- Type de commande SQL
- Nombre d'enregistrements affectés par la commande
- Durée d'exécution de la commande
- Pour chaque erreur rencontrée :
  - le code d'erreur
  - le texte de l'erreur s'il est disponible

Si le script est correctement exécuté (aucune erreur rencontrée), la variable système OK prend la valeur 1.
En cas d'erreur, la variable système OK prend ou non la valeur 0 en fonction du paramètre *actionErreur* :

- Si *actionErreur* vaut "SQL Abandonner si erreur" (valeur 1), OK prend la valeur 0.
- Si *actionErreur* vaut "SQL Confirmer si erreur" (valeur 2), la variable OK prend la valeur 0 si l'utilisateur choisit de stopper l'opération et 1 s'il choisit de la poursuivre.
- Si *actionErreur* vaut "SQL Continuer si erreur" (valeur 3), la variable OK vaut toujours 1.

Note Si vous utilisez cette commande pour exécuter des actions consommatrices de mémoire telles que l'importation massive de données, vous pouvez envisager de faire appel à la nouvelle commande SQL ALTER DATABASE afin de désactiver temporairement des options SQL.

SVG

Le nouveau thème "SVG" regroupe toutes les commandes SVG de 4D v12. Dans les versions précédentes de 4D, ces commandes étaient placées dans le thème "XML Utilitaires".

Quatre nouvelles commandes SVG font leur apparition dans 4D v12 : SVG FIXER ATTRIBUT, SVG LIRE ATTRIBUT, SVG Chercher ID elements par rect, SVG MONTRER ELEMENT.

Note sur le moteur de rendu SVG

Le moteur de rendu SVG a été mis à jour dans 4D v12. Outre les nouvelles commandes décrites ci-dessous, de nombreux nouveaux éléments et attributs d'éléments sont désormais pris en charge, tels que les "pattern" ou le "CoreText". Le **composant SVG** de 4D tire parti de ces nouveautés.

SVG FIXER ATTRIBUT

SVG FIXER ATTRIBUT({*; }objetImage; ID_element; nomAttribut; valeurAttribut{; nomAttribut2; valeurAttribut2; ...; nomAttributN; valeurAttributN})

Paramètres	Type		Description
*	*	→	Si spécifié, objetImage est un nom d'objet (chaîne) Si omis, objetImage est une variable ou un champ
objetImage	Image	→	Nom d'objet (si * spécifié) ou Variable ou champ (si * omis)
ID_element	Chaîne	→	ID de l'élément dont un ou plusieurs attribut(s) sont à définir
nomAttribut	Chaîne	→	Attribut à définir
valeurAttribut	Chaîne | Valeur	→	Nouvelle valeur d'attribut

La nouvelle commande SVG FIXER ATTRIBUT permet de modifier la valeur d'un attribut existant dans l'arbre de rendu SVG d'une image affichée.

Les modifications effectuées par cette commande s'appliquent à la représentation de l'image, elle ne sont pas stockées et sont perdues lorsque l'image est effacée par programmation ou lorsque le formulaire est fermé.

Si vous passez le paramètre optionnel *****, vous indiquez que le paramètre *objetImage* est un nom d'objet (une chaîne). Dans ce cas, la commande s'applique aux paramètres de l'image de rendu attachée à l'objet (à noter que les paramètres et donc l'image de rendu de l'objet ne sont créés que si la commande SVG FIXER ATTRIBUT est appelée au moins une fois).
Si vous ne passez pas le paramètre *****, vous indiquez que le paramètre *objetImage* est une variable. Vous ne passez alors pas une chaîne mais une référence de variable (variable objet uniquement). Dans ce cas, la commande s'applique aux images de rendu de tous les objets qui utilisent la variable (mais pas à l'image de rendu initiale).

Le paramètre *ID_element* permet de définir l'ID (attribut "id" ou "xml:id") de l'élément dont vous souhaitez modifier un ou plusieurs attribut(s).

Passez dans les paramètres *nomAttribut* et *valeurAttribut* respectivement l'attribut à écrire et sa valeur (sous forme de variables, champs ou valeurs littérales). Vous pouvez passer autant de couples attribut/valeur que vous voulez.

La commande SVG FIXER ATTRIBUT vous permet de modifier (mais pas d'ajouter ou de supprimer) la plupart des attributs SVG, comme par exemple 'fill', 'opacity', 'font-family', etc. Pour une définition complète des attributs SVG, reportez-vous aux documents de référence disponibles sur Internet, par exemple http://www.w3.org/TR/SVG11/attindex.html. La mise à jour de l'image de rendu est immédiate, les modifications sont reportées en cascade sur les éléments enfants pour les styles héritables.

A noter que pour des raisons techniques, les attributs de certains éléments ainsi que certains attributs ne sont pas modifiables. Le tableau suivant liste les éléments modifiables, les éléments non modifiables ainsi que les attributs non modifiables :

Eléments dont les attributs sont modifiables	
svg	Restrictions : - "width" et "height" ne sont pas modifiables (1) - "viewBox" n'est modifiable que si "width" et "height" sont définis dans le document d'origine
g	
defs	
use	
filter	Restriction : les éléments enfants fe_xxx ne sont pas modifiables
circle	
ellipse	
line	
polyline	
polygon	
path	
rect	
text	L'attribut spécifique "4d-text" vous permet de modifier le texte d'un élément "text", "tspan" ou "textArea" (cf. exemple)
tspan	
textArea	
Image	
Eléments dont les attributs ne sont pas modifiables	
linearGradient	Cet ensemble désigne tous les éléments référençables ou contenus dans un élément référençable. Cela signifie qu'il n'est pas possible par exemple de redéfinir les attributs d'un gradient (mais il est possible de changer le gradient utilisé). De même, pour changer un marqueur de couleur noire en marqueur rouge, il faudra définir deux marqueurs dans le document SVG (un noir et un rouge) et sélectionner l'un ou l'autre. Il n'est pas possible non plus par exemple de modifier la couleur d'un rectangle s'il a pour parent un élément symbol ou marker
radialGradient	
Stop	
solidColor	
marker	
symbol	
clipPath	
filter et les éléments commençant par fe	
style	
pattern	
Attributs non modifiables	
id ou xml:id	
lang ou xml:lang	

class ou xml:class	
width	Concerne les attributs de l'élément 'svg' uniquement (1)
height	

(1) Ces attributs ne peuvent être modifiés car ils définissent et structurent l'image résultante. Les attributs width et height de l'élément svg servent à définir les dimensions initiales de l'image dans 4D et ces dimensions doivent rester constantes après la création de l'image (il est toutefois possible de modifier les dimensions de l'image résultante avec la commande TRANSFORMER IMAGE de 4D).

Si vous tentez de modifier un attribut d'un élément non pris en charge ou l'un de ses enfants, la commande ne fait rien et aucune erreur n'est générée.

- Modification du contenu d'un élément de type texte :

SVG FIXER ATTRIBUT (*;nom_objet_image;text_element_ID;"4d-text";"Ceci est un texte")

Note Il n'y a pas de *namespace* pour que l'attribut puisse être utilisé dans une feuille de style CSS sans risque de conflit.

SVG LIRE ATTRIBUT

SVG LIRE ATTRIBUT({*; }objetImage; ID_element; nomAttribut; valeurAttribut)

Paramètres	Type		Description
*	*	→	Si spécifié, objetImage est un nom d'objet (chaîne) Si omis, objetImage est une variable ou un champ
objetImage	Image	→	Nom d'objet (si * spécifié) ou Variable ou champ (si * omis)
ID_element	Chaîne	→	ID de l'élément dont vous souhaitez connaître une valeur d'attribut
nomAttribut	Chaîne	→	Attribut dont vous souhaitez connaître la valeur
valeurAttribut	Chaîne | Valeur	←	Valeur d'attribut courante

La nouvelle commande SVG LIRE ATTRIBUT permet de lire la valeur courante de l'attribut *nomAttribut* dans un objet ou une image SVG.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre *objetImage* est un nom d'objet (une chaîne). Dans ce cas, la commande retourne la valeur de l'attribut pour l'image de rendu attachée à l'objet. Cette valeur peut avoir été modifiée par SVG FIXER ATTRIBUT par exemple.
Si vous ne passez pas le paramètre *, vous indiquez que le paramètre *objetImage* est une variable. Vous ne passez alors pas une chaîne mais une référence de variable (variable objet uniquement). Dans ce cas, la commande retourne la valeur de l'attribut pour l'image de rendu initiale (correspondant à la source de données de la variable).

Note Ce principe s'applique également à la commande existante SVG Chercher ID element par coordonnees.

Le paramètre *ID_element* permet de définir l'ID (attribut "id" ou "xml:id") de l'élément dont vous souhaitez lire la valeur d'attribut.

Pour plus d'informations sur les attributs SVG, reportez-vous à la description de la commande SVG FIXER ATTRIBUT. Voici la liste des attributs 4D réservés et dédiés à l'animation :

Attributs	Accès	Commentaire
4D-text	lecture/ écriture	Remplace/lit le contenu du noeud de texte. Utilisable avec les éléments 'text' 'tspan' et 'textArea'
4D-bringToFront	écriture	Si 'true', déplacer le noeud devant les noeuds frères. Utilisable uniquement avec la commande SVG FIXER ATTRIBUT
4D-isOfClass-{IDENT [[S|COMMA] IDENT]*}	lecture	Si l'attribut de la classe héritée du noeud contient tous les noms de classes, retourne 'true' sinon retourne 'false'. Retourne par exemple true pour "4D-isOfClass-land" si la classe héritée du noeud est "land department01")

SVG Chercher ID elements par rect

SVG Chercher ID elements par rect ({*; }objetImage; x; y; largeur; hauteur; tabIds) → Booléen

Paramètres	Type		Description
*	*	→	Si spécifié, objetImage est un nom d'objet (chaîne) Si omis, objetImage est une variable ou un champ
objetImage	Image	→	Nom d'objet (si * spécifié) ou Champ ou variable (si * omis)
x	Entier long	→	Coordonnée horizontale du coin haut gauche du rectangle de sélection
y	Entier long	→	Coordonnée verticale du coin haut gauche du rectangle de sélection
largeur	Entier long	→	Largeur du rectangle de sélection
hauteur	Entier long	→	Hauteur du rectangle de sélection
tabIds	Tab chaîne	←	IDs des éléments dont le rectangle englobant est en intersection avec le rectangle de sélection
Résultat	Booléen	←	Vrai = au moins un élément est trouvé

La nouvelle commande SVG Chercher ID elements par rect remplit le tableau Texte ou Alpha *tabIds* avec les IDs (attribut "id" ou "xml:id") des éléments XML dont le rectangle englobant est en intersection avec le rectangle de sélection à l'emplacement défini par les paramètre *x* et *y*.

La commande retourne Vrai si au moins un élément est trouvé (c'est-à-dire si le tableau *tabIds* est non vide) et Faux sinon.

Cette commande permet notamment de gérer des interfaces graphiques interactives.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre *objetImage* est un nom d'objet (une chaîne). Si vous ne passez pas le paramètre, vous indiquez que le paramètre *objetImage* est un champ ou une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de champ ou de variable (champ ou variable objet uniquement).

Les coordonnées passées dans les paramètres *x* et *y* doivent être exprimées en pixels relativement à l'angle supérieur gauche de l'image (0,0). Vous pouvez utiliser les valeurs retournées par les variables système *MouseX* et *MouseY*. Ces variables sont mises à jour dans les événements formulaire Sur clic, Sur double clic ainsi que Sur début survol et Sur survol.

Note Dans le système de coordonnées des images, [x;y] définit toujours le même point, quel que soit le format d'affichage de l'image, hormis pour le format "mosaïque".

Tous les ID d'éléments dont le rectangle englobant est inclus dans le rectangle de sélection sont pris en compte, même ceux situés sous d'autres éléments.

SVG MONTRER ELEMENT

SVG MONTRER ELEMENT ({*; }objetImage; id{; marge})

Paramètres	Type		Description
*	*	→	Si spécifié, objetImage est un nom d'objet (chaîne) Si omis, objetImage est une variable ou un champ
objetImage	Image	→	Nom d'objet (si * spécifié) ou Variable ou champ (si * omis)
id	Chaîne	→	Attribut id de l'élément à visualiser
marge	Entier long	→	Marge de visibilité (en pixels par défaut)

La nouvelle commande SVG MONTRER ELEMENT déplace le document SVG *objetImage* de façon à rendre visible l'élement dont l'attribut "id" est désigné par le paramètre *id*.

Si vous passez le paramètre optionnel *, vous indiquez que le paramètre *objetImage* est un nom d'objet (une chaîne). Dans ce cas, la commande s'applique à l'image de rendu attachée à l'objet. Si vous ne passez pas le paramètre *, vous indiquez que le paramètre *objetImage* est une variable. Vous ne passez alors pas une chaîne mais une référence de variable (variable objet uniquement). Dans ce cas, la commande s'applique aux images de rendu de tous les objets qui utilisent la variable (mais pas à l'image de rendu initiale).

La commande déplace le document SVG de manière à ce que la totalité de l'objet, dont les limites sont définies par son rectangle englobant, soit visible. Le paramètre *marge* permet de paramétrer l'amplitude du déplacement en définissant la distance devant séparer l'objet visualisé des bords du document. Autrement dit, le rectangle englobant sera augmenté de *marge* pixels en largeur et en hauteur. Par défaut, la valeur de déplacement est de 4 pixels.

Cette commande n'a d'effet qu'en mode d'affichage "top left" (avec barres de défilement).

XML

Note Le thème "XML" remplace le thème "XML Utilitaires" des versions précédentes de 4D. Il regroupe désormais les commandes XML génériques s'appliquant à la fois aux commandes DOM et SAX, ainsi que les commandes XSLT et SVG.

Les commandes XML intégrées ont été modifiées dans 4D v12 et de nouvelles commandes ont été ajoutées. Les mécanismes de conversion des types de données entre 4D et le XML ont été renforcés.

Note sur l'usage de BLOBs XML dans 4D v12

Les structures XML sont basées sur des données de type texte. Toutefois, les commandes XML de 4D (par exemple DOM Analyser variable XML) acceptent généralement des paramètres de type BLOBs pour la manipulation des données XML, quel que soit leur type d'origine. Cette possibilité était nécessaire dans les versions précédentes de 4D, où la taille des variables de type Texte était limitée à 32 Ko.

Depuis 4D v11, les variables et champs texte peuvent contenir jusqu'à 2 Go de données. Il est désormais fortement déconseillé de stocker des textes dans des BLOBs. L'usage de BLOBs est à réserver au traitement de données binaires. Pour une meilleure conformité aux spécifications du XML, dans 4D v12 les données binaires sont automatiquement encodées en Base64, même si le BLOB contient du texte.

Note Les commandes ENCODER et DECODER (renommées ENCODER BASE64 et DECODER BASE64) peuvent désormais manipuler des paramètres de type Texte. Ces commandes sont utiles dans le cadre de l'encodage de données XML. Ces commandes sont incluses dans le thème "Outils" (cf. paragraphe "Outils", page 186).

XML FIXER OPTIONS

XML FIXER OPTIONS(refElément | document; sélecteur; valeur{; sélecteur2; valeur2;...;sélecteurN; valeurN})

Paramètres	Type		Description
refElément | document	Chaîne | DocRef	→	Référence d'élément XML racine ou Référence de document ouvert
sélecteur1...N	Entier	→	Option à définir
valeur1...N	Entier	→	Valeur de l'option

La nouvelle commande XML FIXER OPTIONS permet de modifier la valeur d'un ou plusieurs paramètre(s) XML pour la session courante et l'utilisateur courant.

Cette commande s'applique aux structures XML de type "arbre" (DOM) ou "document" (SAX). Vous pouvez passer en premier paramètre soit une référence d'élément racine (*refElément*), soit une référence de document SAX ouvert (*document*).

Les options définies par cette commande sont utilisées uniquement dans le sens 4D vers XML (elle n'a pas d'effet sur la lecture de valeurs XML dans 4D). Les commandes utilisant ces options sont les suivantes :

- DOM ECRIRE VALEUR ELEMENT XML
- DOM ECRIRE ATTRIBUT XML
- SAX AJOUTER VALEUR ELEMENT XML

Passez dans *sélecteur* l'option à modifier et dans *valeur* la nouvelle valeur de l'option. Vous pouvez passer autant de couples *sélecteur*/*valeur* que vous souhaitez.
Vous devez utiliser les constantes décrites ci-dessous, placées dans le thème "XML".

- *sélecteur* = XML Encodage chaînes (1)
  Ce sélecteur définit la manière dont les chaînes 4D sont converties en valeurs d'éléments. Il ne concerne pas les conversions en attributs pour lesquelles le XML impose l'utilisation de caractères d'échappement.
  - *valeur* = XML Avec échappement (1) (valeur par défaut) : conversion des chaînes 4D en valeurs d'éléments XML avec remplacement des caractères. Les données de type texte sont automatiquement analysées de manière à ce que les caractères interdits (<&>') soient remplacés par des entités XML (&<> '").

- *valeur* = XML Données brutes (2) : les chaînes 4D sont envoyées en tant que données brutes, 4D n'effectue ni encodage ni analyse. Les valeurs 4D seront converties si possible en fragments XML et insérées en tant qu'enfant de l'élément cible. Si une valeur ne peut pas être considérée comme fragment XML, elle est insérée sous forme de donnée brute dans un nouveau noeud CDATA.

- *sélecteur* = XML Encodage dates (2)
  Ce sélecteur définit la manière dont les dates 4D seront converties. Par exemple, le !01/01/2003! dans le fuseau horaire de Paris.
  - *valeur* = XML ISO (1) (valeur par défaut) : utilisation du format xs:datetime sans indication de fuseau horaire. Résultat : "2003-01-01". La partie heure, si elle est présente dans la valeur 4D (via le SQL), est perdue.
  - *valeur* = XML Local (2) : utilisation du format xs:date avec indication de fuseau horaire. Résultat : "2003-01-01 +01:00". La partie heure, si elle est présente dans la valeur 4D (via le SQL), est perdue.
  - *valeur* = XML Datetime local (3) : utilisation du format xs:dateTime (ISO 8601). Indication de fuseau horaire. Ce format permet de conserver la partie heure, si elle est présente dans la valeur 4D (via le SQL). Résultat : "<Date>2003-01-01T00:00:00 +01:00</Date>".
  - *valeur* = XML UTC (4) : utilisation du format xs:date. Résultat : "2003-01-01Z". La partie heure, si elle est présente dans la valeur 4D (via le SQL), est perdue.
  - *valeur* = XML Datetime UTC (5) : utilisation du format xs:dateTime (ISO 8601). Ce format permet de conserver la partie heure, si elle est présente dans la valeur 4D (via le SQL). Résultat : "<Date>2003-01-01T00:00:00Z</Date>".

Note Les valeurs XML Local et XML Datetime local ne fournissent pas de dates exprimées en UTC (Temps Universel Coordonné), elles sont converties sans modification mais indiquent le décalage horaire. Ces formats sont utiles dans le cas de conversions successives et réciproques (*round tripping*).
Les valeurs XML UTC et XML Datetime UTC sont équivalentes aux précédentes du point de vue du formatage, mais sont exprimées en UTC. Ces formats sont à privilégier pour assurer l'interopérabilité. Les valeurs ne sont pas modifiées.

- *sélecteur* = XML Encodage heures (3)
  Ce sélecteur définit la manière dont les heures 4D seront converties. Par exemple, ?02/00/46? (heure de Paris). L'encodage diffère suivant que vous souhaitez exprimer une heure ou une durée.

  Pour les valeurs de type heure :

  - *valeur* = XML Datetime UTC (5) : heure exprimée en UTC (Temps Universel Coordonné). A noter que la conversion en UTC est automatique. Résultat : "<Duree>0000-00-00T01:00:46Z</Duree>".
  - *valeur* = XML Datetime local (3) : heure exprimée avec le décalage horaire de la machine du moteur de 4D. Résultat : "<Duree>0000-00-00T02:00:46+01:00</Duree>".
  - *valeur* = XML Datetime local absolu (1) (valeur par défaut) : heure exprimée sans indication de fuseau horaire. Pas de modification de la valeur. Résultat : "<Duree>0000-00-00T02:00:46</Duree>".

  Pour les valeurs de type durée :

  - *valeur* = XML Secondes (4) : nombre de secondes depuis minuit, pas de modification de la valeur puisqu'elle exprime une durée. Résultat : "<Duree>7246</Duree>".
  - *valeur* = XML Durée (2) : durée exprimée de manière conforme au XML Schema Part 2: Datatypes Second Edition. Pas de modification de la valeur puisqu'elle exprime une durée. Résultat : "<Duree>PT02H00M46S</Duree>".

- *sélecteur* = XML Encodage binaire (5)
  Ce sélecteur définit la manière dont les données binaires seront converties.
  - XML Base64 (1) (valeur par défaut) : les données binaires sont simplement converties en base64.
  - XML Data URI scheme (2) : les données binaires sont converties en base64 et l'en-tête "data:;base64" est ajouté. Ce format permet principalement à un navigateur de décoder automatiquement une image, et est également requis pour l'insertion d'images SVG. Pour plus d'informations, voir http://en.wikipedia.org/wiki/Data_URI_scheme.

- *sélecteur* = XML Encodage images (6)
  Ce sélecteur définit la manière dont les images doivent converties (avant l'encodage en base64).
  - XML Convertir en PNG (1) (valeur par défaut) : les images sont converties en PNG avant d'être encodées en base64.

- XML Codec natif (2) : les images sont converties dans leur premier CODEC natif de stockage avant d'être encodées en base64. Vous devez utiliser ces options pour encoder des images SVG (voir exemple 1).

- *sélecteur* = XML Indentation (4)
  Ce sélecteur définit l'indentation du document XML.
  - XML Avec indentation (1) (valeur par défaut) : le document est indenté.
  - XML Sans indentation (2) : le document n'est pas indenté, son contenu est placé sur une seule ligne.

▸ Exemple 1 : Insertion d'une image SVG

XML FIXER OPTIONS ($refImageElem;XML Encodage binaire;
XML Data URI scheme)
XML FIXER OPTIONS ($refImageElem;XML Encodage images ;
XML Codec natif)
DOM ECRIRE ATTRIBUT XML ($refImageElem;"xlink:href";VarImage)

Référence : XML LIRE OPTIONS

XML LIRE OPTIONS

XML LIRE OPTIONS(refElément | document; sélecteur; valeur{; sélecteur2; valeur2;...;sélecteurN; valeurN})

Paramètres	Type		Description
refElément | document	Chaîne | DocRef	→	Référence d'élément XML racine ou Référence de document ouvert
sélecteur1...N	Entier	→	Option à lire
valeur1...N	Entier	←	Valeur courante de l'option

La nouvelle commande XML LIRE OPTIONS permet de lire la valeur courante d'un ou plusieurs paramètre(s) XML pour la session courante et l'utilisateur courant.

Passez dans *sélecteur* une constante du thème "XML" indiquant l'option à connaître. La valeur courante de l'option est retournée dans le paramètre *valeur*.

Pour plus d'informations sur les options à passer dans le paramètre *sélecteur* et les valeurs retournées, reportez-vous à la description de la commande XML FIXER OPTIONS.

Référence : XML FIXER OPTIONS

XML DECODER

XML DECODER (valeurXML; objet4D)

Paramètres	Type		Description
valeurXML	Texte	→	Valeur de type texte provenant d'une structure XML
objet4D	Champ | Variable	←	Variable ou champ 4D devant recevoir la valeur XML convertie

La commande XML DECODER convertit une valeur stockée en tant que chaîne XML en une valeur 4D typée. La conversion est effectuée automatiquement en fonction des règles suivantes :

Valeur	Exemples	Conversion sur système français
numérique	<Prix>8,5</Prix> <Prix>8.5</Prix>	Réel : 8,5
booléenne	<Double>1</Double> <Double>0</Double> ou <Double>vrai</Double> <Double>faux</Double>	Booléen : Vrai/Faux
BLOB		Décodage base64
Images		Décodage base64 + commande BLOB vers image
Dates	2009-10-25T01:03:20+01:00	Suppression de la partie heure et du fuseau horaire : !25/10/2009!
Heures	2009-10-25T01:03:20+01:00	Suppression de la partie date et du fuseau horaire : ?01:03:20?

- Importation de données depuis un document XML dans lequel les valeurs sont stockées en tant qu'attributs.

  Exemple de document XML :

  <CD Date="2003-01-01T00:00:00Z" Description="Ce double CD réédité par EMI en 1995 réunit 4 Stabat mater. Celui de Rossini interprété par l'orchestre Symphonique de Berlin dirigé par Karl Forster. Il est suivi de l'œuvre de Verdi, Philharmonia Orchestra dirigé par Carlo Maria Giulini. Sur le deuxième CD, on trouve Francis Poulenc interprété par Régine Crespin. Cette compilation se termine avec une version moins connue, celle du polonais Karol Szymanowski. Orchetsre Symphonique de la Radio Nationale polonaise dirigée par Antoni Wit" Double="true" Duree="7246"

Genre="Musique sacrée" ID_CD="5" Interprete="Divers" Prix="8.5" Titre="4 Stabat mater"/>

```
Repeter
  MyEvent:=SAX Lire noeud XML(DocRef)

  Au cas ou
  : (MyEvent=Début élément XML)
    TABLEAU TEXTE(tAttrNames;0)
    TABLEAU TEXTE(tAttrValues;0)
    SAX LIRE ELEMENT XML(DocRef;vName;vPrefix;tAttrNames;tAttrValues)
    Si (vName="CD")
      CREER ENREGISTREMENT([CD])
      Boucle ($i;1;Taille tableau(tAttrNames))
        $attrName:=tAttrNames{$i}
        Au cas ou
        : ($attrName="ID_CD")
          XML DECODER(tAttrValues{$i};[CD]ID_CD)
        : ($attrName="Titre")
          [CD]Œuvre:=tAttrValues{$i}
        : ($attrName="Prix")
          XML DECODER(tAttrValues{$i};[CD]Prix)
        : ($attrName="Date")
          XML DECODER(tAttrValues{$i};[CD]Date saisie)
        : ($attrName="Duree")
          XML DECODER(tAttrValues{$i};[CD]Durée_totale)
        : ($attrName="Double")
          XML DECODER(tAttrValues{$i};[CD]CD_Double)
        Fin de cas
      Fin de boucle
    Fin de si
  ...
  Fin de cas
Jusque (MyEvent=Fin document XML)
```

XML DOM

Plusieurs nouvelles commandes ont été ajoutées dans le thème XML DOM : DOM Creer element XML tableaux, DOM SUPPRIMER ATTRIBUT XML, DOM Inserer element XML, DOM Ajouter element xml, DOM Ajouter noeud enfant XML, DOM LIRE NOEUDS ENFANTS XML, DOM Lire ref document xml. De plus, du fait de l'implémentation des commandes XML FIXER OPTIONS et XML LIRE OPTIONS, la commande existante DOM ECRIRE OPTIONS XML a été renommée DOM FIXER DECLARATION XML et son fonctionnement a été adapté.

DOM Creer element XML tableaux

DOM Creer element XML tableaux (refElément; xChemin{; tabNomsAttributs; tabValeursAttributs}{; tabNomsAttributs2; tabValeursAttributs2; ...; tabNomsAttributsN; tabValeursAttributsN}) → Chaîne

Paramètres	Type		Description
refElément	Chaîne	→	Référence d'élément XML racine
xChemin	Texte	→	Chemin XPath de l'élément XML à créer
tabNomsAttributs	Tab	→	Tableau de noms d'attributs
tabValeursAttributs	Tab	→	Tableau de valeurs d'attributs
Résultat	Chaîne	←	Référence de l'élément XML créé

La nouvelle commande DOM Creer element XML tableaux permet d'ajouter un nouvel élément dans l'élément XML *refElément* ainsi que, facultativement, des attributs et leurs valeurs sous forme de tableaux.

Hormis la prise en charge de tableaux (cf. ci-dessous), cette commande est identique à DOM Creer element XML. Reportez-vous à la description de cette commande pour le détail de son fonctionnement.

Facultativement, la commande DOM Creer element XML tableaux permet de passer plusieurs couples d'attributs et de valeurs d'attributs sous forme de tableaux dans les paramètres *tabNomsAttributs* et *tabValeursAttributs*. Vous pouvez passer dans *tabValeursAttributs* des tableaux de type texte, date, numérique et image. 4D effectue automatiquement les conversions nécessaires, vous pouvez modifier ces conversions à l'aide de la nouvelle commande XML FIXER OPTIONS.

Les tableaux doivent avoir été créés au préalable et fonctionner par paires. Vous pouvez passer autant de couples de tableaux et autant d'éléments dans chaque couple que vous voulez.

- Nous souhaitons créer l'élément suivant :

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<RootElement>
  <Elem1>
    <Elem2>
      <Elem3 Font="Verdana" Size="10" Style="Bold"></Elem3>
    </Elem2>
  </Elem1>
</RootElement>
```

Pour cela, il suffit d'écrire :

```
TABLEAU TEXTE(tNomsAtt;3)
TABLEAU TEXTE(tValeursAtt;3)
tNomsAtt{1}:="Font"
tValeursAtt{1}:="Verdana"
tNomsAtt{2}:="Style"
tValeursAtt{2}:="10"
tNomsAtt{3}:="Style"
tValeursAtt{3}:="Bold"
vRefRacine:=DOM Creer ref XML("RootElement")
vxPath:="/RootElement/Elem1/Elem2/Elem3"
vRefElement:=DOM Creer element XML tableaux(vRefRacine;
                                            vxPath;tNomsAtt;tValeursAtt)
```

Référence : DOM Creer element XML

DOM SUPPRIMER ATTRIBUT XML

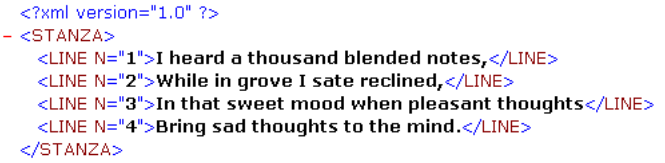
DOM SUPPRIMER ATTRIBUT XML(refElément; nomAttribut)

Paramètres	Type		Description
refElément	Chaîne	→	Référence d'élément XML
nomAttribut	Chaîne	→	Attribut à supprimer

La nouvelle commande DOM SUPPRIMER ATTRIBUT XML supprime, s'il existe, l'attribut désigné par *nomAttribut* de l'élément XML dont la référence est passée dans le paramètre *refElément*.

Si l'attribut a été correctement supprimé, la variable système OK prend la valeur 1. Si aucun attribut nommé *nomAttribut* n'existe dans *refElément*, une erreur est retournée et la variable système OK prend la valeur 0.

- Soit la structure suivante :



Le code suivant permet de supprimer le premier attribut "N=1" :

```
C_BLOB(maVarBlob)
C_ALPHA(16;$ref_XML_Parent;$ref_XML_Enfant)
C_ENTIER LONG($NumLigne)

$ref_XML_Parent:=DOM Analyser variable XML(maVarBlob)
$ref_XML_Enfant:=DOM Lire premier element XML
                                        enfant($ref_XML_Parent)
DOM SUPPRIMER ATTRIBUT XML($ref_XML_Enfant;"N")
```

Référence : DOM LIRE ATTRIBUT XML PAR INDEX, DOM LIRE ATTRIBUT XML PAR NOM

DOM Inserer element XML

DOM Inserer element XML (refElémentCible; refElémentSource; indexEnfant) → Chaîne

Paramètres	Type		Description
refElémentCible	Chaîne	→	Référence de l'élément XML parent
refElémentSource	Chaîne	→	Référence de l'élément XML à insérer
indexEnfant	Entier long	→	Index de l'enfant de l'élément cible avant lequel le nouvel élement doit être inséré
Résultat	Chaîne	←	Référence du nouvel élément XML

La nouvelle commande DOM Inserer element XML permet d'insérer un nouvel élément XML parmi les enfants de l'élément XML dont la référence est passée dans le paramètre *refElémentCible*.

Passez dans *refElémentSource* l'élément à insérer. Cet élément doit être passé en tant que référence d'un élément XML existant dans un arbre DOM.

Le paramètre *indexEnfant* permet de désigner l'enfant de l'élément parent avant lequel le nouvel élément doit être inséré. Passez un numéro d'index dans ce paramètre. Si la valeur est invalide (par exemple s'il n'existe pas d'élément enfant de cet index), le nouvel élément sera ajouté avant le premier enfant de l'élément parent.

La commande retourne la référence de l'élément XML obtenu.

- Dans la structure suivante, nous souhaitons inverser le premier et le deuxième livre :

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<NewBooks>
 <Book>
      <Title>Services Web Open Source</Title>
      <Author>Collectif</Author>
      <Date>2003</Date>
      <ISBN>2-7440-1507-5</ISBN>
      <Publisher>Wrox</Publisher>
  </Book>
<Book>
      <Title>Construire des services Web XML</Title>
      <Author>Scott Short</Author>
      <Date>2002</Date>
      <ISBN>2-10-006476-2</ISBN>
      <Publisher>Microsoft Press</Publisher>
  </Book>
</NewBooks>
```

Pour cela, il suffit d'exécuter le code suivant :

```
C_TEXTE($refRoot)
$refRoot:=DOM Analyser source XML("") `sélection du document XML
Si (OK=1)
  C_TEXTE($newStruct)
  $newStruct:=DOM Creer ref XML ("NewBooks")

  $refBook:=DOM Chercher element XML($refRoot;"/BookCatalo-
gue/Book[1]")
  $refNewElement:=DOM Ajouter element XML($newStruct;$refBook)

  $refBook:=DOM Chercher element XML ($refRoot;"/BookCatalo-
gue/Book[2]")
  C_TEXTE($refNewElement)
  $refNewElement:=DOM INSERER ELEMENT XML($newStruct;$refBook;1)
```

```
   DOM FERMER XML($newStruct)
   DOM FERMER XML($refRoot)
Fin de si
```

Référence : DOM Ajouter element xml

DOM Ajouter element xml

DOM Ajouter element XML (refElémentCible; refElémentSource) → Chaîne

Paramètres	Type		Description
refElémentCible	Chaîne	→	Référence de l'élément XML parent
refElémentSource	Chaîne	→	Référence de l'élément XML à ajouter
Résultat	Chaîne	←	Référence du nouvel élément XML

La nouvelle commande DOM Ajouter element xml permet d'ajouter un nouvel élément XML aux enfants de l'élément XML dont la référence est passée dans le paramètre *refElémentCible*.

Passez dans *refElémentSource* l'élément à ajouter. Cet élément doit être passé en tant que référence d'un élément XML existant dans un arbre DOM. Il est ajouté après le dernier élément enfant existant de *refElémentCible*.

- Voir l'exemple de la commande DOM Inserer element XML.

Référence : DOM Inserer element XML

DOM Ajouter noeud enfant XML

DOM Ajouter noeud enfant XML (refElément; typeEnfant; valeurEnfant) → Chaîne

Paramètres	Type		Description
refElément	Chaîne	→	Référence d'élément XML
typeEnfant	Entier long	→	Type d'enfant à ajouter
valeurEnfant	Chaîne | BLOB	→	Texte ou variable (Texte ou BLOB) dont la valeur doit être insérée en tant que noeud enfant
Résultat	Chaîne	←	Référence de l'élément XML enfant

La nouvelle commande DOM Ajouter noeud enfant XML permet d'ajouter la valeur *valeurEnfant* au noeud XML désigné par *refElément*.

Le type de noeud créé est défini par le paramètre *typeEnfant*. Passez dans ce paramètre l'une des constantes suivantes, placées dans le thème "XML" :

Type de noeud enfant	Constante (valeur)
Texte	XML Donnée (6)
Instruction de traitement	XML Instruction de traitement (3)
Commentaire	XML Commentaire (2)
CDATA	XML CDATA (7)
DOCTYPE	XML DOCTYPE (10)
Elément	XML Elément (11)

Passez dans *valeurEnfant* les données à insérer. Vous pouvez passer une chaîne ou une variable 4D (chaîne ou BLOB). Le contenu de ce paramètre sera toujours converti en texte.

Note Si le paramètre *refElément* désigne le noeud Document (noeud de plus haut niveau), la commande insère un noeud "Doctype" avant tout autre noeud. Il en va de même pour les instructions de traitement et les commentaires, qui sont toujours insérés avant le noeud racine (mais après le noeud Doctype).

- Ajout d'un noeud de type texte :

```
Reference := DOM Creer element XML(refElement;"monElement")
DOM ECRIRE VALEUR ELEMENT XML(Reference  ; "Bonjour")
temp:= DOM Creer element XML (Reference  ; "br")
temp := DOM Ajouter noeud enfant XML (Reference ; XML Donnée; "La")
temp := DOM Creer element XML (Reference ; "br")
temp := DOM Ajouter noeud enfant XML(Reference ; XML Donnée; "France")
```

Résultat :
**<monElement>Bonjour
La
France</monElement>**

- Ajout d'un noeud de type instruction de traitement :

```
$Txt_instruction:="xml-stylesheet type = \"text/xsl\" href=\"style.xsl\""
Reference:= DOM Ajouter noeud enfant XML(refElement;XML Instruction de traitement ; $Txt_instruction )
```

Résultat (inséré avant le premier élément) :
<?xml-stylesheet type="text/xsl" href="style.xsl"?>

- Ajout d'un noeud de type commentaire :

```
Reference := DOM Ajouter noeud enfant XML (refElement;
                                    XML Commentaire; "Hello world")
```

Résultat :
<!--Hello world-->

- Ajout d'un noeud de type CDATA :

```
Reference := DOM Ajouter noeud enfant XML (refElement; XML CDATA;
                                    "12 < 18")
```

Résultat :
<element><![CDATA[12 < 18]]></element>

- Ajout ou remplacement d'un noeud de type déclaration Doctype :

```
Reference :=DOM Ajouter noeud enfant XML(refElement;XML DOCTYPE;
                                    "Books SYSTEM \"Book.DTD\"")
```

Résultat (inséré avant le premier élément) :
<!DOCTYPE Books SYSTEM "Book.DTD">

- Ajout ou remplacement d'un noeud de type Elément.
  - si le paramètre *valeurEnfant* est un fragment XML, il est inséré en tant que noeuds enfants :

```
Reference:=DOM Ajouter noeud enfant XML(refElement;XML Elément;
                    "<child>simon</child><child>eva</child>")
```

Résultat :

```
<parent>
  <child>simon</child>
  <child>eva</child>
</parent>
```

  - sinon, un nouvel élément enfant vide est ajouté :

```
Reference:=DOM Ajouter noeud enfant XML(refElement;XML Elément;
                                    "tbreak")
```

Résultat :

```
<parent>
  <tbreak/>
</parent>
```

Si le contenu de *valeurEnfant* est invalide, une erreur est retournée.

Référence : DOM LIRE NOEUDS ENFANTS XML

DOM LIRE NOEUDS ENFANTS XML

DOM LIRE NOEUDS ENFANTS XML(refElément; tabTypesEnfants; tabRefs-Noeuds)

Paramètres	Type		Description
refElément	Chaîne	→	Référence d'élément XML
tabTypesEnfants	Tab Entier long	←	Types des noeuds enfants
tabRefsNoeuds	Tableau Texte	←	Références ou Valeurs des noeuds enfants

La nouvelle commande DOM LIRE NOEUDS ENFANTS XML retourne les types et les références ou valeurs de tous les noeuds enfants de l'élément XML désigné par *refElément*.

Les types des noeuds enfants sont retournés dans le tableau *tabTypesEnfants*. Vous pouvez comparer les valeurs renvoyées par la commande avec les constantes suivantes du thème "XML" :

Constante	Type	Valeur
XML Commentaire	Entier long	2
XML Instruction de traitement	Entier long	3
XML Donnée	Entier long	6
XML CDATA	Entier long	7
XML DOCTYPE	Entier long	10
XML Elément	Entier long	11

Pour plus d'informations, reportez-vous à la description de la commande DOM Ajouter noeud enfant XML.

Le tableau *tabRefsNoeuds* reçoit les valeurs ou les références des éléments en fonction de leur nature (contenus ou instructions).

- Soit la structure XML suivante :
**<monElement>Bonjour
La
FRANCE</monElement>**

Après l'exécution de ces instructions :

refElement:=**DOM Chercher element XML** ($root;"monElement")
DOM LIRE NOEUDS ENFANTS XML(refElement;$tabtype;$tabtext)

... les tableaux *$tabtype* et *$tabtext* contiendront les valeurs suivantes :
$tabtype{1}=6 $tabtext{1} = "Bonjour"
$tabtype{2}=11 $tabtext{2} = "AEF1233456878977" (référence de l'élément
)
$tabtype{3}=6 $tabtext{3} = "La"
$tabtype{4}=11 $tabtext{4} = "AEF1237897734568" (référence de l'élément
)
$tabtype{5}=6 $tabtext{5} = "FRANCE"

Référence : DOM Ajouter noeud enfant XML

DOM Lire ref document xml

DOM Lire ref document XML (refElément) → Chaîne

Paramètres	Type		Description
refElément	Chaîne	→	Référence d'un élément existant dans un arbre DOM
Résultat	Chaîne	←	Référence du premier élément d'un arbre DOM (noeud document)

La nouvelle commande DOM Lire ref document xml permet de récupérer la référence de l'élément "document" de l'arbre DOM dont vous avez passé la référence dans *refElément*. L'élément document est le premier élément d'un arbre DOM ; c'est le parent de l'élément racine.

La référence de l'élément document vous permet de manipuler les noeuds "Doctype" et "Instructions de traitement". Elle ne peut être utilisée qu'avec les commandes DOM Ajouter noeud enfant XML et DOM LIRE NOEUDS ENFANTS XML.
A ce niveau, vous pouvez uniquement ajouter des instructions de traitement, des commentaires ou remplacer le noeud Doctype. Vous ne pouvez pas y créer de noeud CDATA ou texte.

- Dans cet exemple nous cherchons à retrouver la déclaration de DTD du document XML :

```
C_TEXTE($refRoot)
$refRoot:=DOM Analyser source XML("")
Si (OK=1)
  C_TEXTE($refDocument)
     // on cherche le noeud document, puisque c'est le noeud auquel est
     // rattaché le nœud DOCTYPE avant le noeud root
  $refDocument:=DOM Lire ref document xml($refRoot)
  TABLEAU TEXTE($arrType;0)
  TABLEAU TEXTE($arrValue;0)
```

```
    // sur ce nœud on cherche parmi les enfants le nœud de type DOCTYPE
  DOM LIRE NOEUDS ENFANTS XML($refDocument;$arrType;$arrValue)
  C_TEXTE($text)
  $text:=""
  $pos:=Chercher dans tableau($arrType;XML Doctype)
  Si ($pos>-1)
    // On récupère dans $text la déclaration de DTD
    $text:=$text+"Doctype: "+$arrValue{$pos}+Caractere(Retour chariot)
  Fin de si
  DOM FERMER XML($refRoot)
Fin de si
```

Référence : DOM LIRE NOEUDS ENFANTS XML, DOM Ajouter noeud enfant XML

DOM FIXER DECLARATION XML

DOM FIXER DECLARATION XML(refElément; encodage{; autonome*{; indentation}*}))

Paramètres	Type		Description
refElément	Chaîne	→	Référence d'élément XML
encodage	Chaîne	→	Jeu de caractères du document XML (UTF-8 par défaut)
autonome	Booléen	→	Vrai=le document est autonome, Faux (défaut)=le document n'est pas autonome
indentation	*Booléen*	→	*Obsolète, ne plus utiliser*

Notes de compatibilité

- La commande DOM FIXER DECLARATION XML était nommée DOM ECRIRE OPTIONS XML dans les versions précédentes de 4D.
- Les options DOM ne sont plus dépendantes des options SAX.

Le paramètre *indentation* est conservé pour des raisons de compatibilité avec les versions précédentes de 4D mais son usage est déconseillé dans 4D v12. Désormais, pour définir l'indentation du document, il est fortement recommandé d'utiliser la nouvelle commande XML FIXER OPTIONS.

DOM LIRE VALEUR ELEMENT XML

DOM LIRE VALEUR ELEMENT XML(refElément; valeurElément{; cDATA})

Désormais, si l'objet reçu dans *valeurElément* est un BLOB, la commande tentera automatiquement de le décoder en base64. En effet, si le BLOB a été créé par la commande DOM ECRIRE VALEUR ELEMENT XML, il a été automatiquement encodé en base64. Il n'est plus nécessaire d'appeler la commande DECODER BASE64.

Référence : SAX LIRE VALEUR ELEMENT XML

XML SAX

Trois commandes du thème XML SAX ont été modifiées. Du fait de l'implémentation des commandes XML FIXER OPTIONS et XML LIRE OPTIONS, la commande existante SAX ECRIRE OPTIONS XML a été renommée SAX FIXER DECLARATION XML et son fonctionnement a été adapté.

SAX FIXER DECLARATION XML

SAX FIXER DECLARATION XML(document; encodage{; autonome*{; indentation}*})

Paramètres	Type		Description
document	DocRef	→	Référence de document ouvert
encodage	Chaîne	→	Jeu de caractères du document XML (UTF-8 par défaut)
autonome	Booléen	→	Vrai=le document est autonome, Faux (défaut)=le document n'est pas autonome
indentation	*Booléen*	→	*Obsolète, ne plus utiliser*

Notes de compatibilité

- La commande SAX FIXER DECLARATION XML était nommée SAX ECRIRE OPTIONS XML dans les versions précédentes de 4D.
- Les options SAX sont désormais valides uniquement pour le document ouvert et ne sont plus conservées après la fermeture de ce document.

Le paramètre *indentation* est conservé pour des raisons de compatibilité avec les versions précédentes de 4D mais son usage est déconseillé dans 4D v12. Désormais, pour définir l'indentation du document, il est fortement recommandé d'utiliser la nouvelle commande XML FIXER OPTIONS.

SAX OUVRIR ELEMENT XML TABLEAUX

SAX OUVRIR ELEMENT XML TABLEAUX (document; balise{; tabNomsAttributs; tabValeursAttributs}{; tabNomsAttributs2; tabValeursAttributs2; ...; tabNomsAttributsN; tabValeursAttributsN})

Outre les tableaux de type texte, la commande SAX OUVRIR ELEMENT XML TABLEAUX accepte désormais des tableaux de type date, numérique, booléen et image comme paramètre(s) *tabValeursAttributs*. 4D effectue automatiquement les conversions nécessaires, vous pouvez modifier ces conversions à l'aide de la nouvelle commande XML FIXER OPTIONS.

SAX LIRE VALEUR ELEMENT XML

SAX LIRE VALEUR ELEMENT XML(document; valeur)

A l'image de la commande DOM LIRE VALEUR ELEMENT XML, 4D tentera de convertir la *valeur* obtenue dans le type de la variable passée en paramètre. Si le type est BLOB, la commande tentera automatiquement de le décoder en base64.

Référence : DOM LIRE VALEUR ELEMENT XML

Nouvelles variables système

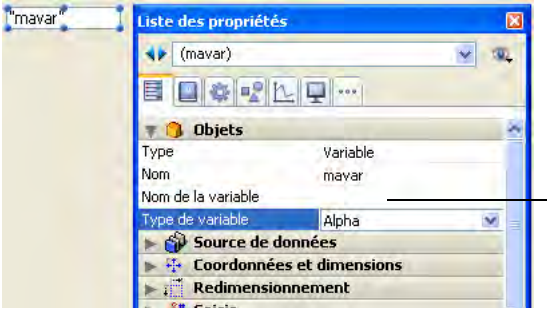
Deux nouvelles variables système sont disponibles dans 4D v12. Utilisables dans le cadre d'une méthode appelée sur erreur, ces variables permettent de déterminer précisément la source de l'erreur :

- *Error method* : variable système de type texte. Cette variable contient le nom complet de la méthode ayant déclenché l'erreur.
- *Error line* : variable système de type entier long. Cette variable contient le numéro de la ligne à l'origine de l'erreur dans la méthode ayant déclenché l'erreur.

A l'instar de la variable existante *Error*, ces deux variables ne peuvent être utilisées que durant l'exécution d'une méthode d'interception d'erreurs installée par la commande APPELER SUR ERREUR.
Si vous souhaitez que ces variables soient accessibles dans la méthode ayant provoqué l'erreur, copiez leur valeur dans votre propre variable process.

Variables dynamiques

Vous pouvez désormais laisser à 4D le soin de créer dynamiquement et en fonction des besoins, les variables associées à vos objets de formulaires (boutons, variables saisissables, cases à cocher, etc.). Pour cela, il suffit de laisser vide le champ "Nom de la variable" dans la Liste des propriétés pour l'objet :



Lorsqu'une variable n'est pas nommée, au moment du chargement du formulaire, 4D crée pour l'objet une nouvelle variable avec un nom calculé unique dans l'espace des variables process de l'interpréteur (ce qui permet d'utiliser ce mécanisme même en mode compilé). Cette variable temporaire sera détruite à la fermeture du formulaire.

Pour que ce principe puisse fonctionner en mode compilé, il est impératif que les variables dynamiques soient explicitement typées via le menu "Type de variable" de la Liste des propriétés. Si vous définissez une variable dynamique (variable non nommée) et sélectionnez la valeur **Aucun** dans le menu "Type de variable", une erreur de typage sera retournée par le compilateur.

Note Comme dans les versions précédentes de 4D, lorsque la variable est nommée, le menu "Type de variable" ne type pas réellement la variable mais permet uniquement de mettre à jour les options de la Liste des propriétés (hormis pour les variables image). Pour typer une variable nommée, il est nécessaire d'utiliser le langage.

Dans le code 4D, les variables dynamiques sont accessibles via un pointeur obtenu avec la commande OBJET Lire pointeur. Par exemple :

```
// affecter l'heure 12:00:00 à la variable de l'objet "hdebut"
$p := Objet Lire pointeur(Objet nommé;"hdebut")
$p-> := ?12:00:00?
```

L'intérêt de ce nouveau mécanisme est double :

- D'une part, il permet le développement de composants de type "sous-formulaire" pouvant être utilisés plusieurs fois dans un même formulaire hôte. Prenons le cas par exemple d'un sous-formulaire *datepicker* inséré deux fois dans un formulaire hôte pour définir une date de début et une date de fin. Ce sous-formulaire va utiliser des objets pour le choix du jour du mois et de l'année. Il faut donc que ces objets travaillent avec des variables différentes pour la date de début et la date de fin. Laisser 4D créer leur variable avec un nom unique permet de résoudre cette difficulté.
- D'autre part, il permet de limiter la consommation mémoire. En effet, les objets de formulaire ne travaillent qu'avec des variables process ou interprocess. Or, en mode compilé, une instance de chaque variable process est créée dans tous les process, y compris les process du serveur. Cette instance consomme de la mémoire, même si le formulaire n'est pas utilisé au cours de la session. Laisser 4D créer dynamiquement les variables au chargement des formulaires permet donc d'économiser la mémoire.

Notes
• Lorsqu'il n'y a pas de nom de variable, c'est le nom de l'objet encadré de guillemets qui est affiché dans l'éditeur de formulaires.
• Ce mécanisme est disponible pour tous les objets associés à un nom de variable sauf les listbox.

Constantes

Des fonctionnalités supplémentaires sont accessibles via de nouvelles constantes, décrites dans cette section.

Creer fenetre

La nouvelle constante Avec bouton barre outils Mac OS permet d'afficher un bouton de gestion de la barre d'outils dans une fenêtre créée sous Mac OS (cf. paragraphe "Bouton barre d'outils (Mac OS)", page 121).

La constante Aspect métal a été renommée Aspect texture, en conformité avec l'évolution de l'interface Mac OS (l'option "Aspect métal" de la palette des propriétés de l'éditeur de formulaires a également été renommée "Texture (sous Mac OS)").

Constante	Type	Valeur
Avec bouton barre outils Mac OS	Entier long	8192
Aspect texture	Entier long	2048

- Exemple d'utilisation :

$NewWin:=**Creer fenetre**(10;10;1010;810;Fenêtre standard+Avec bouton barre outils Mac OS)

Note La constante Avec bouton barre outils Mac OS peut également être utilisée avec la commande **Creer fenetre formulaire**.

Documents système

Une nouvelle constante a été ajoutée dans le thème "Documents système" : Séparateur dossier. Cette constante a une valeur différente en fonction du système d'exploitation depuis lequel elle est appelée :

Constante	Type	Valeur
Séparateur dossier	Chaîne	\ (Windows) ou : (Mac OS)

Cette constante permet de construire des chemins d'accès valides sans tenir compte de la plate-forme d'exécution.

Evenements formulaire

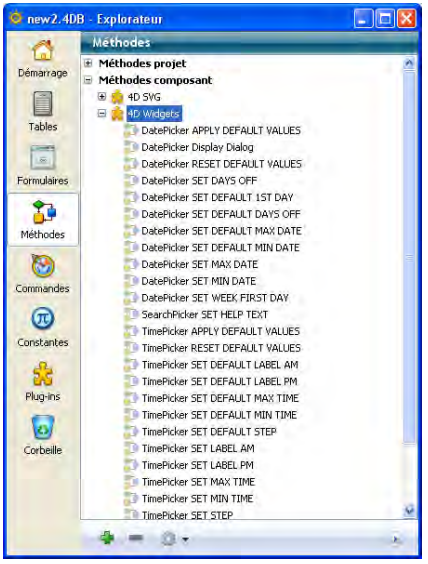
Deux nouveaux événements formulaire ont été ajoutés dans le thème "Evénements formulaire" :

Constante	Type	Valeur	Description
Sur modif variable liée	Entier long	54	Généré lorsque la variable liée à un sous-formulaire est modifiée. Voir paragraphe "Gestion de la variable liée", page 76.
Sur bouton barre outils Mac	Entier long	55	Généré lorsque l'utilisateur clique sur le bouton de gestion de la barre d'outils sous Mac OS. Voir paragraphe "Bouton barre d'outils (Mac OS)", page 121.

4D Widgets

4D v12 intègre des objets de formulaire évolués fournis via un nouveau composant, nommé 4D Widgets. Ces objets sont décrits dans le paragraphe “Widgets”, page 84.

Le paramétrage de ce composant peut être effectué via des méthodes projet spécifiques, listées dans la page Méthodes de l’Explorateur :



Cette section décrit la syntaxe de ces méthodes.

DatePicker

Le composant DatePicker propose des objets d’interface graphiques intuitives pour la saisie et la représentation des dates dans les formulaires. Pour plus d’informations, reportez-vous au paragraphe “DatePicker”, page 86.

Les méthodes de ce composant permettent de personnaliser le fonctionnement de ces objets. La commande DatePicker Display Dialog permet d’afficher un objet calendrier DatePicker sous forme de pop up menu.

DatePicker Display Dialog

DatePicker Display Dialog {(haut; gauche)} → Date

Paramètres	Type		Description
haut	Entier long	→	Emplacement haut de la fenêtre
gauche	Entier long	→	Emplacement gauche de la fenêtre
Résultat	Date	←	Date sélectionnée par l'utilisateur

La commande DatePicker Display Dialog ouvre un calendrier DatePicker dans une fenêtre pop up (une fenêtre de type pop up est automatiquement refermée lorsque l'utilisateur clique en-dehors de la fenêtre ou appuie sur la touche **Entrée** ou **Echap**).

Les deux paramètres optionnels permettent de définir l'emplacement de l'angle supérieur gauche de la fenêtre à ouvrir. Ces deux paramètres doivent être passés ensemble ; si un seul est passé, il est ignoré.
Si ces paramètres sont omis, la fenêtre est ouverte à l'emplacement du clic.

La fonction retourne la date sélectionnée par l'utilisateur dans le calendrier DatePicker. Si la fenêtre est refermée sans qu'une date ait été sélectionnée, la commande retourne une date vide.

- Cet exemple affiche un calendrier DatePicker en cas de clic sur un bouton :

```
OBJET LIRE COORDONNEES (*;"MonBtnCalendrier";$x1;$y1;$x2;$y2)
$MaDateLocale:=DatePicker Display Dialog ($x1;$y1)
Si($MaDateLocale #!00/00/00!)
   [Evenement]DateRV:=$MaDateLocale
Fin de si
```

DatePicker SET MIN DATE

DatePicker SET MIN DATE (nomObjet; dateMin)

Paramètres	Type		Description
nomObjet	Texte	→	Nom d'objet sous-formulaire
dateMin	Date	→	Limite inférieure de date saisissable

La commande DatePicker SET MIN DATE permet de définir la date saisissable minimale dans un calendrier DatePicker (les jours situés avant cette date minimum apparaissent grisés dans le calendrier).

Le paramètre *nomObjet* désigne l'instance de sous-formulaire à laquelle la commande doit être appliquée. Vous devez passer dans ce paramètre un nom d'objet sous-formulaire affiché dans le formulaire courant.

La date *dateMin* doit être exprimée dans le format de saisie par défaut correspondant à la langue du système. Si vous passez une date vide (!00/00/00!), toutes les dates qui précèdent la date courante sont saisissables.

Note Si la date saisissable minimale est postérieure à la date saisissable maximale (cf. DatePicker SET MAX DATE), aucune date ne sera saisissable.

- Le formulaire courant contient deux calendriers DatePicker situés dans deux objets sous-formulaires nommés "DP1" et "DP2".

```
//Inactivation de toutes les dates avant le 1er janvier 2009 dans le premier calendrier
DatePicker SET MIN DATE("DP1";!01/01/2009!)

//Inactivation de toutes les dates avant le 1er mars 2009 dans le deuxième calendrier
DatePicker SET MIN DATE("DP2";!01/03/2009!)
```

Référence : DatePicker SET MAX DATE

DatePicker SET MAX DATE

DatePicker SET MAX DATE (nomObjet; dateMax)

Paramètres	Type		Description
nomObjet	Texte	→	Nom d'objet sous-formulaire
dateMax	Date	→	Limite supérieure de date saisissable

La commande DatePicker SET MAX DATE permet de définir la date saisissable maximale dans un calendrier DatePicker (les jours situés après cette date maximale apparaissent grisés dans le calendrier).

Le paramètre *nomObjet* désigne l'instance de sous-formulaire à laquelle la commande doit être appliquée. Vous devez passer dans ce paramètre un nom d'objet sous-formulaire affiché dans le formulaire courant.

La date *dateMax* doit être exprimée dans le format de saisie par défaut correspondant à la langue du système. Si vous passez une date vide (!00/00/00!), toutes les dates qui suivent la date courante sont saisissables.

Note Si la date saisissable maximale est antérieure à la date saisissable minimale (cf. DatePicker SET MIN DATE), aucune date ne sera saisissable.

- Inactivation de toutes les dates après le 31 décembre 2009 dans l'objet nommé "DateRetour" :

DatePicker SET MAX DATE("DateRetour";!31/12/2009!)

Référence : DatePicker SET MIN DATE

DatePicker SET WEEK FIRST DAY

DatePicker SET WEEK FIRST DAY (nomObjet; numJour)

Paramètres	Type		Description
nomObjet	Texte	→	Nom d'objet sous-formulaire
numJour	Entier long	→	Numéro du premier jour à afficher

La commande DatePicker SET WEEK FIRST DAY permet de désigner le premier jour de la semaine à afficher dans la partie gauche d'un calendrier DatePicker. Par défaut, le premier jour est lundi.

Le paramètre *nomObjet* désigne l'instance de sous-formulaire à laquelle la commande doit être appliquée. Vous devez passer dans ce paramètre un nom d'objet sous-formulaire affiché dans le formulaire courant.

Passez dans le paramètre *numJour* une constante 4D du thème "Jours et mois".

- Paramétrage du premier jour au dimanche :

DatePicker SET WEEK FIRST DAY("moncalendrier";Dimanche)

- Paramétrage du premier jour au jeudi :

DatePicker SET WEEK FIRST DAY("moncalendrier";Jeudi)

Référence : DatePicker SET DAYS OFF

DatePicker SET DAYS OFF

DatePicker SET DAYS OFF (nomObjet; typeJour; ptrTabJoursCongés)

Paramètres	Type		Description
nomObjet	Texte	→	Nom d'objet sous-formulaire
typeJour	Entier long	→	Types de jours fériés
ptrTabJoursCongés	Pointeur	→	Pointeur vers le tableau date ou booléen des jours de congés

La commande DatePicker SET DAYS OFF permet de définir les "jours de congés" à faire figurer dans le calendrier DatePicker. Ces jours sont affichés en gras italique et restent sélectionnables par l'utilisateur.

Le paramètre *nomObjet* désigne l'instance de sous-formulaire à laquelle la commande doit être appliquée. Vous devez passer dans ce paramètre un nom d'objet sous-formulaire affiché dans le formulaire courant.

La commande permet de définir des jours de congés récurrents hebdomadaires ou annuels ainsi que des jours "fériés" ponctuels. Vous précisez le type de jour de congé défini par la méthode via le paramètre *typeJour* :

- 0 = jours de congés répétés chaque semaine (par défaut, le samedi et le dimanche)
- 1 = jours de congés répétés d'année en année (tels que le 1er janvier ou le 25 décembre)
- 2 = jours fériés ponctuels, définis pour une seule année

Vous définissez les jours à l'aide d'un tableau et en passant un pointeur vers ce tableau comme paramètre *ptrTabJoursCongés*. Le type de tableau dépend de la valeur passée dans *typeJour* :

- si vous avez passé 0 dans *typeJour* (jours de congés hebdomadaires), vous devez passer dans *ptrTabJoursCongés* un pointeur vers un tableau booléen comportant 7 éléments. Chaque élément à Vrai désigne un jour de congé hebdomadaire.
- si vous avez passé 1 ou 2 dans *typeJour* (jours de congés annuels ou ponctuels), vous devez passer dans *ptrTabJoursCongés* un pointeur vers un tableau date. Dans ce tableau, chaque élément doit contenir une date valide, indiquant un jour de congé. Les dates doivent être exprimées dans le format par défaut correspondant à la langue du système. Si vous avez passé 1 dans *typeJour* (jours récurrents), l'année est ignorée, vous pouvez passer n'importe quelle valeur.

- Désignation du vendredi comme jour de congé hebdomadaire (au lieu des samedi et dimanche par défaut) :

```
TABLEAU BOOLEEN ($tbJoursCongés;7)
   //Par défaut, tous les éléments d'un tableau booléen sont à Faux, il n'est
   //donc pas nécessaire d'ajouter du code d'initialisation
$tbJoursCongés{Vendredi }:=Vrai
DatePicker SET DAYS OFF ("moncalendrier";0;->$tbJoursCongés)
```

- Désignation de jours fériés ponctuels :

```
TABLEAU DATE($tdJoursUniques;0)
   `L'année est prise en compte
AJOUTER A TABLEAU($tdJoursUniques;!15/02/2008!)
AJOUTER A TABLEAU($tdJoursUniques;!12/02/2009!)
AJOUTER A TABLEAU($tdJoursUniques;!17/02/2010!)
DatePicker SET DAYS OFF ("moncalendrier";2;->$tdJoursUniques)
```

Référence : DatePicker SET WEEK FIRST DAY, DatePicker SET DEFAULT DAYS OFF

DatePicker SET DEFAULT DAYS OFF

DatePicker SET DEFAULT DAYS OFF (typeJour; ptrTabJoursCongés)

Paramètres	Type		Description
typeJour	Entier long	→	Types de jours fériés
ptrTabJoursCongés	Pointeur	→	Pointeur vers le tableau date ou booléen des jours de congés

La commande DatePicker SET DEFAULT DAYS OFF vous permet de définir les "jours de congés" à faire figurer dans la totalité des calendriers du composant DatePicker. Ces jours sont affichés en gras italique et restent sélectionnables par l'utilisateur.

A noter que ce paramétrage est pris en compte uniquement pour les calendriers créés par la suite et n'est pas appliqué aux calendriers existants. Si vous souhaitez l'appliquer aux calendriers existants, vous devez utiliser la commande DatePicker APPLY DEFAULT VALUES.

La commande permet de définir des jours de congés récurrents hebdomadaires ou annuels ainsi que des jours "fériés" ponctuels. Vous précisez le type de jour de congé défini par la méthode via le paramètre *typeJour* :

- 0 = jours de congés répétés chaque semaine (par défaut, le samedi et le dimanche)
- 1 = jours de congés répétés d'année en année (tels que le 1er janvier ou le 25 décembre)
- 2 = jours fériés ponctuels, définis pour une seule année

Vous définissez les jours à l'aide d'un tableau et en passant un pointeur vers ce tableau comme paramètre *ptrTabJoursCongés*. Le type de tableau dépend de la valeur passée dans *typeJour* :

- si vous avez passé 0 dans *typeJour* (jours de congés hebdomadaires), vous devez passer dans *ptrTabJoursCongés* un pointeur vers un tableau booléen comportant 7 éléments. Chaque élément à Vrai désigne un jour de congé hebdomadaire.
- si vous avez passé 1 ou 2 dans *typeJour* (jours de congés annuels ou ponctuels), vous devez passer dans *ptrTabJoursCongés* un pointeur vers un tableau date. Dans ce tableau, chaque élément doit contenir une date valide, indiquant un jour de congé. Les dates doivent être exprimées dans le format par défaut correspondant à la langue du système. Si vous avez passé 1 dans *typeJour* (jours récurrents), l'année est ignorée, vous pouvez passer n'importe quelle valeur.

▸ Désignation des jours fériés récurrents (exemple valide en France) :

```
TABLEAU DATE($tdJoursRépétés;0)
   `L'année est ignorée, nous utilisons 2000 par défaut
AJOUTER A TABLEAU($tdJoursRépétés;!01/01/2000!)
AJOUTER A TABLEAU($tdJoursRépétés;!01/05/2000!)
AJOUTER A TABLEAU($tdJoursRépétés;!08/05/2000!)
AJOUTER A TABLEAU($tdJoursRépétés;!14/07/2000!)
AJOUTER A TABLEAU($tdJoursRépétés;!15/08/2000!)
AJOUTER A TABLEAU($tdJoursRépétés;!01/11/2000!)
AJOUTER A TABLEAU($tdJoursRépétés;!11/11/2000!)
AJOUTER A TABLEAU($tdJoursRépétés;!25/12/2000!)
DatePicker SET DEFAULT DAYS OFF (1;->$tdJoursRépétés)
```

Référence : DatePicker SET DAYS OFF

DatePicker SET DEFAULT MIN DATE

DatePicker SET DEFAULT MIN DATE (dateMin)

Paramètres	Type		Description
dateMin	Date	→	Limite inférieure de date saisissable

La commande DatePicker SET DEFAULT MIN DATE vous permet de définir la date saisissable minimale pour la totalité des calendriers du composant DatePicker.

A noter que ce paramétrage est pris en compte uniquement pour les calendriers créés par la suite et n'est pas appliqué aux calendriers existants. Si vous souhaitez l'appliquer aux calendriers existants, vous devez utiliser la commande DatePicker APPLY DEFAULT VALUES.

- Désignation de la date minimale au 1er janvier 2000 :

  DatePicker SET DEFAULT MIN DATE (!01/01/2000!)

Référence : DatePicker SET MIN DATE, DatePicker SET DEFAULT MAX DATE

DatePicker SET DEFAULT MAX DATE

DatePicker SET DEFAULT MAX DATE (dateMax)

Paramètres	Type		Description
dateMax	Date	→	Limite supérieure de date saisissable

La commande DatePicker SET DEFAULT MAX DATE vous permet de définir la date saisissable maximale pour la totalité des calendriers du composant DatePicker.

A noter que ce paramétrage est pris en compte uniquement pour les calendriers créés par la suite et n'est pas appliqué aux calendriers existants. Si vous souhaitez l'appliquer aux calendriers existants, vous devez utiliser la commande DatePicker APPLY DEFAULT VALUES.

Référence : DatePicker SET MAX DATE, DatePicker SET DEFAULT MIN DATE

DatePicker SET DEFAULT 1ST DAY

DatePicker SET DEFAULT 1ST DAY(numJour)

Paramètres	Type		Description
numJour	Entier long	→	Premier jour de la semaine

La commande DatePicker SET DEFAULT 1ST DAY vous permet de désigner le premier jour de la semaine à afficher par défaut dans la partie gauche des calendriers DatePicker.

A noter que ce paramétrage est pris en compte uniquement pour les calendriers créés par la suite et n'est pas appliqué aux calendriers existants. Si vous souhaitez l'appliquer aux calendriers existants, vous devez utiliser la commande DatePicker APPLY DEFAULT VALUES.

Référence : DatePicker SET WEEK FIRST DAY

DatePicker APPLY DEFAULT VALUES

DatePicker APPLY DEFAULT VALUES (nomObjet)

Paramètres	Type		Description
nomObjet	Texte	→	Nom d'objet sous-formulaire

La commande DatePicker APPLY DEFAULT VALUES permet de réinitialiser tous les paramètres DatePicker à leur valeur par défaut pour l'objet de sous-formulaire *nomObjet*.

Ces valeurs par défaut peuvent être les réglages d'usine mais peuvent également avoir été modifiées via les commandes SET DEFAULT du composant.
L'action de la commande est immédiate : les valeurs par défaut de *nomObjet* sont instantanément modifiées. A noter que la variable associée à l'objet pourra également être modifiée afin de tenir compte des nouvelles valeurs. Par exemple, si les nouvelles valeurs par défaut fixent la date minimale au 01/01/2000 et que la variable associée à *nomObjet* était le 05/05/1995, sa valeur est automatiquement ramenée au 01/01/2000.

Les paramètres DatePicker incluent :

- les dates saisissables minimale et maximale
- le premier jour de la semaine
- les "jours de congés" hebdomadaires, annuels et spécifiques

▸ Cet exemple réinitialise les paramètres de l'objet Date1 à leurs réglages par défaut :

DatePicker APPLY DEFAULT VALUES ("Date1")

DatePicker RESET DEFAULT VALUES

DatePicker RESET DEFAULT VALUES

Paramètres	Type	Description
Cette commande ne nécessite aucun paramètre		

La commande DatePicker RESET DEFAULT VALUES réinitialise les paramètres du composant DatePicker à leurs "réglages d'usine". Après l'exécution de cette commande :

- les dates saisissables minimale et maximale sont 00/00/00 (ce qui signifie qu'il n'y a pas de limites)
- le premier jour de la semaine est 2 (lundi)
- les "jours de congés" hebdomadaires sont le samedi et le dimanche
- aucun jour férié annuel ni spécifique n'est défini.

A noter que ce paramétrage est pris en compte uniquement pour les calendriers créés par la suite et n'est pas appliqué aux calendriers existants. Si vous souhaitez l'appliquer aux calendriers existants, vous devez utiliser la commande DatePicker APPLY DEFAULT VALUES.

SearchPicker

Le composant SearchPicker permet d'inclure dans vos formulaires des zones de recherche standard. Pour plus d'informations, reportez-vous au paragraphe "SearchPicker", page 85.

La méthode de ce composant permet de définir le texte d'aide de la zone.

SearchPicker SET HELP TEXT

SearchPicker SET HELP TEXT (nomObjet; texteAide)

Paramètres	Type		Description
nomObjet	Texte	→	Nom d'objet sous-formulaire
texteAide	Texte	→	Texte à afficher

La commande SearchPicker SET HELP TEXT vous permet d'afficher un texte gris non saisissable en arrière-plan de la zone de recherche désignée par *nomObjet*. Ce texte disparaît lorsque l'utilisateur clique dans la zone.

▶ Affichage du mot "Country" dans la zone, indiquant que la recherche portera sur cette variable :

SearchPicker SET HELP TEXT("vRech";"Country")

TimePicker

Le composant TimePicker propose des objets graphiques pour la saisie et la représentation des heures dans les formulaires. Pour plus d'informations, reportez-vous au paragraphe "TimePicker", page 90.

Les méthodes de ce composant permettent de personnaliser le fonctionnement de ces objets.

TimePicker SET MIN TIME

TimePicker SET MIN TIME (nomObjet; heureMin)

Paramètres	Type		Description
nomObjet	Texte	→	Nom d'objet sous-formulaire
heureMin	Heure	→	Limite inférieure d'heure saisissable

La commande TimePicker SET MIN TIME permet de définir l'heure saisissable minimale qui sera acceptée par l'objet désigné par *nomObjet*. Si une valeur d'heure inférieure est saisie, elle sera rejetée.

Référence : TimePicker SET MAX TIME

TimePicker SET MAX TIME

TimePicker SET MAX TIME (nomObjet; heureMax)

Paramètres	Type		Description
nomObjet	Texte	→	Nom d'objet sous-formulaire
heureMax	Heure	→	Limite supérieure d'heure saisissable

La commande TimePicker SET MAX TIME permet de définir l'heure saisissable maximale qui sera acceptée par l'objet désigné par *nomObjet*. Si une valeur d'heure supérieure est saisie, elle sera rejetée.

Référence : TimePicker SET MIN TIME

TimePicker SET STEP

TimePicker SET STEP (nomObjet; intervalle)

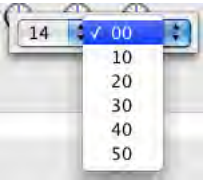
Paramètres	Type		Description
nomObjet	Texte	→	Nom d'objet sous-formulaire
intervalle	Heure	→	Intervalle entre deux valeurs d'heures

La commande TimePicker SET STEP permet de définir l'intervalle entre les valeurs d'heures disponibles pour l'objet désigné par *nomObjet*. Ce paramétrage s'applique aux TimePickers affichés sous forme de pop up menus uniquement.

La valeur d'*intervalle* doit être comprise entre 1 mn et 1 heure et doit pouvoir être représentée sous forme de divisions entières de 60 mn. En pratique, seules les valeurs de 1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30 et 60 mn sont possibles. Toute valeur différente sera automatiquement arrondie de manière à correspondre à ce principe.

- Paramétrage du TimePicker en pop up menu nommé "heure1", limitation des heures saisissables de 8h30 à 16h30 et intervalles de 10 mn :

TimePicker SET MIN TIME ("heure1";?08:30:00?)
TimePicker SET MAX TIME ("heure1";?16:30:00?)
TimePicker SET STEP ("heure1";?00:10:00?)



TimePicker SET LABEL AM

TimePicker SET LABEL AM (nomObjet; libellé)

Paramètres	Type		Description
nomObjet	Texte	→	Nom d'objet sous-formulaire
libellé	Texte	→	Libellé à utiliser pour AM

La commande TimePicker SET LABEL AM permet de modifier le libellé "AM" dans les objets TimePicker affichant le format AM/PM. La commande s'applique à l'objet désigné par *nomObjet*. Par défaut, les libellés système am/pm sont utilisés.

- Utilisation par défaut du libellé "du matin" au lieu du libellé système pour AM :

TimePicker SET LABEL AM("horloge"; "du matin")

Référence : TimePicker SET LABEL PM

TimePicker SET LABEL PM

TimePicker SET LABEL PM (nomObjet; libellé)

Paramètres	Type		Description
nomObjet	Texte	→	Nom d'objet sous-formulaire
libellé	Texte	→	Libellé à utiliser pour PM

La commande TimePicker SET LABEL PM permet de modifier le libellé "PM" dans les objets TimePicker affichant le format AM/PM. La commande s'applique à l'objet désigné par *nomObjet*. Par défaut, les libellés système am/pm sont utilisés.

- Utilisation par défaut du libellé "du soir" au lieu du libellé système pour PM :

```
TimePicker SET LABEL PM("horloge"; "du soir")
```

Référence : TimePicker SET LABEL AM

TimePicker SET DEFAULT MIN TIME

TimePicker SET DEFAULT MIN TIME (heureMin)

Paramètres	Type		Description
heureMin	Heure	→	Limite inférieure d'heure saisissable

La commande TimePicker SET DEFAULT MIN TIME permet de définir l'heure saisissable minimale qui sera acceptée par défaut par tous les objets TimePicker.

Ce paramétrage est pris en compte uniquement pour les objets créés par la suite et n'est pas appliqué aux objets existants. Si vous souhaitez l'appliquer aux objets existants, vous devez utiliser la commande TimePicker APPLY DEFAULT VALUES.

Référence : TimePicker SET MAX TIME

TimePicker SET DEFAULT MAX TIME

TimePicker SET DEFAULT MAX TIME (heureMax)

Paramètres	Type		Description
heureMax	Heure	→	Limite supérieure d'heure saisissable

La commande TimePicker SET DEFAULT MAX TIME permet de définir l'heure saisissable maximale qui sera acceptée par défaut par tous les objets TimePicker.

Ce paramétrage est pris en compte uniquement pour les objets créés par la suite et n'est pas appliqué aux objets existants. Si vous souhaitez l'appliquer aux objets existants, vous devez utiliser la commande TimePicker APPLY DEFAULT VALUES.

Référence : TimePicker SET MIN TIME

TimePicker SET DEFAULT STEP

TimePicker SET DEFAULT STEP (intervalle)

Paramètres	Type		Description
intervalle	Heure	→	Intervalle entre deux valeurs d'heures

La commande TimePicker SET DEFAULT STEP permet de définir par défaut l'intervalle entre les valeurs d'heures pour tous les objets TimePicker.

Ce paramétrage est pris en compte uniquement pour les objets créés par la suite et n'est pas appliqué aux objets existants. Si vous souhaitez l'appliquer aux objets existants, vous devez utiliser la commande TimePicker APPLY DEFAULT VALUES.

Référence : TimePicker SET STEP

TimePicker SET DEFAULT LABEL AM

TimePicker SET DEFAULT LABEL AM (libellé)

Paramètres	Type		Description
libellé	Texte	→	Libellé à utiliser pour AM

La commande TimePicker SET DEFAULT LABEL AM permet de modifier le libellé "AM" par défaut pour tous les objets TimePicker affichant le format AM/PM.

Ce paramétrage est pris en compte uniquement pour les objets créés par la suite et n'est pas appliqué aux objets existants. Si vous souhaitez l'appliquer aux objets existants, vous devez utiliser la commande TimePicker APPLY DEFAULT VALUES.

Référence : TimePicker SET LABEL PM

TimePicker SET DEFAULT LABEL PM

TimePicker SET DEFAULT LABEL PM (libellé)

Paramètres	Type		Description
libellé	Texte	→	Libellé à utiliser pour PM

La commande TimePicker SET DEFAULT LABEL PM permet de modifier le libellé "PM" par défaut pour tous les objets TimePicker affichant le format AM/PM.

Ce paramétrage est pris en compte uniquement pour les objets créés par la suite et n'est pas appliqué aux objets existants. Si vous souhaitez l'appliquer aux objets existants, vous devez utiliser la commande TimePicker APPLY DEFAULT VALUES.

Référence : TimePicker SET LABEL AM

TimePicker APPLY DEFAULT VALUES

TimePicker APPLY DEFAULT VALUES (nomObjet)

Paramètres	Type		Description
nomObjet	Texte	→	Nom d'objet sous-formulaire

La commande TimePicker APPLY DEFAULT VALUES permet de réinitialiser tous les paramètres TimePicker à leur valeur par défaut courante pour l'objet de sous-formulaire *nomObjet*.

Ces valeurs par défaut peuvent être les réglages d'usine mais peuvent également avoir été modifiées via les commandes SET DEFAULT du composant.
L'action de la commande est immédiate : les valeurs par défaut de *nomObjet* sont instantanément modifiées. A noter que la variable associée à l'objet pourra également être modifiée afin de tenir compte des nouvelles valeurs. Par exemple, si les nouvelles valeurs par défaut fixent l'heure minimale à 07:00:00 et que la variable associée à *nomObjet* était 06:00:00, sa valeur est automatiquement ramenée à 07:00:00.

Les paramètres TimePicker incluent :

- les heures saisissables minimales et maximales,
- les libellés AM et PM ,
- les intervalles de minutes.

Référence : TimePicker RESET DEFAULT VALUES

TimePicker RESET DEFAULT VALUES

TimePicker RESET DEFAULT VALUES

Paramètres	Type	Description
Cette commande ne nécessite aucun paramètre		

La commande TimePicker RESET DEFAULT VALUES réinitialise les paramètres du composant TimePicker à leurs "réglages d'usine". Après l'exécution de cette commande :

- l'heure saisissable minimale est 08:00:00
- l'heure saisissable maximale est 20:00:00
- les libellés AM et PM sont ceux définis dans le système
- l'intervalle de minutes est 00:15:00

A noter que ce paramétrage est pris en compte uniquement pour les objets TimePicker créés par la suite et n'est pas appliqué aux objets existants. Si vous souhaitez l'appliquer aux objets existants, vous devez appeler la commande TimePicker APPLY DEFAULT VALUES.

4D SVG

La version 12 du composant 4D SVG comporte plusieurs nouvelles méthodes et des méthodes modifiées. Toutes ces nouveautés sont décrites ci-dessous.

Attributs

SVG_SET_CLASS

SVG_SET_CLASS (objetSVG ; nomClasse)

Paramètre	Type		Description
objetSVG	Ref_SVG	→	Référence d'un élément SVG
nomClasse	Texte	→	Nom de la classe

La commande SVG_SET_CLASS affecte la classe *nomClasse* à l'objet passé dans *objetSVG*. Une erreur est générée si *objetSVG* n'est pas une référence valide.

▸ Reportez-vous à l'exemple de la commande SVG_Define_style.

Référence : http://www.yoyodesign.org/doc/w3c/svg1/styling.html#StyleElement

SVG_SET_CLIP_PATH

SVG_SET_CLIP_PATH (objetSVG; idRognage)

Paramètre	Type		Description
objetSVG	Ref_SVG	→	Référence d'un élément SVG
idRognage	Texte	→	Nom du tracé de rognage

La commande SVG_SET_CLIP_PATH assigne le tracé de rognage nommé *idRognage* à l'objet SVG désigné par *objetSVG*. Une erreur est générée si *objetSVG* n'est pas une référence valide ou si le tracé le rognage n'est pas défini.

- Définition d'un tracé de rognage circulaire ensuite attribué à une image.

```
//Définition d'un tracé circulaire
$Dom_clipPath:=SVG_Define_clip_path ($Dom_SVG;"theClip")
$Dom_circle:=SVG_New_circle ($Dom_clipPath;150;100;100)

//Création d'un groupe
$Dom_g:=SVG_New_group ($Dom_SVG)

//Insertion d'une image
$Txt_path:= Dossier 4D(6)+"logo.svg"
LIRE FICHIER IMAGE($Txt_path;$Pic_buffer)
$Dom_picture:=SVG_New_embedded_image ($Dom_g;$Pic_buffer)
SVG_SET_ID ($Dom_picture;"MyPicture")

//Application du rognage au groupe
SVG_SET_CLIP_PATH ($Dom_g;"theClip")
```

- La même image avec un tracé de rognage rectangulaire à coin arrondi :

```
//Définition d'un tracé rectangulaire
$Dom_clipPath:=SVG_Define_clipPath ($Dom_SVG;"theClip")
$Dom_rect:=SVG_New_rect ($Dom_clipPath;5;10;320;240;10;10)
```

```
//Création d'un groupe
$Dom_g:=SVG_New_group ($Dom_SVG)

//Insertion d'une image
$Txt_path:= Dossier 4D(6)+"logo.svg"
LIRE FICHIER IMAGE($Txt_path;$Pic_buffer)
$Dom_picture:=SVG_New_embedded_image ($Dom_g;$Pic_buffer)
SVG_SET_ID ($Dom_picture;"MyPicture")

//Application du rognage au groupe
SVG_SET_CLIP_PATH ($Dom_g;"theClip")
```

Référence : http://www.yoyodesign.org/doc/w3c/svg1/masking.html#EstablishingANewClippingPath

SVG_SET_FILL_RULE

SVG_SET_FILL_RULE (objetSVG; modeRemplissage)

Paramètre	Type		Description
objetSVG	Ref_SVG	→	Référence d'un élément SVG
modeRemplissage	Texte	→	Mode de remplissage de l'objet

La commande SVG_SET_FILL_RULE permet de préciser le mode de remplissage de l'objet SVG désigné par *objetSVG*. Une erreur est générée si *objetSVG* n'est pas une référence valide.

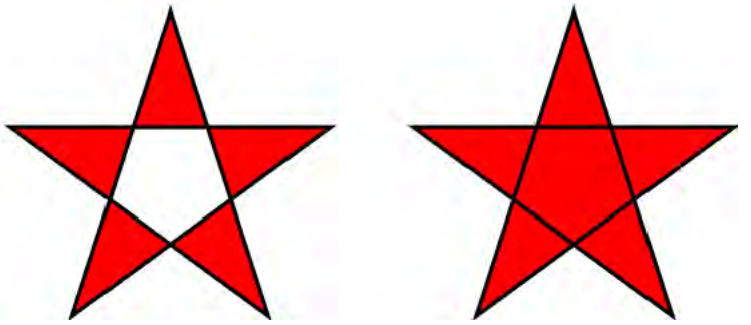
Le paramètre *modeRemplissage* doit contenir l'une des valeurs suivantes : "nonzero", "evenodd" ou "inherit". Dans le cas contraire, une erreur est générée.

- Illustration des modes de remplissage :

```
//Création d'un tracé avec le mode de remplissage 'evenodd'
$Dom_path:=SVG_New_path ($Dom_SVG;250;75)
SVG_PATH_LINE_TO ($Dom_path;323;301;131;161;369;161;177;301)
SVG_PATH_CLOSE ($Dom_path)
```

```
SVG_SET_FILL_BRUSH ($Dom_path;"red")
SVG_SET_STROKE_WIDTH ($Dom_path;3)
SVG_SET_FILL_RULE ($Dom_path;"evenodd")

//Création d'un objet similaire avec le mode de remplissage 'nonzero'
$Dom_path:=SVG_New_path ($Dom_SVG;250;75)
SVG_PATH_LINE_TO ($Dom_path;323;301;131;161;369;161;177;301)
SVG_PATH_CLOSE ($Dom_path)
SVG_SET_FILL_BRUSH ($Dom_path;"red")
SVG_SET_STROKE_WIDTH ($Dom_path;3)
SVG_SET_FILL_RULE ($Dom_path;"nonzero")
//Déplacement horizontal
SVG_SET_TRANSFORM_TRANSLATE ($Dom_path;300)
```



Référence : http://www.yoyodesign.org/doc/w3c/svg1/painting.html#FillRuleProperty

SVG_SET_SHAPE_RENDERING

SVG_SET_SHAPE_RENDERING (objetSVG; typeRendu)

Paramètre	Type		Description
objetSVG	Ref_SVG	→	Référence d'un élément SVG
typeRendu	Texte	→	Type de rendu de l'élément

La commande SVG_SET_SHAPE_RENDERING permet de fixer les compromis à faire pour le rendu des éléments graphiques de l'objet désigné par *objetSVG*. Si *objetSVG* n'est pas un objet SVG, une erreur est générée.

Le paramètre *typeRendu* doit contenir l'une des valeurs suivantes : "auto", "optimizeSpeed", "crispEdges", "geometricPrecision" ou "inherit". Dans le cas contraire, une erreur est générée.

Référence : http://www.yoyodesign.org/doc/w3c/svg1/painting.html#ShapeRenderingProperty

SVG_SET_STROKE_DASHARRAY

SVG_SET_STROKE_DASHARRAY (objetSVG; tiret ; valeur1{ ;...; valeurN})

Paramètre	Type		Description
objetSVG	Ref_SVG	→	Référence d'un élément SVG
tiret	Réel	→	Longueur du premier tiret
valeur1...N	Entier long	→	Longueur des blancs et des tirets

La commande SVG_SET_STROKE_DASHARRAY permet de définir le motif de tirets et de blancs utilisé pour le liseré du tracé de l'objet SVG désigné par *objetSVG*. Une erreur est générée si *objetSVG* n'est pas une référence SVG valide.

La valeur entière du paramètre *tiret* indique la longueur du premier tiret du motif pointillé. Si les paramètres *valeur1* à *valeurN* sont omis, le pointillé sera constitué d'une succession de tirets et de blancs de la même longueur.

La valeur décimale du paramètre *tiret*, si elle est non nulle, indique à partir de quelle distance débuter les tirets.

Si *tiret* vaut 0, le motif pointillé est supprimé.

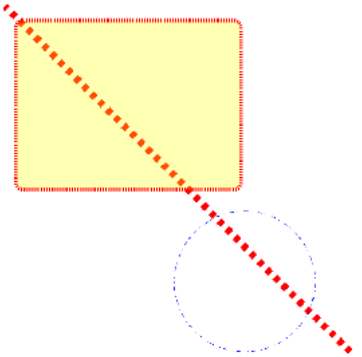
Les paramètres *valeur1* à *valeurN* spécifient en alternance les longueurs des blancs et des tirets qui suivent le premier tiret. Si on fournit un nombre impair de valeurs (premier tiret compris), la liste des valeurs est répétée jusqu'à produire un nombre pair de valeurs.

- Illustrations du tracé de pointillés :

```
//Ligne
$Dom_line:=SVG_New_line ($Dom_SVG;10;10;500;500)
SVG_SET_STROKE_WIDTH ($Dom_line;10)
SVG_SET_STROKE_DASHARRAY ($Dom_line;8,099)
SVG_SET_STROKE_BRUSH ($Dom_line;"red")

//Rectangle
$Dom_rect:=SVG_New_rect ($Dom_SVG;25;30;320;240;10;10;"red";
                                                "yellow:30")
SVG_SET_STROKE_WIDTH ($Dom_rect;5)
SVG_SET_STROKE_DASHARRAY ($Dom_rect;2)
```

```
//Cercle
$Dom_circle:=SVG_New_circle ($Dom_SVG;350;400;100;"blue";"none")
SVG_SET_STROKE_DASHARRAY ($Dom_circle;2;4;6;8)
```



Référence : http://www.yoyodesign.org/doc/w3c/svg1/painting.html#strokedasharrayProperty

SVG_SET_STROKE_MITERLIMIT

SVG_SET_STROKE_MITERLIMIT (objetSVG; jointure)

Paramètre	Type		Description
objetSVG	Ref_SVG	→	Référence d'un élément SVG
jointure	Entier long	→	Valeur de jointure

La commande SVG_SET_STROKE_MITERLIMIT permet de définir l'étendue de la jointure entre le tracé et le liseré de l'objet SVG désigné par *objetSVG*. Une erreur est générée si *objetSVG* n'est pas une référence valide.

Si le paramètre *jointure* est égal à -1, la valeur sera la valeur par défaut (4). Si le paramètre *jointure* est égal à 0, alors la définition de l'attribut est supprimée. Toute autre valeur < 0 provoquera une erreur.

Référence : http://www.yoyodesign.org/doc/w3c/svg1/painting.html#strokemiterlimitProperty

Couleurs et dégradés

SVG_Color_RGB_from_CMYK

SVG_Color_RGB_from_CMYK (cyan ; magenta ; jaune ; noir {; format}) → Résultat

Paramètre	Type		Description
cyan	Entier long	→	Valeur de cyan
magenta	Entier long	→	Valeur de magenta
jaune	Entier long	→	Valeur de jaune
noir	Entier long	→	Valeur de noir
format	Entier long	→	Format de la couleur
Résultat	Texte	←	Chaîne couleur

La commande SVG_Color_RGB_from_CMYK retourne une chaîne exprimant la couleur correspondant aux paramètres quadrichromiques *cyan*, *magenta*, *jaune* et *noir* passés en arguments. La chaîne retournée est par défaut de la forme "RGB(rouge,vert,bleu)", syntaxe reconnue par les moteurs de rendu SVG.

cyan, *magenta*, *jaune* et *noir* sont des entiers longs compris entre 0 et 100%.

Le paramètre optionnel *format* permet de spécifier le format désiré pour la chaîne couleur retournée. Les valeurs sont :

Valeur	Format
1 (défaut)	rgb(r,g,b)
2	#rgb
3	#rrggbb
4	rgb(r%, g%, b%)

SVG_Color_RGB_from_HLS

SVG_Color_RGB_from_HLS (teinte ; luminosité ; saturation {; format}) → Résultat

Paramètre	Type		Description
teinte	Entier long	→	Valeur de la teinte
luminosité	Entier long	→	Valeur de la luminosité
saturation	Entier long	→	Valeur de la saturation
format	Entier long	→	Format de la couleur
Résultat	Texte	←	Chaîne couleur

La commande SVG_Color_RGB_from_HLS retourne une chaîne exprimant la couleur correspondant aux paramètres *teinte*, *luminosité* et *saturation* passés en arguments. La chaîne retournée est par défaut de la forme "RGB(rouge,vert,bleu)", syntaxe reconnue par les moteurs de rendu SVG.

teinte est un entier long compris entre 0 et 360°.

luminosité et *saturation* sont des entiers longs compris entre 0 et 100%.

Le paramètre optionnel *format* permet de spécifier le format désiré pour la chaîne couleur retournée. Les valeurs sont :

Valeur	Format
1 (défaut)	rgb(r,g,b)
2	#rgb
3	#rrggbb
4	rgb(r%, g%, b%)

SVG_GET_COLORS_ARRAY

SVG_GET_COLORS_ARRAY (ptrTabNomsCouls)

Paramètre	Type		Description
ptrTabNomsCouls	Pointeur	→	Pointeur vers le tableau recevant les noms de couleurs

La commande SVG_GET_COLORS_ARRAY remplit le tableau pointé par le paramètre *ptrTabNomsCouls* avec le nom des couleurs reconnues par le SVG.

Référence : http://www.w3.org/TR/SVG/types.html#ColorKeywords

Dessin

SVG_Add_object

SVG_Add_object (objetSVGCible ; objetSVGSource)→ Résultat

Paramètre	Type		Description
objetSVGCible	Ref_SVG	→	Référence de l'élément parent
objetSVGSource	Ref_SVG	→	Référence de l'objet à ajouter
Résultat	Ref_SVG	←	Référence de l'objet SVG

La commande SVG_Add_object permet de placer dans le conteneur SVG désigné par *objetSVGCible* un objet SVG désigné par *objetSVGSource* et retourne sa référence. Le conteneur SVG peut être la racine du document ou toute autre référence à un objet SVG pouvant contenir ce type d'élément.

Structure et Définitions

SVG_Define_clip_path

SVG_Define_clip_path (objetSVGParent ; idRognage) → Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	→	Référence de l'élément parent
idRognage	Texte	→	Nom du tracé de rognage
Résultat	Ref_SVG	←	Référence du tracé de rognage

La commande SVG_Define_clip_path définit un nouveau tracé de rognage dans le conteneur SVG désigné par *objetSVGParent* et retourne sa référence. Si *objetSVGParent* n'est pas (ou n'appartient pas à) un document SVG, une erreur est générée.

Le paramètre *idRognage* désigne le nom du tracé de rognage. Le nom sera utilisé pour associer un tracé de rognage à un objet. Si un élément de même nom existe déjà dans le document, une erreur est générée.

▸ Voir l'exemple de la commande SVG_SET_CLIP_PATH.

Référence : http://www.yoyodesign.org/doc/w3c/svg1/masking.html#EstablishingANewClippingPath

SVG_Define_pattern

SVG_Define_pattern (objetSVGParent ; idMotif{; largeur {; hauteur {; x {; y {; unité {; viewBox}}}}}}) → Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	→	Référence de l'élément parent
idMotif	Texte	→	Nom du motif
largeur	Entier long	→	Largeur du motif
hauteur	Entier long	→	Hauteur du motif
x	Entier long	→	Position x du motif
y	Entier long	→	Position y du motif
unité	Texte	→	Unité des longueurs et positions
viewBox	Texte	→	Rectangle de visualisation
Résultat	Ref_SVG	←	Référence du motif

La commande SVG_Define_pattern définit un nouveau motif personnalisé dans le conteneur SVG désigné par *objetSVGParent* et retourne sa référence. Si *objetSVGParent* n'est pas (ou n'appartient pas à) un document SVG, une erreur est générée.

Le paramètre *idMotif* spécifie le nom du motif. Le nom sera utilisé pour associer le motif à un objet. Si un élément de même nom existait, une erreur est générée.

Les paramètres optionnels *largeur*, *hauteur*, *x*, *y*, *unité* et *viewBox* définissent le rectangle de référence du motif, c'est-à-dire la manière dont la mosaïque de motif sera placée et espacée.

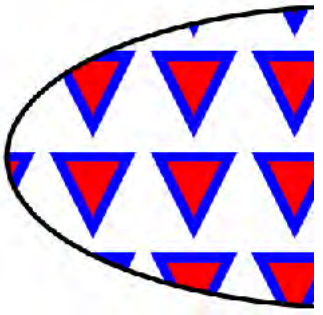
Le motif sera associé comme peinture de remplissage ou de contour en passant la chaîne "url(#id)" comme valeur lorsqu'une expression couleur est attendue.

▸ Définition d'un motif et utilisation pour le remplissage d'une ellipse :

```
//Définition du pattern
$Dom_pattern:=SVG_Define_pattern ($Dom_SVG;"MyPattern";100;100;
                                  0;0;"";"0 0 10 10")
$Dom_path:=SVG_New_path ($Dom_pattern;0;0)

SVG_PATH_MOVE_TO ($Dom_path;0;0)
SVG_PATH_LINE_TO ($Dom_path;7;0)
SVG_PATH_LINE_TO ($Dom_path;3,5;7)
SVG_PATH_CLOSE ($Dom_path)
SVG_SET_FILL_BRUSH ($Dom_path;"red")
SVG_SET_STROKE_BRUSH ($Dom_path;"blue")
```

```
//Tracé d'une ellipse remplie avec le motif
$Dom_ellipse:=SVG_New_ellipse($Dom_SVG;400;200;350;150;"black";
                                        "url(#MyPattern)";5)
```

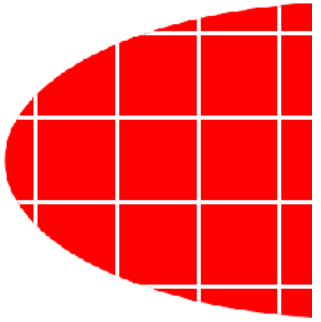


- Définition d'un motif et utilisation pour le remplissage et les contours d'une ellipse :

```
//Définition du motif
$Dom_pattern:=SVG_Define_pattern ($Dom_SVG;"MyPattern ";80;80;0;
                                        0;"";"0 0 20 20")
$Dom_rect:=SVG_New_rect ($Dom_pattern;0;0;20;20;0;0;"white";"red")

//Tracé d'une ellipse
$Dom_ellipse:=SVG_New_ellipse ($Dom_SVG;400;200;350;150)

//Utiliser le motif pour le remplissage et les contours
SVG_SET_FILL_BRUSH ($Dom_ellipse;"url(#MyPattern)")
SVG_SET_STROKE_BRUSH ($Dom_ellipse;"url(#MyPattern)")
```



Référence : http://www.yoyodesign.org/doc/w3c/svg1/pservers.html#Patterns

SVG_Define_style

SVG_Define_style (objetSVGParent ; style { ; type {; media}}) → Résultat

Paramètre	Type		Description
objetSVGParent	Ref_SVG	→	Référence de l'élément parent
style	Texte	→	Définition du style OU Chemin d'accès du fichier à utiliser
type	Texte	→	Type de contenu
media	Texte	→	Descripteur de media
Résultat	Ref_SVG	←	Référence du style

La commande SVG_Define_style définit une nouvelle feuille de style dans le conteneur SVG désigné par *objetSVGParent* et retourne sa référence. Si *objetSVGParent* n'est pas (ou n'appartient pas à) un document SVG, une erreur est générée.

Le paramètre *style* permet l'incorporation de feuilles de styles directement dans un contenu SVG :

- Si le paramètre *style* contient un chemin d'accès valide à un fichier CSS, la définition du style est faite à l'aide du mécanisme de référencement de feuilles de style externes. Le chemin, s'il commence par le caractère # ou la chaîne " file : ", exprime un chemin relatif dont la racine est le dossier "Resources " de la base.
- Le paramètre *style* peut également être un URL du type "http://… ", dans ce cas la feuille de style sera référencée comme ressource externe.

Le paramètre optionnel *type* spécifie le langage de la feuille de style du contenu de l'élément. La valeur par défaut est "text/css".

Le paramètre optionnel *media* indique le media de destination souhaité pour l'information de style. Si vous omettez ce paramètre, la valeur par défaut utilisée est "all". Si la valeur n'est pas comprise dans la liste des types de medias reconnus par CSS2, une erreur est générée.

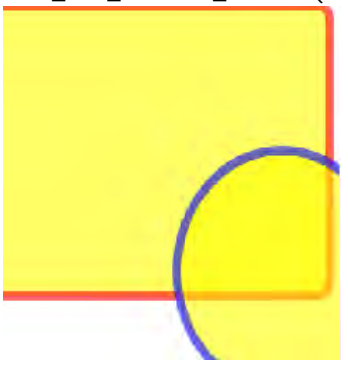
▸ Définition d'un style incorporé et surcharge d'un des éléments :

```
//Définition du style
$Txt_style:=".colored {fill: yellow; fill-opacity: 0.6; stroke: red;
                                    stroke-width:8; stroke-opacity: 0.6}"
SVG_Define_style ($Dom_SVG;$Txt_style)

//Création d'un groupe et affectation d'un style par défaut
$Dom_g:=SVG_New_group ($Dom_SVG)
SVG_SET_CLASS ($Dom_g;"colored")
```

```
//Tracé d'un rectangle
$Dom_rect:=SVG_New_rect ($Dom_g;25;30;320;240;10;10;"";"")

//Tracé d'un cercle et surcharge du style avec une couleur de contour personnalisée
$Dom_circle:=SVG_New_circle ($Dom_g;300;250;100;"";"")
SVG_SET_STROKE_BRUSH ($Dom_circle;"blue")
```



- Référencement du fichier "monstyle.css" placé dans le dossier "dev" du dossier "Resources" :

```
//Définition du style
SVG_Define_style ($Dom_svg;"#dev/monstyle.css")

//Création d'un groupe et affectation d'un style par défaut
$Dom_g:=SVG_New_group ($Dom_SVG)
SVG_SET_CLASS ($Dom_g;"colored")

//Tracé d'un rectangle
$Dom_rect:=SVG_New_rect ($Dom_g;25;30;320;240;10;10;"";"")
```

Fichier monstyle.css :

```
.colored {fill: red; fill-opacity: 0.6; stroke: blue; stroke-width:8; stroke-opacity: 0.6}
```

Référence : http://www.yoyodesign.org/doc/w3c/svg1/styling.html#StyleElement

SVG_DELETE_OBJECT

SVG_DELETE_OBJECT (objetSVG)

Paramètre	Type		Description
objetSVG	Ref_SVG	→	Référence de l'élément à supprimer

La commande SVG_DELETE_OBJECT supprime l'objet SVG désigné par *objetSVG* du document auquel il appartient. Une erreur est générée si *objetSVG* n'est pas une référence valide.

SVG_Get_default_encoding

SVG_Get_default_encoding → Résultat

Paramètre	Type		Description
Résultat	Texte	←	Jeu de caractères

La commande SVG_Get_default_encoding retourne l'encodage utilisé lors de la création d'un nouveau document.

SVG_SET_DEFAULT_ENCODING

SVG_SET_DEFAULT_ENCODING {(encodage)}

Paramètre	Type		Description
encodage	Texte	→	Jeu de caractères

La commande SVG_SET_DEFAULT_ENCODING permet de fixer l'encodage qui sera utilisé lors des prochaines créations de documents.

Si le paramètre *encodage* est omis, la commande rétablit le jeu de caractères par défaut : "UTF-8".

SVG_SET_PATTERN_CONTENT_UNITS

SVG_SET_PATTERN_CONTENT_UNITS(motifObjet ; sysCoordonnées)

Paramètre	Type		Description
motifObjet	Ref_SVG	→	Référence du motif à modifier
sysCoordonnées	Texte	→	Système de coordonnées à utiliser

La commande SVG_SET_PATTERN_CONTENT_UNITS définit le système de coordonnées pour le contenu du motif désigné par *motifObjet*. Si *motifObjet* n'est pas un motif, une erreur est générée.

Le paramètre *sysCoordonnées* spécifie le nom du système à utiliser. Il doit être égal à "userSpaceOnUse" ou "objectBoundingBox" sinon une erreur est générée.

Référence : http://www.yoyodesign.org/doc/w3c/svg1/pservers.html#PatternContentUnitsAttribute

SVG_SET_PATTERN_UNITS

SVG_SET_PATTERN_UNITS (motifObjet ; sysCoordonnées)

Paramètre	Type		Description
motifObjet	Ref_SVG	→	Référence du motif à modifier
sysCoordonnées	Texte	→	Système de coordonnées à utiliser

La commande SVG_SET_PATTERN_UNITS définit le système de coordonnées pour les attributs *x*, *y*, *largeur* et *hauteur* du motif désigné par *motifObjet*. Si *motifObjet* n'est pas un motif, une erreur est générée.

Le paramètre *sysCoordonnées* spécifie le nom du système à utiliser. Il doit être égal à "userSpaceOnUse" ou "objectBoundingBox" sinon une erreur est générée.

Référence : http://www.yoyodesign.org/doc/w3c/svg1/pservers.html#PatternUnitsAttribute

Texte

SVG_APPEND_TEXT_TO_TEXTAREA

SVG_APPEND_TEXT_TO_TEXTAREA (objetSVG ; texteAjout)

Paramètre	Type		Description
objetSVG	Ref_SVG	→	Référence d'un élément texte
texteAjout	Texte	→	Texte à ajouter

La commande SVG_APPEND_TEXT_TO_TEXTAREA permet d'ajouter du texte au contenu textuel de l'objet texte désigné par *objetSVG*. Si *objetSVG* n'est pas un objet "textArea", une erreur est générée.

Les caractères retour à la ligne sont automatiquement remplacés par des éléments "<tbreak/>".

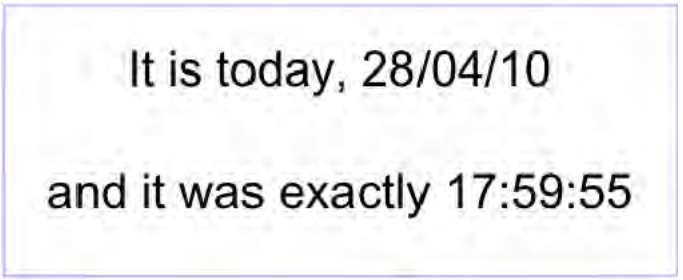
- Ajout de texte

```
//Afficher les contours à l'aide de l'élément 'rect'
$Dom_rect:=SVG_New_rect ( $Dom_SVG;10;10;500;200;0;0;"blue:50";
                                                    "none")

//Créer le texte
$Dom_text:=SVG_New_textArea ($Dom_SVG;"It is today, ";10;30;500;200;
                                                    "'Arial'";36;0;3)

//Ajout de la date et de 2 RC
SVG_APPEND_TEXT_TO_TEXTAREA ($Dom_text; Chaine(Date du jour)+
"\r\r")
```

```
//Enfin, ajout de l'heure courante
SVG_APPEND_TEXT_TO_TEXTAREA ($Dom_text;"and it was exactly "+
                                Chaine(Heure courante))
```



SVG_Get_text

SVG_Get_text (objetSVG) → Résultat

Paramètre	Type		Description
objetSVG	Ref_SVG	→	Référence d'un élément texte
Résultat	Texte	←	Contenu de texte

La commande SVG_Get_text retourne le contenu textuel de l'élément désigné par *objetSVG*. Si *objetSVG* n'est pas une référence d'objet texte ('text', 'textArea' ou 'tspan'), une erreur est générée.

Dans le cas d'un objet textArea, les éléments <tbreak/> sont convertis en CR.

SVG_SET_TEXT_KERNING

SVG_SET_TEXT_KERNING (objetSVG ; crénage {; unité })

Paramètre	Type		Description
objetSVG	Ref_SVG	→	Référence d'un élément texte
crénage	Réel	→	Espacement des lettres
unité	Texte	→	Unité de la valeur de crénage

La commande SVG_SET_TEXT_KERNING permet de modifier l'espacement entre les caractères (le crénage) de l'objet texte désigné par *objetSVG*. Si *objetSVG* n'est pas un objet texte SVG, une erreur est générée.

Le paramètre optionnel *unité* permet de préciser l'unité de la valeur d'espacement. La valeur par défaut est "%".

Si *crénage* vaut -1, la valeur d'espacement est fixée sur 'auto'.

Note Sous Windows, l'implémentation est limitée au texte de gauche à droite et de haut en bas (désactivée pour le texte de droite à gauche) et aux éléments 'text' et 'tspan' ; sous Mac OS, la prise en charge n'est pas limitée.

- Exemples de variation de crénage :

```
//Référence
$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;40;"";36)

$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;80;"";36)
SVG_SET_TEXT_KERNING ($Dom_text;0,5)
$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;120;"";36)
SVG_SET_TEXT_KERNING ($Dom_text;1)
$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;160;"";36)
SVG_SET_TEXT_KERNING ($Dom_text;1,5)
$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;200;"";36)
SVG_SET_TEXT_KERNING ($Dom_text;2)
$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;240;"";36)
SVG_SET_TEXT_KERNING ($Dom_text;1,5)
$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;280;"";36)
SVG_SET_TEXT_KERNING ($Dom_text;1)
$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;320;"";36)
SVG_SET_TEXT_KERNING ($Dom_text;0,5)
$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;360;"";36)
SVG_SET_TEXT_KERNING ($Dom_text;0)
```

Référence : http://www.yoyodesign.org/doc/w3c/svg1/text.html#Kerning-Property

SVG_SET_TEXT_LETTER_SPACING

SVG_SET_TEXT_LETTER_SPACING (objetSVG; espacement {; unité})

Paramètre	Type		Description
objetSVG	Ref_SVG	→	Référence d'un élément texte
espacement	Réel	→	Espacement des lettres
unité	Texte	→	Unité de la valeur d'espacement

La commande SVG_SET_TEXT_LETTER_SPACING permet de modifier l'espacement des lettres de l'objet texte désigné par *objetSVG* en plus de tout espacement dû à la propriété 'kerning'. Si *objetSVG* n'est pas un objet texte SVG, une erreur est générée.

Le paramètre optionnel *unité* permet de préciser l'unité de la valeur d'espacement. La valeur par défaut est "%".

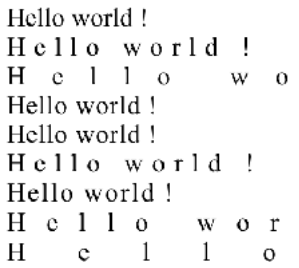
Si *espacement* vaut -1, la valeur d'espacement est fixée sur 'normal'.

- Exemples de variation d'espacement :

```
//Référence
$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;40;"";36)

$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;80;"";36)
SVG_SET_TEXT_LETTER_SPACING ($Dom_text;1)
$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;120;"";36)
SVG_SET_TEXT_LETTER_SPACING ($Dom_text;1;"em")
$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;160;"";36)
SVG_SET_TEXT_LETTER_SPACING ($Dom_text;1;"px")
$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;200;"";36)
SVG_SET_TEXT_LETTER_SPACING ($Dom_text;1;"pt")
$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;240;"";36)
SVG_SET_TEXT_LETTER_SPACING ($Dom_text;1;"pc")
$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;280;"";36)
SVG_SET_TEXT_LETTER_SPACING ($Dom_text;1;"mm")
$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;320;"";36)
SVG_SET_TEXT_LETTER_SPACING ($Dom_text;1;"cm")
$Dom_text:=SVG_New_text ($Dom_SVG;"Hello world !";20;360;"";36)
```

SVG_SET_TEXT_LETTER_SPACING ($Dom_text;1;"in")



Référence : http://www.yoyodesign.org/doc/w3c/svg1/text.html#LetterSpacingProperty

SVG_SET_TEXT_RENDERING

SVG_SET_TEXT_RENDERING (objetSVG ; rendu)

Paramètre	Type		Description
objetSVG	Ref_SVG	→	Référence d'un élément texte
rendu	Texte	→	Valeur de rendu

La commande SVG_SET_TEXT_RENDERING permet de définir les compromis à utiliser pour le rendu du texte de l'objet texte désigné par *objetSVG*. Si *objetSVG* n'est pas un objet texte SVG, une erreur est générée.

Le paramètre *rendu* peut prendre l'une des valeurs suivantes : "auto", "optimizeSpeed", "optimizeLegibility", "geometricPrecision" ou "inherit". Dans le cas contraire, une erreur est générée.

Référence : http://www.yoyodesign.org/doc/w3c/svg1/painting.html#TextRenderingProperty

SVG_SET_TEXT_WRITING_MODE

SVG_SET_TEXT_WRITING_MODE (objetSVG ; modeEcriture)

Paramètre	Type		Description
objetSVG	Ref_SVG	→	Référence d'un élément texte
modeEcriture	Texte	→	Direction de l'écriture

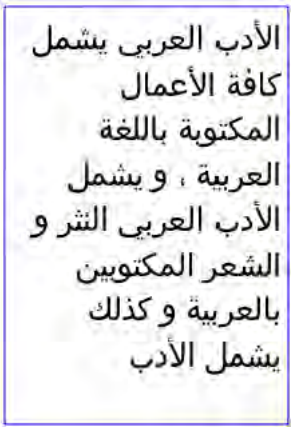
La commande SVG_SET_TEXT_WRITING_MODE permet de définir si la direction d'écriture de l'objet texte désigné par *objetSVG* sera de gauche à droite, de droite à gauche ou de bas en haut. Si *objetSVG* n'est pas un objet texte SVG, une erreur est générée.

Le paramètre *modeEcriture* peut prendre une des valeurs suivantes : "lr-tb", "rl-tb", "tb-rl", "lr", "rl", "tb" ou "inherit ". Dans le cas contraire, une erreur est générée.

- Ecriture de droite à gauche :

```
//Cadre
SVG_New_rect ($Dom_SVG;5;5;210;310;0;0;"blue";"none")

//Texte
$Dom_text:=SVG_New_textArea ($Dom_SVG;$Txt_sample;10;10;200;
            300;"Baghdad 'Arial Unicode MS'";25)
SVG_SET_TEXT_WRITING_MODE($Dom_text;"rl")
```



SVG_SET_TEXTAREA_TEXT

SVG_SET_TEXTAREA_TEXT (objetSVG ; leTexte)

Paramètre	Type		Description
objetSVG	Ref_SVG	→	Référence d'un élément texte
leTexte	Texte	→	Texte à définir

La commande SVG_SET_TEXTAREA_TEXT permet de fixer/remplacer la contenu textuel de l'objet texte désigné par *objetSVG*. Si *objetSVG* n'est pas un objet "textArea", une erreur est générée.

Les caractères retour à la ligne sont automatiquement remplacés par des éléments "<tbreak/>".

Commandes modifiées

SVG_New_textArea

La commande SVG_New_textArea remplace désormais les retours à la ligne par des éléments <tbreak/>.

Référence : http://www.w3.org/TR/SVGTiny12/text.html#tbreakElement

SVG_SET_FONT_FAMILY SVG_SET_FONT_FAMILY (objetSVG ; *police1{ ;...; policeN}*)

Paramètre	Type		Description
objetSVG	Ref_SVG	→	Référence d'un élément SVG
police1...N	Texte	→	Nom(s) de police(s)

La commande SVG_SET_FONT_FAMILY accepte désormais plusieurs noms de polices en paramètre. Lorsque plusieurs noms sont passés, la commande construit automatiquement la liste des polices et/ou des familles génériques.

- Passage de plusieurs noms de police :

```
SVG_SET_FONT_FAMILY(objetSVG ; "Lucida grande" ; "Sans-serif")
// construira la liste : " 'Lucida grande' 'Sans-serif'"
```

Utilitaires

SVG_ABOUT SVG_ABOUT

Paramètre	Type	Description
Cette commande ne requiert pas de paramètres		

La commande SVG_ABOUT affiche un dialogue avec le logo 4D SVG et indiquant le numéro de version du composant :

SVGTool_SET_VIEWER_CALLBACK

SVGTool_SET_VIEWER_CALLBACK (nomMéthode4D)

Paramètre	Type		Description
nomMéthode4D	Texte	→	Nom de méthode 4D

La commande SVGTool_SET_VIEWER_CALLBACK permet d'installer *nomMéthode4D* comme méthode projet appelée lors des événements Sur clic et Sur survol survenant sur l'image affichée par la commande SVGTool_SHOW_IN_VIEWER.

Cette méthode reçoit un paramètre texte qui est l'id de l'élément survolé ou cliqué tel que fourni par la commande 4D SVG Chercher ID element par coordonnees. Ce paramètre doit impérativement être déclaré dans la méthode projet *nomMéthode4D* de la base hôte en y insérant la ligne C_TEXTE($1).

Vous devez affecter la propriété "Partagée entre composants et base hôte" à la méthode *nomMéthode4D*.

Si la méthode n'existe pas ou n'est pas partagée, l'erreur -10508 est générée par 4D.

Commande modifiée

SVG_References_array

SVG_References_array (pointeurTabRef) → *Résultat*

Paramètre	Type		Description
pointeurTabRef	Pointeur	→	Tableau chaîne des références de documents
Résultat	*Entier long*	←	*Nombre de références*

La commande SVG_References_array retourne désormais le nombre de références trouvées.

- Boucle sur le nombre d'éléments :

```
TABLEAU TEXTE ( MonTableau ; 0 )
Boucle ($i ; 1 ;  SVG_References_array ( ->MonTableau ))
  …
Fin de boucle
```

6 SQL

Cette section présente les nouveautés et modifications apportées au moteur SQL de 4D v12 :

- Nouvelle fonction de réplication de bases via le SQL,
- Définition directe d'une clé primaire,
- Prise en charge des jointures externes (OUTER JOINS),
- Utilisation de bases externes,
- Prise en charge des champs UUID,
- Nouvelles commandes et commandes modifiées.

Note L'utilisation des nouvelles instructions SQL dans l'éditeur de code est facilité via des macros-commandes spécifiques. Pour pouvoir en bénéficier, vous devez mettre à jour votre fichier "Macros.xml". Pour plus d'informations sur ce point, reportez-vous au paragraphe "Mise à jour du fichier Macros.xml", page 17.

Réplication via le SQL

4D v12 propose un nouveau mécanisme permettant de synchroniser les données de deux ou plusieurs bases 4D via le SQL. Ce mécanisme autorise la mise en place d'une ou plusieurs base(s) miroir, garantissant la disponibilité permanente des données.

Le principe est le suivant : une base de données cible réplique en local les données d'une base de données source distante. La mise à jour s'effectue périodiquement par la base locale qui va chercher les données sur la base distante. La réplication est effectuée au niveau de la table : vous répliquez les données d'une table de la base distante dans une table de la base locale.

Ce principe est rendu possible par l'ajout de marqueurs et de nouvelles commandes SQL.
Une nouvelle propriété de table permet d'activer le mécanisme de réplication dans la base distante et la base locale. Côté base locale, la nouvelle commande SQL REPLICATE permet de rapatrier les données d'une table de la base distante puis de les intégrer dans une table de la base locale. Enfin, la nouvelle commande SQL SYNCHRONIZE permet d'effectuer la synchronisation des deux tables.

Nouveaux champs virtuels

Chaque table de la base 4D se voit attribuer trois nouveaux champs "virtuels" : __ROW_ID, __ROW_STAMP et __ROW_ACTION. Ces champs sont appelés virtuels pour les différencier des champs "classiques", car ils disposent de propriétés spécifiques : ils sont renseignés automatiquement, peuvent être lus mais non modifiés par les utilisateurs et n'apparaissent pas dans les tables système de la base. Le tableau suivant décrit ces champs ainsi que leur mode d'utilisation :

Champ virtuel	Type	Contenu	Utilisation
__ROW_ID	Int32	ID d'enregistrement	Dans toute instruction SQL sauf REPLICATE ou SYNCHRONIZE
__ROW_STAMP	Int64	Informations de réplication de l'enregistrement	Dans toute instruction SQL
__ROW_ACTION	Int16	Type d'action effectuée sur l'enregistrement : 1 = Ajout ou modification 2 = Suppression	Uniquement avec la commande REPLICATE ou SYNCHRONIZE

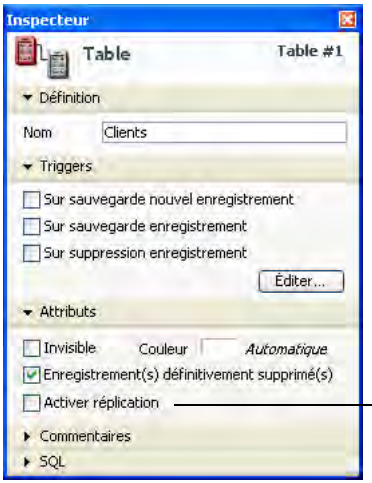
Lorsque les mécanismes de réplication sont activés, dès qu'un enregistrement est créé, modifié ou supprimé, les informations correspondantes sont automatiquement mises à jour dans les champs virtuels de cet enregistrement.

Activation de la réplication

Par défaut les mécanismes permettant la réplication ne sont pas activés. Vous devez les activer explicitement côté base distante et côté base locale pour chaque table utilisée dans la réplication ou synchronisation.

A noter que l'activation ne déclenche pas la réplication ; pour que les données soient effectivement répliquées dans la base locale ou synchronisées, vous devez utiliser les nouvelles commandes REPLICATE ou SYNCHRONIZE.

Pour activer le mécanisme interne de réplication dans chaque table (côté base distante et côté base locale), vous devez utiliser la nouvelle propriété de table **Activer réplication**, accessible dans l'Inspecteur des tables :



Pour que le mécanisme de réplication puisse fonctionner, une clé primaire doit avoir été définie dans les tables impliquées côté base distante et côté base locale. Vous pouvez effectuer cette création via l'éditeur de structure (cf. paragraphe "Définir la clé primaire", page 24***) ou les commandes SQL (cf.*** paragraphe "Définir une clé primaire en création de colonne", page 300***).***

Lorsque cette option est cochée, 4D génère les informations nécessaires au mécanisme de réplication des enregistrements de la table (basées notamment sur la clé primaire de la table). Ces informations sont stockées dans les champs virtuels __ROW_STAMP et __ROW_ACTION.

Note Il est possible d'activer et d'inactiver la génération des informations de réplication via les commandes SQL CREATE TABLE et ALTER TABLE. Pour cela, les nouveaux mots-clés ENABLE REPLICATE et DISABLE REPLICATE sont disponibles. Pour plus d'informations, reportez-vous à la description de ces commandes dans le paragraphe "Commandes SQL modifiées", page 321.

Mise à jour côté base locale

Une fois que le mécanisme de réplication est activé dans chaque table de chaque base, vous pouvez l'exploiter depuis la base locale à l'aide de la nouvelle commande SQL REPLICATE. Cette commande est décrite dans le paragraphe "Nouvelles commandes SQL", page 311.

Définir une clé primaire en création de colonne

Vous pouvez désormais définir une clé primaire (PRIMARY KEY) directement lors de la création d'une table (via la commande CREATE TABLE) ou de l'ajout d'une colonne (via ALTER TABLE). Dans les versions précédentes de 4D, il était nécessaire d'utiliser la commande ALTER TABLE avec une colonne existante.

Les syntaxes de ces commandes ont donc été modifiées, le mot-clé PRIMARY KEY est désormais accepté lors de la création de la colonne.

Pour plus d'informations, reportez-vous à la description de ces commandes dans le paragraphe paragraphe "Commandes SQL modifiées", page 321.

Note 4D vous permet également de gérer les clés primaires directement dans l'éditeur de structure. Pour plus d'informations, reportez-vous au paragraphe "Définir la clé primaire", page 24.

Prise en charge des jointures

Le moteur SQL de 4D v12 étend la prise en charge des jointures et permet en particulier d'effectuer des jointures externes (*OUTER joins*).

Les jointures peuvent être internes ou externes, implicites ou explicites. Les jointures internes (*INNER joins*) implicites, effectuées via la commande SELECT, étaient déjà prises en charge par 4D v11 SQL (cf. ci-dessous). 4D v12 permet désormais de générer des jointures internes et externes explicites à l'aide du mot-clé SQL **JOIN**.

Note L'implémentation actuelle des jointures dans le moteur SQL de 4D n'inclut pas :
- les jointures naturelles.
- le constructeur USING dans les jointures internes.

Présentation

Les jointures permettent de mettre en relation les enregistrements de deux ou plusieurs tables et de combiner le résultat dans une nouvelle table, appelée jointure.

Vous générez des jointures via des instructions SELECT définissant des conditions de jointure. Avec les jointures explicites, les conditions peuvent être complexes mais elles doivent toujours être basées sur un test d'égalité entre des colonnes incluses dans la jointure. Par exemple, il n'est pas possible d'utiliser l'opérateur >= dans une condition de jointure explicite.
Les jointures implictes autorisent tout type de comparaison.
En interne, les comparaisons d'égalité sont effectuées directement par le moteur de 4D, ce qui leur assure une grande rapidité d'exécution.

Note Généralement, dans le moteur de base de données, l'ordre des tables est déterminé par l'ordre défini dans la recherche. Lorsque vous utilisez des jointures, l'ordre des tables est déterminé par la liste des tables. Dans l'exemple suivant :
SELECT * FROM T1 RIGHT OUTER JOIN T2 ON T2.depID = T1.depID;
... l'ordre des tables est T1 puis T2 (comme elles apparaissent dans la liste des tables) et non T2 puis T1 (comme elles apparaissent dans la condition de jointure).

Jointures internes explicites

Une jointure interne (*inner join*) est une jointure basée sur une comparaison d'égalité entre deux colonnes. Considérons par exemple les trois tables suivantes :

- **Employees**

name	depID	cityID
Alain	10	30
Anne	11	39
Bernard	10	33
Fabrice	12	35
Martine	15	30
Philippe	NULL	33
Thomas	10	NULL

- **Departments**

depID	depName
10	Program
11	Engineering
NULL	Marketing
12	Development
13	Quality

- **Cities**

cityID	cityName
30	Paris
33	New York
NULL	Berlin

Note Cette structure d'exemple sera utilisée tout au long de ce chapitre.

Pour rappel, voici un exemple de jointure interne implicite (déjà pris en charge dans 4D v11 SQL) :

```
SELECT *
   FROM employees, departments
   WHERE employees.DepID = departments.DepID;
```

Dans 4D v12, vous pouvez désormais utiliser le mot-clé JOIN afin de définir une jointure interne explicite :

```
SELECT *
   FROM employees
   INNER JOIN departments
      ON employees.DepID = departments.DepID;
```

Cette requête peut être insérée dans le code 4D de la manière suivante :

```
TABLEAU TEXTE (aName;0)
TABLEAU TEXTE (aDepName;0)
TABLEAU ENTIER (aEmpDepID;0
TABLEAU ENTIER (aDepID;0)
Debut SQL
  SELECT *
  FROM employees
       INNER JOIN departments
         ON employees.depID = departments.depID
       INTO :aName, :aEmpDepID, :aDepID, :aDepName;
Fin SQL
```

Résultat de cette jointure :

aName	*aEmpDepID*	*aDepID*	*aDepName*
Alain	*10*	*10*	*Program*
Anne	*11*	*11*	*Engineering*
Bernard	*10*	*10*	*Program*
Fabrice	*12*	*12*	*Development*
Thomas	*10*	*10*	*Program*

A noter que ni les employés Philippe et Martine ni les départements Marketing et Quality n'apparaissent dans la jointure résultante car :

- Philippe n'a pas de département associé (NULL),
- l'ID de département de Martine n'existe pas dans la table Departments,
- il n'y a pas d'employé associé au département d'ID 13 (Quality),
- le département Marketing n'a pas d'ID associé (NULL).

Jointures croisées (CROSS)

Une jointure interne dans laquelle la clause WHERE et la clause ON sont omises est appelée jointure *croisée* ou cartésienne. Effectuer une jointure croisée consiste à associer chaque ligne d'une table à chaque ligne d'une autre table.
Le résultat d'une jointure croisée est le produit cartésien des tables, contenant *m* x *n* lignes, où *m* est le nombre de lignes de la première table et *n* est le nombre de lignes de la seconde table. Ce produit représente l'ensemble des combinaisons possibles formées par la concaténation des lignes des tables.

Les syntaxes suivantes sont équivalentes :

```
SELECT * FROM T1 INNER JOIN T2
```

```
SELECT * FROM T1, T2
```

```
SELECT * FROM T1 CROSS JOIN T2;
```

Voici un exemple de code 4D intégrant une jointure croisée :

```
TABLEAU TEXTE (aName;0)
TABLEAU TEXTE (aDepName;0)
TABLEAU ENTIER (aEmpDepID;0
TABLEAU ENTIER (aDepID;0)
Debut SQL
  SELECT *
  FROM employees CROSS JOIN departments
       INTO :aName, :aEmpDepID, :aDepID, :aDepName;
Fin SQL
```

Résultat de cette jointure avec notre base d'exemple :

aName	*aEmpDepID*	*aDepID*	*aDepName*
Alain	*10*	*10*	*Program*
Anne	*11*	*10*	*Program*
Bernard	*10*	*10*	*Program*
Fabrice	*12*	*10*	*Program*
Martine	*15*	*10*	*Program*
Philippe	*NULL*	*10*	*Program*
Thomas	*10*	*10*	*Program*
Alain	*10*	*11*	*Engineering*
Anne	*11*	*11*	*Engineering*
Bernard	*10*	*11*	*Engineering*
Fabrice	*12*	*11*	*Engineering*
Martine	*15*	*11*	*Engineering*
Philippe	*NULL*	*11*	*Engineering*
Thomas	*10*	*11*	*Engineering*
Alain	*10*	*NULL*	*Marketing*
Anne	*11*	*NULL*	*Marketing*
Bernard	*10*	*NULL*	*Marketing*
Fabrice	*12*	*NULL*	*Marketing*
Martine	*15*	*NULL*	*Marketing*
Philippe	*NULL*	*NULL*	*Marketing*
Thomas	*10*	*NULL*	*Marketing*
Alain	*10*	*12*	*Development*
Anne	*11*	*12*	*Development*
Bernard	*10*	*12*	*Development*
Fabrice	*12*	*12*	*Development*
Martine	*15*	*12*	*Development*
Philippe	*NULL*	*12*	*Development*
Thomas	*10*	*12*	*Development*
Alain	*10*	*13*	*Quality*
Anne	*11*	*13*	*Quality*
Bernard	*10*	*13*	*Quality*
Fabrice	*12*	*13*	*Quality*
Martine	*15*	*13*	*Quality*
Philippe	*NULL*	*13*	*Quality*
Thomas	*10*	*13*	*Quality*

Note Pour des raisons de performances, les jointures croisées sont à utiliser avec précaution.

Jointures externes

4D v12 vous permet désormais de générer des jointures externes (*OUTER JOINs*). Dans une jointure externe, il n'est pas nécessaire qu'il existe une correspondance entre les lignes des tables jointes. La table résultante contient toutes les lignes des tables (ou d'au moins une des tables de la jointure) même s'il n'y a pas de ligne correspondante. Ce principe permet de s'assurer que toutes les informations d'une table sont exploitées, même si des lignes ne sont pas renseignées entre les différentes tables jointes.

Il existe trois types de jointures externes, définies par les mots-clés LEFT, RIGHT et FULL. LEFT et RIGHT permettent de désigner la table (située à gauche ou à droite du mot-clé) dont la totalité des données devra être traitée. FULL indique une jointure externe bilatérale.

Note Seules les jointures externes explicites sont prises en charge par 4D v12.

Jointures externes gauches (LEFT OUTER JOIN)

Le résultat d'une jointure externe gauche (ou jointure gauche) contient toujours tous les enregistrements de la table située à gauche du mot-clé même si la condition de jointure ne trouve pas d'enregistrement correspondant dans la table de droite. Cela signifie que si pour une ligne de la table gauche la requête trouve zéro ligne correspondant dans la table droite, la jointure contiendra la ligne avec la valeur NULL pour chaque colonne de la table de droite. Autrement dit, une jointure externe gauche retourne toutes les lignes de la table gauche *plus* celles de la table droite qui correspondent à la condition de jointure (ou NULL si aucune ne correspond). A noter que si la table la droite contient plus d'une ligne correspondant au prédicat de la jointure pour une ligne de la table gauche, les valeurs de la table gauche seront répétées pour chaque ligne distincte de la table droite.

Voici un exemple de code 4D effectuant une jointure externe gauche :

```
TABLEAU TEXTE (aName;0)
TABLEAU TEXTE (aDepName;0)
TABLEAU ENTIER (aEmpDepID;0
TABLEAU ENTIER (aDepID;0)
Debut SQL
  SELECT *
    FROM employees
    LEFT OUTER JOIN departments
      ON employees.DepID = departments.DepID;
      INTO :aName, :aEmpDepID, :aDepID, :aDepName;
Fin SQL
```

Résultat de cette jointure avec notre base d'exemple (les lignes additionnelles sont en rouge) :

aName	*aEmpDepID*	*aDepID*	*aDepName*
Alain	*10*	*10*	*Program*
Anne	*11*	*11*	*Engineering*
Bernard	*10*	*10*	*Program*
Fabrice	*12*	*12*	*Development*
Thomas	*10*	*10*	*Program*
Martine	*15*	*NULL*	*NULL*
Philippe	*NULL*	*NULL*	*NULL*

Jointures externes droites (RIGHT OUTER JOIN)

A l'exact opposé de la jointure externe gauche (cf. paragraphe ci-dessus), le résultat d'une jointure externe droite contient toujours tous les enregistrements de la table située à droite du mot-clé même si la condition de jointure ne trouve pas d'enregistrement correspondant dans la table gauche.

Voici un exemple de code 4D effectuant une jointure externe droite :

```
TABLEAU TEXTE (aName;0)
TABLEAU TEXTE (aDepName;0)
TABLEAU ENTIER (aEmpDepID;0
TABLEAU ENTIER (aDepID;0)
Debut SQL
  SELECT *
    FROM employees
    RIGHT OUTER JOIN departments
      ON employees.DepID = departments.DepID;
      INTO :aName, :aEmpDepID, :aDepID, :aDepName;
Fin SQL
```

Résultat de cette jointure avec notre base d'exemple (les lignes additionnelles sont en rouge) :

aName	*aEmpDepID*	*aDepID*	*aDepName*
Alain	*10*	*10*	*Program*
Anne	*11*	*11*	*Engineering*
Bernard	*10*	*10*	*Program*
Fabrice	*12*	*12*	*Development*
Thomas	*10*	*10*	*Program*
NULL	*NULL*	*NULL*	*Marketing*
NULL	*NULL*	*13*	*Quality*

Jointures externes bilatérales (FULL OUTER JOIN)

La jointure externe bilatérale combine simplement les résultats d'une jointure externe gauche et d'une jointure externe droite. La table jointure résultante contient tous les enregistrements des tables gauche et droite et remplit les champs manquants de chaque côté avec des valeurs NULL.

Voici un exemple de code 4D effectuant une jointure externe bilatérale :

```
TABLEAU TEXTE (aName;0)
TABLEAU TEXTE (aDepName;0)
TABLEAU ENTIER (aEmpDepID;0
TABLEAU ENTIER (aDepID;0)
Debut SQL
  SELECT *
    FROM employees
    FULL OUTER JOIN departments
      ON employees.DepID = departments.DepID;
      INTO :aName, :aEmpDepID, :aDepID, :aDepName;
Fin SQL
```

Résultat de cette jointure avec notre base d'exemple (les lignes additionnelles sont en rouge) :

aName	*aEmpDepID*	*aDepID*	*aDepName*
Alain	*10*	*10*	*Program*
Anne	*11*	*11*	*Engineering*
Bernard	*10*	*10*	*Program*
Fabrice	*12*	*12*	*Development*
Thomas	*10*	*10*	*Program*
Martine	*15*	*NULL*	*NULL*
Philippe	*NULL*	*NULL*	*NULL*
NULL	*NULL*	*NULL*	*Marketing*
NULL	*NULL*	*13*	*Quality*

Jointures multiples dans une seule requête

Il est possible de combiner plusieurs jointures dans une même instruction SELECT. Il est également possible de mixer des jointures internes implicites ou explicites et des jointures externes explicites.

Voici un exemple de code 4D effectuant des jointures multiples :

```
TABLEAU TEXTE (aName;0)
TABLEAU TEXTE (aDepName;0)
TABLEAU TEXTE (aCityName;0)
TABLEAU ENTIER (aEmpDepID;0
TABLEAU ENTIER (aEmpCityID;0)
TABLEAU ENTIER (aDepID;0)
TABLEAU ENTIER (aCityID;0)
Debut SQL
  SELECT *
    FROM employees
    FULL OUTER JOIN departments
      ON employees.DepID = departments.DepID;
      INTO :aName, :aEmpDepID, :aDepID, :aDepName;
    FROM (employees RIGHT OUTER JOIN departments
          ON employees.depID = departments.depID)
          LEFT OUTER JOIN cities
          ON employees.cityID = cities.cityID
    INTO :aName, :aEmpDepID, :aEmpCityID, :aDepID, :aDepName;
                                          :aCityID, :aCityName;
Fin SQL
```

Résultat de cette jointure avec notre base d'exemple :

aName	*aEmpDepID*	*aEmpCityID*	*aDepID*	*aDepName*	*aCityID*	*aCityName*
Alain	*10*	*30*	*10*	*Program*	*30*	*Paris*
Anne	*11*	*39*	*11*	*Engineering*	*0*	
Bernard	*10*	*33*	*10*	*Program*	*33*	*New York*
Fabrice	*12*	*35*	*12*	*Development*	*0*	
Thomas	*10*	*NULL*	*10*	*Program*	*0*	

Utiliser des bases externes

4D v12 vous permet de créer, modifier et utiliser des "bases externes" via le langage SQL.

Une base externe est une base 4D indépendante de la base 4D principale, mais que vous pouvez manipuler depuis la base 4D principale. Utiliser une base externe signifie désigner temporairement cette base comme base courante, c'est-à-dire comme base cible des requêtes SQL exécutées par 4D. Par défaut, la base courante est la base principale.

Vous pouvez créer directement une base externe depuis la base principale avec la nouvelle commande SQL CREATE DATABASE. La base est stockée sur disque dans des fichiers standard (fichiers .4db et .4dd). Vous pouvez créer autant de bases externes que vous voulez depuis la base 4D principale.
Une fois créée, une base externe peut être désignée comme base courante à l'aide de la nouvelle commande SQL USE DATABASE. Elle peut alors être modifiée via les commandes SQL standard (CREATE TABLE, ALTER TABLE, etc.) et vous pouvez y stocker des données. La nouvelle fonction DATABASE_PATH permet de connaître à tout moment la base courante.

L'intérêt majeur des bases externes réside dans le fait qu'elles peuvent être créées et manipulées via des composants 4D. Cette nouveauté permet de développer des composants pouvant créer des tables et des champs en fonction de leurs besoins.

Note Une base externe est une base 4D standard. Elle peut être ouverte et manipulée en tant que base principale par une application 4D ou 4D Server. Inversement, toute base 4D standard peut être utilisée comme base externe. Toutefois, il est impératif de ne pas activer le système de gestion des accès (via l'affectation d'un mot de passe au *Super_Utilisateur*) dans une base externe sinon elle ne sera plus accessible via la commande SQL USE DATABASE.

Prise en charge des champs UUID

4D v12 permet de générer et de stocker des numéros UUID (cf. paragraphe “Prise en charge des UUID”, page 19).

Créer un champ UUID

Dans l’éditeur de structure, un champ UUID est défini en assignant une propriété spécifique à champ Alpha (cf. paragraphe “Format UUID”, page 21).
Côté 4D SQL, un champ UUID est identifié via un nouveau type de données : **UUID**. Ce type de données peut être utilisé avec les commandes de création ou de modification de champs. Par exemple :

```
Debut SQL
  CREATE TABLE ACTORS_FAN
    (ID      UUID,
    Nom    VARCHAR(30));
Fin SQL
```

Ou bien :

```
Debut SQL
  CREATE TABLE ACTORS_FAN
    (Nom   VARCHAR(30));
  ALTER TABLE ACTORS_FAN
    ADD ID UUID;
Fin SQL
```

Générer des UUID automatiquement

Dans 4D, il est possible de générer automatiquement des numéros UUID dans les enregistrements d’un champ UUID grâce à la propriété "UUID Auto" (cf. paragraphe “UUID auto”, page 22).
Côté 4D SQL, cette propriété est prise en charge via un nouveau mot-clé : **AUTO_GENERATE**. Ce mot-clé peut être utilisé avec les commandes de création ou de modification de champs. Par exemple :

```
Debut SQL
  CREATE TABLE ACTORS_FAN
    (ID      UUID AUTO_GENERATE,
    Nom    VARCHAR(30));
Fin SQL
```

Nouvelles commandes SQL

Plusieurs nouvelles commandes SQL sont disponibles dans 4D v12 :

- les commandes liées à la gestion des bases externes : CREATE DATABASE, USE DATABASE,
- les commandes liées aux nouvelles implémentations : ALTER DATABASE, REPLICATE et SYNCHRONIZE.

CREATE DATABASE

CREATE DATABASE [IF NOT EXISTS] DATAFILE <Chemin d'accès complet>

La nouvelle commande CREATE DATABASE vous permet de créer une nouvelle base de données externe (fichiers .4db et .4dd) à un emplacement spécifique (cf. paragraphe "Utiliser des bases externes", page 309).

Si la contrainte **IF NOT EXISTS** est passée, la base de données n'est pas créée et aucune erreur n'est générée si une base du même nom existe déjà à l'emplacement défini.
Si la contrainte **IF NOT EXISTS** n'est pas passée, la base de données n'est pas créée et le message d'erreur "Cette base de données existe déjà. Echec de la commande CREATE DATABASE" est affiché si une base du même nom existe déjà à l'emplacement défini.

La clause **DATAFILE** vous permet de définir le nom complet (chemin d'accès complet + nom) de la nouvelle base de données externe. Vous devez passer le nom du fichier de structure. Le programme ajoute automatiquement l'extension ".4db" au fichier si elle n'est pas déjà définie et crée le fichier de données. Le chemin d'accès peut être exprimé soit en syntaxe POSIX, soit en syntaxe système. Il peut être absolu ou relatif au fichier de structure de la base 4D principale.

- syntaxe POSIX (type URL) : les noms de dossiers sont séparés par une barre oblique ("/"), quelle que soit la plate-forme que vous utilisez, par exemple : ../basesexternes/mabase.4db
- syntaxe système : chemin d'accès respectant la syntaxe de la plate-forme courante, par exemple :
  - (Mac OS) Disque:Applications:monserv:basesexternes:mabase.4db
  - (Windows) C:\Applications\monserv\basesexternes\mabase.4db

Après l'exécution réussie de la commande CREATE DATABASE, la nouvelle base de données créée ne devient pas automatiquement la base courante. Pour cela, vous devez explicitement la déclarer en tant que base courante à l'aide de la nouvelle commande USE DATABASE.

- Création des fichiers de base externe *ExternalDB.4DB* et *ExternalDB.4DD* à l'emplacement C:/MaBase/ :

```
Debut SQL
  CREATE DATABASE IF NOT EXISTS DATAFILE 'C:/MaBase/ExternalDB';
Fin SQL
```

- Création des fichiers de base externe *TestDB.4DB* et *TestDB.4DD* à côté du fichier de structure de la base principale :

```
Debut SQL
  CREATE DATABASE IF NOT EXISTS DATAFILE 'TestDB';
Fin SQL
```

- Création des fichiers de base externe *External.4DB* et *External.4DD* à l'emplacement défini par l'utilisateur :

```
C_TEXTE($chemin)
$chemin:=Selectionner dossier("Dossier de destination de la base externe :")
$chemin:= $chemin+"External"
Debut SQL
  CREATE DATABASE DATAFILE <<$chemin>>;
Fin SQL
```

Référence : USE DATABASE

USE DATABASE

USE [LOCAL | REMOTE] DATABASE
{DATAFILE <Chemin d'accès complet> | **SQL_INTERNAL** | **DEFAULT}**
[AUTO_CLOSE]

La nouvelle commande USE DATABASE vous permet de désigner une base externe (cf. paragraphe "Utiliser des bases externes", page 309), comme base de données courante, c'est-à-dire vers laquelle seront dirigées les prochaines requêtes SQL dans le process courant. Tous les types de requêtes SQL sont concernés : requêtes incluses dans une structure Debut SQL/Fin SQL, commandes SQL EXECUTER ou SQL EXECUTER SCRIPT, etc.

Si vous travaillez en configuration monoposte, la base externe doit être située sur votre machine 4D.
Si vous travaillez en mode distant, la base externe peut être située sur le poste local ou sur la machine 4D Server.

Si vous utilisez 4D en mode distant, le mot-clé **REMOTE** vous permet de désigner une base externe située sur 4D Server.
Pour des raisons de sécurité, ce mécanisme fonctionne uniquement avec les connexions distantes natives, c'est-à-dire dans le contexte d'une base 4D distante connectée à 4D Server. Les connexions via ODBC ou *pass-through* ne sont pas autorisées.
Si aucun mot-clé n'est spécifié, l'option **LOCAL** est utilisée par défaut.
Si vous utilisez 4D en mode local, les mot-clés **REMOTE** et **LOCAL** sont ignorés : les connexions sont toujours locales.

Pour désigner la base externe à utiliser, passez son chemin complet (chemin d'accès + nom) dans la clause **DATAFILE**. Le chemin d'accès peut être exprimé soit en syntaxe POSIX, soit en syntaxe système. Il peut être absolu ou relatif au fichier de structure de la base 4D principale.
En mode distant, si le mot-clé **REMOTE** est passé, ce paramètre désigne le chemin d'accès de la base à partir du poste serveur. S'il est omis ou si le mot-clé **LOCAL** est passé, ce paramètre désigne le chemin d'accès de la base sur le poste 4D local.

Vous devez désigner une base 4D externe valide et dans laquelle le système de contrôle des accès n'a pas été activé (via l'attribution d'un mot de passe au Super_Utilisateur). Dans le cas contraire, une erreur est générée.

Pour rétablir la base principale en tant que base courante, exécutez la commande en passant le mot-clé **SQL_INTERNAL** ou **DEFAULT**.

Passez **AUTO_CLOSE** si vous souhaitez fermer physiquement la base externe à l'issue de son utilisation, c'est-à-dire lorsque vous changerez de base courante. En effet, l'ouverture d'une base externe étant une opération qui nécessite du temps, pour des raisons d'optimisation 4D maintient en mémoire des informations relatives aux bases externes ouvertes durant la session utilisateur. Ces informations sont maintenues en mémoire tant que l'application 4D est lancée. Les réouvertures suivantes d'une même base externe sont alors accélérées. Toutefois, ce principe empêche le partage des bases externes entre plusieurs applications 4D car la base externe reste ouverte en

lecture/écriture pour la première application qui l'a utilisée. Si plusieurs applications 4D doivent pouvoir utiliser simultanément une même base externe, passez le mot-clé **AUTO_CLOSE** afin de libérer physiquement la base externe après son utilisation.

Cette restriction ne s'applique pas aux process d'une même application : différents process d'une application peuvent toujours accéder à une même base externe en lecture/écriture sans qu'il soit nécessaire de forcer sa fermeture.
A noter que lorsque plusieurs process utilisent la même base externe, elle n'est libérée physiquement que lorsque le dernier process qui l'utilise est refermé, même lorsque l'option **AUTO_CLOSE** a été passée. Vous devez tenir compte de ce fonctionnement pour les opérations de partage inter-applications et de suppression des bases externes.

- Utilisation d'une base externe pour une requête puis retour à la base principale :

```
Debut SQL
  USE DATABASE DATAFILE 'C:/MaBase/Noms'
  SELECT Name FROM emp INTO :tNoms1
  USE DATABASE SQL_INTERNAL
Fin SQL
```

Référence : CREATE DATABASE, DATABASE_PATH

ALTER DATABASE

ALTER DATABASE {ENABLE | DISABLE} {INDEXES | CONSTRAINTS}

La nouvelle commande ALTER DATABASE active ou inactive des options SQL de la base courante pour la session courante.

Cette commande vous permet de désactiver temporairement des options SQL afin d'accélérer certaines opérations consommatrices en ressources. Par exemple, désactiver les index et les contraintes avant de débuter un import d'une grande quantité de données peut réduire de façon significative la durée de l'import. Les contraintes incluent les clés primaires, les clés étrangères, les attributs d'unicité et de nullité.

ALTER DATABASE s'applique à la totalité de la base. Autrement dit, si un utilisateur inactive une option, elle sera inactivée pour tous les utilisateurs de la base.

- Exemple d'import avec désactivation temporaire de toutes les options SQL :

```
Debut SQL
  ALTER DATABASE DISABLE INDEXES;
  ALTER DATABASE DISABLE CONSTRAINTS;
Fin SQL
SQL EXECUTER SCRIPT ("C:\Exported_data\Export.sql"; SQL Continuer si erreur)
Debut SQL
  ALTER DATABASE ENABLE INDEXES;
  ALTER DATABASE ENABLE CONSTRAINTS;
Fin SQL
```

REPLICATE

REPLICATE liste_répliquée
FROM ref_table
[**WHERE** critère_recherche]
[**LIMIT** {nombre_entier | ref_langage_4d}]
[**OFFSET** {nombre_entier | ref_langage_4d}]
FOR REMOTE [**STAMP**] {nombre_entier | ref_langage_4d}
[, LOCAL [STAMP] {nombre_entier | ref_langage_4d }]
[{**REMOTE OVER LOCAL** | **LOCAL OVER REMOTE**}]
[**LATEST REMOTE** [**STAMP**] ref_langage_4d
[, LATEST LOCAL [STAMP] ref_langage_4d]]
INTO {liste_cible | ref_table(nom_sql_1;...;nom_sql_N)};

La nouvelle commande REPLICATE vous permet de répliquer les données d'une table d'une base A dans celles d'une table d'une base B. Par convention, la base sur laquelle est exécutée la commande est nommée "base locale" et la base de laquelle les données sont répliquées est nommée "base distante".

Cette commande peut être utilisée uniquement dans le cadre d'un système de réplication de base. Pour que le système fonctionne, la réplication doit avoir été activée côté base distante et côté base locale et chaque table impliquée doit comporter une clé primaire. Pour plus d'informations, reportez-vous au paragraphe "Réplication via le SQL", page 297.

Note Si vous souhaitez mettre en place un système de synchronisation complète, reportez-vous à la description de la nouvelle commande SYNCHRONIZE.

Passez dans *liste_répliquée* une liste de champs (virtuels ou classiques) séparés par une virgule. Les champs doivent appartenir à la table *ref_table* de la base distante.
La clause FROM doit être suivie d'un argument de type *ref_table* permettant de désigner la table de la base distante depuis laquelle répliquer les données des champs *liste_répliquée*.

Note Les champs virtuels de la table distante ne pourront être stockés que dans des tableaux de la base locale.

Côté base distante

La clause optionnelle WHERE permet d'appliquer un filtre préalable aux enregistrements de la table dans la base distante ; seuls les enregistrements qui satisfont au *critère_recherche* sont pris en compte par la commande.

4D récupère ensuite les valeurs des champs *liste_répliquée* pour tous les enregistrements désignés par la clause FOR REMOTE STAMP. La valeur passée dans cette clause peut être soit :

- une valeur de type **entier long** > **0** : dans ce cas, les enregistrements dont la valeur du marqueur __ROW_STAMP est supérieure ou égale à cette valeur sont récupérés.
- **0** : dans ce cas, tous les enregistrements dont la valeur du marqueur __ROW_STAMP est différente de 0 sont récupérés. A noter que les enregistrements existant éventuellement avant l'activation de la réplication ne seront donc pas pris en compte (leur valeur de __ROW_STAMP = 0).
- **-1** : dans ce cas, tous les enregistrements de la table distante sont récupérés, autrement dit tous les enregistrements dont la valeur du marqueur __ROW_STAMP >= 0. A la différence du cas précédent, tous les enregistrements de la table, y compris les enregistrements existant éventuellement avant l'activation, seront pris en compte.
- **-2** : dans ce cas, tous les enregistrements supprimés de la table distante (après activation de la réplication) sont récupérés, autrement dit tous les enregistrements dont la valeur du marqueur __ROW_ACTION = 2.

Enfin, vous pouvez appliquer à la sélection obtenue les clauses optionnelles OFFSET et LIMIT :

- Lorsqu'elle est passée, la clause OFFSET permet d'exclure les *N* premiers enregistrements de la sélection (*N* étant la valeur passée à la clause).
- Lorsqu'elle est passée, la clause LIMIT permet de restreindre la sélection aux *M* premiers enregistrements (*M* étant la valeur passée à la clause). Si la clause OFFSET a également été passée, la clause LIMIT est appliquée à la sélection obtenue après exécution de OFFSET.

Une fois l'ensemble des clauses appliquées, la sélection résultante est envoyée à la base locale.

Côté base locale

Les valeurs récupérées sont directement écrites dans la *liste_cible* de la base locale ou dans les champs classiques définis par *nom_sql* de la table *ref_table* de la base locale. L'argument *liste_cible* peut contenir soit une liste de champs standard, soit une liste de tableaux du même type que les champs distants (mais pas une combinaison des deux). Si la destination de la commande est une liste de champs, les enregistrements cibles seront automatiquement créés, modifiés ou supprimés en fonction de l'action stockée dans le champ virtuel __ROW_ACTION.

Les conflits (clés primaires identiques) sont résolus à l'aide de la clause de gestion de conflits (option REMOTE OVER LOCAL ou LOCAL OVER REMOTE) :

- si vous passez l'option REMOTE OVER LOCAL ou omettez la clause de gestion de conflits, en cas de conflit l'enregistrement source (base distante) remplace toujours l'enregistrement cible (base locale). A noter que si vous utilisez cette option, il est inutile de passer la clause LOCAL STAMP car dans ce cas, elle est ignorée.
- si vous passez l'option LOCAL OVER REMOTE, en cas de conflit l'enregistrement cible (base locale) n'est pas modifié. La commande tient compte du marqueur local (LOCAL STAMP). La clause LOCAL STAMP, lorsqu'elle est utilisée, permet de limiter la réplication dans la table locale aux enregistrements en conflit dont la valeur du marqueur est inférieure ou égale à celle qui est passée. Cette clause permet de réduire la sélection d'enregistrements en conflit qui seront répliqués dans la table locale.

- Si vous passez les clauses LATEST REMOTE STAMP et/ou LATEST LOCAL STAMP, 4D retourne dans les variables *ref_langage_4d* correspondantes les valeurs des derniers marqueurs des tables distante et locale. Ces informations peuvent être utiles si vous souhaitez personnaliser la gestion de la procédure de synchronisation. Ces valeurs correspondent à la valeur des marqueurs juste après la fin de l'opération de réplication : si vous les utilisez dans une instruction REPLICATE ou SYNCHRONIZE ultérieure, vous n'avez pas besoin de les incrémenter, ils le sont automatiquement avant d'être retournés par la commande REPLICATE.

  Si l'opération de réplication de déroule correctement, la variable système OK prend la valeur 1. Vous pouvez contrôler cette valeur depuis une méthode 4D.
  Si des erreurs se produisent durant l'opération de réplication, l'opération est stoppée à la première erreur rencontrée. La dernière variable source (si elle a été définie) est valorisée avec le marqueur de l'enregistrement dans lequel l'erreur s'est produite. La variable système OK prend la valeur 0. L'erreur générée peut être interceptée par une méthode de gestion d'erreurs installée par la commande APPELER SUR ERREUR.

Note Les opérations effectuées par la commande REPLICATE ne tiennent pas compte des contraintes d'intégrité des données. Cela signifie par exemple que les règles de clés étrangères, d'unicité, etc. ne sont pas vérifiées. Si les données reçues peuvent remettre en cause l'intégrité des données, vous devez les vérifier à l'issue de l'opération de réplication. Le moyen le plus simple est de verrouiller via le langage 4D ou le langage SQL les enregistrements devant être modifiés.

SYNCHRONIZE

SYNCHRONIZE
[LOCAL] TABLE ref_table (nom_sql_1;...;nom_sql_N)
WITH
[REMOTE] TABLE ref_table (nom_sql_1;...;nom_sql_N)
FOR REMOTE [STAMP] {nombre_entier | ref_langage_4d},
LOCAL [STAMP] {nombre_entier | ref_langage_4d}
{**REMOTE OVER LOCAL** | **LOCAL OVER REMOTE**}
LATEST REMOTE [STAMP] ref_langage_4d,
LATEST LOCAL [STAMP] ref_langage_4d;

La nouvelle commande SYNCHRONIZE permet de synchroniser deux tables situées sur deux serveurs 4D SQL différents. Toute modification effectuée sur une table est reportée dans l'autre. Le serveur 4D SQL qui exécute la commande est appelé serveur local (LOCAL), l'autre serveur est appelé serveur distant (REMOTE).

Note La commande SYNCHRONIZE équivaut à la combinaison de deux appels internes à la commande REPLICATE. Le premier appel réplique les données depuis le serveur distant vers le serveur local, le second effectue l'opération inverse : réplication des données du serveur local vers le serveur distant. Les tables à synchroniser doivent donc être configurées pour la réplication :
- elles doivent comporter une clé primaire,
- l'option "Activer réplication" doit être cochée dans l'Inspecteur de chaque table.
Pour plus d'informations, reportez-vous à la description de la commande REPLICATE.

La commande SYNCHRONIZE accepte quatre marqueurs (*stamps*) en "paramètres" : deux marqueurs en entrée et deux marqueurs en sortie (dernière modification). Les marqueurs d'entrée sont utilisés pour indiquer le moment de la dernière synchronisation sur chaque serveur. Les marqueurs de sortie retournent la valeur des marqueurs de modification sur chaque serveur juste après la dernière modification. Grâce à ce principe, lorsque la commande SYNCHRONIZE est appelée régulièrement, il est possible d'utiliser les marqueurs de sortie de la dernière synchronisation en tant que marqueurs d'entrée pour la suivante.

Note Les marqueurs d'entrée et de sortie sont exprimés sous forme de valeurs numériques (stamps), et non de marqueurs de temps (timestamps). Pour plus d'informations sur ces marqueurs, reportez-vous à la description de la commande REPLICATE.

En cas d'erreur, le marqueur de sortie du serveur concerné contient le marqueur de l'enregistrement à l'origine de l'erreur. Si l'erreur provient d'une cause externe à la synchronisation (problèmes réseau par exemple), le marqueur contient 0.
Il y a deux codes d'erreurs différents, l'un pour indiquer une erreur de synchronisation sur le site local et l'autre sur le site distant.
En cas d'erreur, l'état des données dépend de celui de la transaction sur le serveur local. Sur le serveur distant, la synchronisation est toujours effectuée dans le contexte d'une transaction, les données ne peuvent donc pas être altérées par l'opération. Sur le serveur local en revanche, le processus de synchronisation est placé sous le contrôle du développeur. Il est effectué en-dehors de toute transaction si la préférence **Transactions Auto-commit** est désélectionnée (dans le cas contraire, un contexte de transaction est automatiquement créé). Le développeur peut décider de démarrer une transaction, il lui revient de la valider ou de l'annuler après la synchronisation des données.

Les conflits sont résolus à l'aide des clauses REMOTE OVER LOCAL et LOCAL OVER REMOTE, permettant d'indiquer clairement quel serveur sera prioritaire si un même enregistrement a été modifié des deux côtés. Pour plus d'informations sur les mécanismes mis en oeuvre, reportez-vous à la description de la commande REPLICATE.

Note Les opérations effectuées par la commande SYNCHRONIZE ne tiennent pas compte des contraintes d'intégrité des données. Cela signifie par exemple que les règles de clés étrangères, d'unicité, etc. ne sont pas vérifiées. Si les données reçues peuvent remettre en cause l'intégrité des données, vous devez les vérifier à l'issue de l'opération de synchronisation. Le moyen le plus simple est de verrouiller via le langage 4D ou le langage SQL les enregistrements devant être modifiés.

Nouvelle fonction

DATABASE_PATH

DATABASE_PATH()

La fonction DATABASE_PATH retourne le chemin d'accès complet de la base courante. La base courante peut être modifiée à l'aide de la commande SQL USE DATABASE. Par défaut, la base courante est la base 4D principale.

- Supposons que la base externe courante est nommée TestBase.4DB et est située dans le dossier "C:\MesBases". Après l'exécution du code suivant :

```
C_TEXTE(vCrtDatabasePath)
Debut SQL
  SELECT DATABASE_PATH()
    FROM _USER_SCHEMA
    LIMIT 1
    INTO :vCrtDatabasePath;
Fin SQL
```

...la variable *vCrtDatabasePath* contiendra "C:\MesBases\TestBase.4DB".

Référence : CREATE DATABASE, USE DATABASE

Commandes SQL modifiées

Cette section présente les modifications apportées aux commandes SQL existantes de 4D. Ces modifications apparaissent en *caractères italiques*.

CREATE TABLE

CREATE TABLE [IF NOT EXISTS] {nom_sql.}nom_sql({définition_colonne |contrainte_table} ***[PRIMARY KEY]***, ... , {définition_colonne |contrainte_table} ***[PRIMARY KEY]***) ***[{ENABLE | DISABLE} REPLICATE]***

La commande CREATE TABLE accepte deux nouveaux mots-clés :

- PRIMARY KEY : permet de définir la clé primaire au moment de la création de la table (cf. paragraphe "Définir une clé primaire en création de colonne", page 300).

Note Le mot-clé PRIMARY KEY est accepté depuis la version 11.4 de 4D.

- {ENABLE | DISABLE} REPLICATE : permet d'activer et d'inactiver le mécanisme autorisant la réplication de la table (cf. paragraphe "Activation de la réplication", page 298).

ALTER TABLE

ALTER TABLE nom_sql
{ADD définition_colonne ***[PRIMARY KEY]***|
DROP nom_sql |
ADD définition_clé_primaire |
DROP PRIMARY KEY |
ADD définition_clé_étrangère |
DROP CONSTRAINT nom_sql |
[{ENABLE | DISABLE} REPLICATE] |
SET SCHEMA nom_sql}

La commande ALTER TABLE accepte deux nouveaux mots-clés :

- PRIMARY KEY : permet de définir la clé primaire au moment de l'ajout d'une colonne (cf. paragraphe "Définir une clé primaire en création de colonne", page 300).

Note Le mot-clé PRIMARY KEY est accepté depuis la version 11.4 de 4D.

- {ENABLE | DISABLE} REPLICATE : permet d'activer et d'inactiver le mécanisme autorisant la réplication de la table (cf. paragraphe "Activation de la réplication", page 298).

7 Administration de 4D Server

A propos de 4D Server 64 bits

4D Server v12 pour Windows se décline en version 32 bits (standard) et 64 bits. La version 64 bits est destinée à une utilisation sur les systèmes d'exploitation Windows 64 bits.

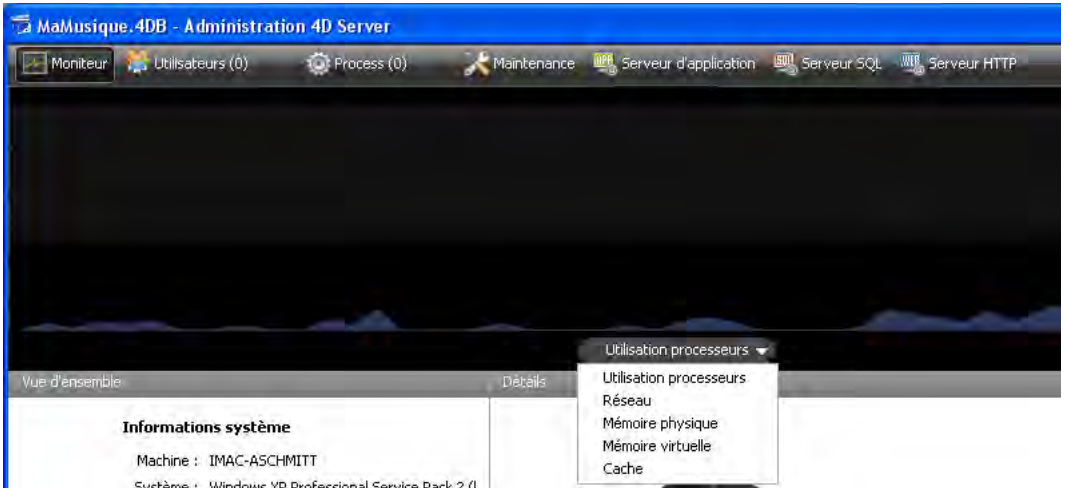
La version 64 bits de 4D Server est actuellement en phase de finalisation et fait l'objet d'un programme de *beta test*. Vous pouvez dès à présent évaluer cette version en vous connectant au site Web de 4D (http://www.4d.com/). Elle sera disponible prochainement en version publique finale.

Pour des informations techniques sur 4D Server 64 bits, reportez-vous au manuel *4D Server 64 bits*, disponible depuis le site de documentation de 4D (http://doc.4d.com/).

Fenêtre d'administration de 4D Server

Nouvelles informations mémoire

La page **Moniteur** de la fenêtre d'administration de 4D Server donne accès à plusieurs informations supplémentaires relatives à la mémoire occupée par l'application. Ces nouvelles informations sont accessibles via le menu situé au centre de la fenêtre :



Ce menu comporte les nouveautés suivantes :

- **Mémoire physique** : Indique la quantité de mémoire RAM installée sur la machine utilisée par 4D Server. Cette option était présente sous le nom "Mémoire" dans les versions précédentes de 4D Server.
- **Mémoire virtuelle** : Affiche dans la zone graphique la quantité de mémoire virtuelle utilisée par l'application 4D Server. Cette mémoire est allouée par le système en fonction des besoins de l'application. La valeur située en bas à droite de la zone indique la quantité de mémoire en cours d'utilisation. La valeur située en haut à gauche indique la quantité maximale de mémoire virtuelle utilisable. La valeur maximale est calculée dynamiquement en fonction des paramètres mémoire généraux de l'application.

- **Cache** : Affiche dans la zone graphique la quantité de mémoire cache utilisée par l'application 4D Server. La valeur située en bas à droite de la zone indique la quantité de mémoire en cours d'utilisation. La valeur située en haut à gauche indique la taille totale de la mémoire cache, telle que définie via les Propriétés de la base.
A noter que lorsque cette option est sélectionnée, le défilement de la zone graphique est ralenti car une analyse efficace du cache s'effectue généralement sur une période d'observation assez longue.

Disparition des utilisateurs "techniques"

Dans les versions précédentes de 4D Server, un utilisateur était automatiquement ajouté dans la liste des utilisateurs dès lors qu'au moins une procédure stockée était lancée sur le serveur et qu'un client était connecté. Le même mécanisme était mis en oeuvre lorsqu'au moins un process Web était lancé. Bien entendu, ce type d'utilisateur, créé pour des raisons techniques, n'était pas comptabilisé dans les licences consommées par l'application. Sa présence pouvait cependant être parfois source de confusion.

Pour plus de transparence, les utilisateurs "techniques" n'apparaissent plus désormais dans la fenêtre d'administration de 4D Server v12. A noter que cette modification n'a aucune conséquence sur le fonctionnement de 4D Server.

Arrêt de la prise en charge des Services sous Mac OS

Pour des raisons techniques, à compter de 4D v12 la fonction d'enregistrement de bases 4D Server comme Services n'est plus prise en charge sous Mac OS. Les commandes de menu correspondantes ne sont plus disponibles dans l'interface de 4D Server sur cette plate-forme.

Ce mécanisme est maintenu sans modification sous Windows.

A Modules PHP

Cette annexe détaille l'implémentation des modules PHP dans 4D v12. Les sujets suivants sont abordés :

- liste des modules standard PHP fournis par défaut avec l'interpréteur PHP de 4D
- liste des modules standard PHP non retenus par 4D,
- instructions d'installation de modules supplémentaires.

Modules fournis par défaut

Le tableau suivant détaille les modules PHP fournis par défaut avec 4D v12.

Modules génériques

Nom	Site web	Description
BCMath	http://php.net/bc	Calculateur binaire qui prend en charge des nombres de n'importe quelle taille et précision représentés sous forme de chaînes.
	Exemple : // Retourner le résultat de ($maValeur*$maValeur*$maValeur) **PHP Executer**("";"bcpow";$résultat;**Chaine**($maValeur);"3";)	
Calendar	http://php.net/calendar	Ensemble de fonctions simplifiant la conversion entre les différents formats de calendriers. Se base sur le Jour Julien.
	Exemple : **PHP Executer**("";"cal_days_in_month";Number of days in Febrary;1;2;Year())	

Nom	Site web	Description
Ctype	http://php.net/ctype	Fonctions vérifiant si un caractère ou une chaîne appartient à une certaine classe de caractères, suivant la configuration locale courante
	Exemple : // Vérifier que tous les caractères de la chaîne fournie sont des signes de ponctuation **PHP Executer**("";"ctype_punct";isPunct;$maChaine)	
Date and Time	http://php.net/datetime	Récupération de la date et de l'heure depuis le serveur où le script PHP s'exécute
	Exemple : //Calcul de l'heure du lever du soleil à Lisbonne, Portugal, Latitude: 38.4 Nord, Longitude: 9 Ouest, Zénith ~= 90, Décalage: +1 GMT **PHP Executer**("";"eval";$resultType;"SUNFUNCS_RET_STRING") **PHP Executer**("";"date_sunrise";SunriseTime;$timestamp;$resultType;38.4l;-9; 90; 1)	
DOM (Document Object Model)	http://php.net/dom	Utilisation de documents XML via l'API DOM de PHP 5
Exif	http://php.net/exif	Travail avec les méta-données des images.
Fileinfo(*)	http://php.net/fileinfo	Détection du type de contenu et de l'encodage d'un fichier.
Filter	http://php.net/filter	Valider et filtrer les données issues de source non sécurisée, comme les entrées des utilisateurs.
	Exemple : **PHP Executer**("";"filter_id";$filterId;"FILTER_VALIDATE_EMAIL") **APPELER SUR ERREUR**("notValidEmail") validEmail:=$emailToValidate $ok:=**PHP Executer**("";"filter_var";$test;$filterId;$emailToValidate;$resultType) **APPELER SUR ERREUR**("")	
FTP (File Transfert Protocol)	http://php.net/ftp	Accès détaillé à un serveur FTP
Hash	http://php.net/hash	Moteur d'empreinte numérique. Permet le traitement direct ou incrémental de message de grandeur arbitraire en utilisant une variété d'algorithmes
	Exemple : **PHP Executer**("";"hash";$md5Result;"md5";$maChaine)	
GD (Graphics Draw) Library	http://php.net/gd	Manipulation d'images

Nom	Site web	Description
Iconv	http://php.net/iconv	Conversion de fichiers entre divers jeux de caractères
JSON (JavaScript Object Notation)	http://php.net/json	Implémentation du format d'échange de données JSON
LDAP	http://php.net/ldap	LDAP est un protocole d'accès aux "serveurs de dossiers" stockant les informations sous forme d'arborescence
LibXML	http://php.net/libxml	Librairie de fonctions et constantes XML
Multibyte String	http://php.net/mbstring	Ensemble de fonctions de manipulation de chaînes qui vous permet de travailler avec les encodages multi-octets ou de traduire des jeux de caractères.
OpenSSL	http://php.net/openssl	Utilisation des fonctions de OpenSSL pour générer et vérifier les signatures, sceller (chiffrer) et ouvrir (déchiffrer) les données.
PCRE (Perl Compatible Regular Expressions)	http://php.net/pcre	Ensemble de fonctions qui implémentent les expressions rationnelles en utilisant la même syntaxe et sémantique que Perl 5
	Exemple : // Cet exemple supprime les espaces superflus d'une chaîne. $maChaîne:="foo o bar" **PHP Executer**("";"preg_replace";$maChaîne;"/\\s\\s+/";" ";$maChaîne) // $maChaîne est maintenant "foo o bar" sans espaces dupliqués	
PDO (PHP Data Objects)	http://php.net/pdo	Interface d'accès à une base de données. Nécessite un driver PDO spécifique à la base de données.
PDO_SQLITE	http://php.net/pdo_sqlite	Pilote qui implémente l'interface de PHP Data Objects (PDO) pour autoriser l'accès de PHP aux bases de données SQLite 3.
Phar (PHP Archive)	http://php.net/phar	Permet d'inclure une application PHP complète dans un fichier unique appelé "phar" (PHP Archive) pour faciliter son installation et sa configuration

Nom	Site web	Description
Reflection	http://php.net/reflection	API de réflexion complète qui permet de faire du *reverse-engineering* sur les classes, les interfaces, les fonctions, les méthodes, les extensions
Session	http://php.net/session	Prise en charge de sessions PHP
	Exemple : Les sessions sont utilisées dans les applications Web pour conserver le contexte entre chaque requête. Lorsque vous appelez PHP Executer dans 4D, le script PHP peut démarrer une session et stocker tout ce qui est utile à conserver comme contexte dans le tableau associé $_SESSION. Si un script PHP utilise des sessions vous devez obtenir l'ID de session retourné par PHP à l'aide de la commande PHP LIRE REPONSE COMPLETE et le définir avant chaque appel à PHP Executer à l'aide de la commande FIXER VARIABLE ENVIRONMENT // Méthode "PHP Exécuter avec contexte" **Si** (<>PHP_Session # "") **FIXER VARIABLE ENVIRONNEMENT**("HTTP_COOKIE";<>PHP_Session) **Fin de si** **Si** (**PHP Executer**($1)) **PHP LIRE REPONSE COMPLETE**($0;$errorInfos;$errorValues;$headerFields; $headerValues) $idx:=**Chercher dans tableau**($headerFields; "Set-Cookie") **Si** ($idx>0) <>PHP_Session:=$headerValues{$idx} **Fin de si** **Fin de si**	
SimpleXML	http://php.net/simpleXML	Outils très simples et faciles à utiliser pour convertir du XML en un objet qui peut être manipulé avec ses propriétés et les itérateurs de tableaux
Sockets	http://php.net/sockets	Implémentation d'une interface bas niveau avec les fonctions de communication par socket basées sur les sockets BSD et fournit la possibilité de fonctionner aussi bien sous forme de client que de serveur.
SPL (Standard PHP Library)	http://php.net/spl	Collection d'interfaces et de classes utilitaires créés pour résoudre les problèmes usuels.
SQLite	http://php.net/sqlite	Extension pour le moteur de base de données SQLite. Ce moteur peut être embarqué.

Nom	Site web	Description
SQLite3	http://php.net/sqlite3	Support pour les bases de données SQLite version 3
Tokenizer	http://php.net/tokenizer	Fonctions vous permettant d'écrire vos propres outils PHP d'analyse ou de modifications sans avoir à vous soucier de la spécification du langage au niveau lexical
XML (eXtensible Markup Language)	http://php.net/xml	Analyse des documents XML
XMLreader	http://php.net/xmlreader	Analyseur XML Pull
XMLwriter	http://php.net/xmlwriter	Génération de flux et de fichiers au format XML
Zlib	http://php.net/zlib	Lecture et écriture de fichiers compressés gzip (.gz)
	Exemple : **LIRE ENTETE HTTP**($names;$values) $pos:=Chercher dans tableau($names;"Accept-Encoding") **Si**($pos>0) **Au cas ou** :(**Position**($values{$pos};"gzip")>0) **FIXER ENTETE HTTP**("Content-Encoding: gzip") **PHP Executer**("";"gzencode";$html;$html) :(**Position**($values{$pos};"deflate")>0) **FIXER ENTETE HTTP**("Content-Encoding: deflate") **PHP Executer**("";"gzdeflate";$html;$html) **Fin de cas** **Fin de si** **ENVOYER TEXTE HTML**($html)	
Zip	http://php.net/zip	Lecture et écriture des archives compressées ZIP et des fichiers s'y trouvant

(*) Non disponible sous Windows dans la version actuelle de 4D v12.

Modules disponibles sous Windows uniquement

Pour des raisons structurelles, les modules PHP suivants ne sont disponibles que sur la plate-forme Windows.

Nom	Site web	Description
COM & .NET	http://php.net/com	COM (Component Object Model) est l'une des méthodes les plus utilisées pour faire communiquer des applications et des composants sur les plates-formes Windows. En outre, 4D prend en charge l'instanciation et la création d'assemblages .Net via la couche COM.
ODBC (Open DataBase Connectivity)	http://php.net/odbc	En plus du support de l'ODBC standard, l'ODBC unifié de PHP vous donne accès à diverses bases de données qui ont emprunté la sémantique des API ODBC pour implémenter leur propres API.
WDDX (Web Distributed Data eXchange)	http://php.net/wddx	Facilite les échanges de données inter-applications Web via le Web, quelle que soit la plate-forme.

Modules désactivés

Les modules PHP suivants n'ont pas été implémentés dans 4D v12. La colonne de droite fournit la raison de cette non-implémentation :

Nom	Site web	Cause - Solution alternative
Mimetype	http://php.net/mime-magic	Obsolète (Deprecated) - Utiliser Fileinfo
POSIX (Portable Operating System Interface)	http://php.net/posix	Obsolète (Deprecated)
Regular Expression (POSIX Extended)	http://php.net/regex	Obsolète (Deprecated) - Utiliser PCRE
Crack	http://php.net/crack	Licence restrictive
ffmpeg	http://ffmpeg-php.sourceforge.net/	Licence restrictive - Utiliser ffmpeg en ligne de commande avec LANCER PROCESS EXTERNE
Image Magick	http://php.net/manual/book.imagick.php	Licence restrictive - Utiliser GD 2
IMAP (Internet Message Access Protocol)	http://php.net/imap	Licence restrictive - Utiliser le plug-in intégré 4D Internet Commands

Nom	Site web	Cause - Solution alternative
PDF (Portable Document Format)	http://php.net/pdf	Licence restrictive - Utiliser Haru PDF
Mysqlnd (MySQL Native Driver)	http://dev.mysql.com/downloads/connector/php-mysqlnd/	Non pertinent dans l'environnement 4D

Installation de modules supplémentaires

L'interpréteur PHP vous laisse la possibilité d'installer des modules supplémentaires. Ce principe vous permet d'accéder à des fonctionnalités spécifiques non présentes par défaut.

Vous pouvez installer plusieurs types d'extensions :

- extensions PECL (PHP Extension Community Library)
- extensions du framework PEAR (PHP Extension and Application Repository)
- extensions du framework Zend
- extensions du framework Symphony
- extensions du framework JELIX
- composants eZ

Les informations d'installation pour chaque type d'extension sont fournies ci-dessous.

Note Les caractéristiques de la version de PHP fournie avec 4D v12 sont les suivantes :
- version 5.3.2
- compilation 32 bits sous Windows et Mac OS
- compilation en mode "thread-safe" sous Windows et Mac OS.

Extensions PECL

Site Web : http://pecl.php.net

▶ Pour ajouter des extensions PECL :

1 **Téléchargez et construisez l'extension PECL souhaitée.**
OU
Prenez l'extension déjà construite depuis un package binaire Windows PHP 5.3 VC9 Non Thread Safe
(http://windows.php.net/download/#php-5.3-nts-VC9-x86**)**

2 **Ajoutez l'extension dans le dossier des extensions.**

3 **Activez l'extension dans le fichier php.ini.**

Attention : si les extensions disponibles sur le site Web PECL sont sous licence PHP qui n'est pas restrictive, certaines peuvent nécessiter des librairies qui, elles, ont une licence plus contraignante.

Des extensions PHP sont disponibles sur d'autres sites Web, mais elles ne bénéficient pas dans ce cas de la garantie de sécurité apportée par la validation du PHP Group.

Extensions PEAR

Site Web : http://pear.php.net

PEAR est un framework entièrement orienté objet.

▶ Pour ajouter des extensions PEAR :

1 **Téléchargez (**http://pear.php.net/package/PEAR/download**) et décompactez le package PEAR dans un dossier nommé "pear".**

2 **Ajoutez ce dossier "pear" dans le "include_path" défini dans le fichier "php.ini".**

3 **Téléchargez et décompactez tout package PEAR dans ce dossier.**

Extensions Zend

1 **Téléchargez (**http://framework.zend.com/download/latest**) et décompactez le framework Zend dans un dossier nommé "zend".**

2 **Ajoutez ce dossier "zend" dans le "include_path" défini dans le fichier "php.ini".**

3 **Lisez la documentation des composants du framework Zend :** http://framework.zend.com/manual/en

Extensions Symphony

Site Web : http://www.symfony-project.org

Le framework Symphony est structuré de manière à être utilisé en tant qu'application Web accompagnée de son contrôleur Web.

1 **Téléchargez et décompactez le framework source ainsi que le bac à sable depuis l'adresse** http://www.symfony-project.org/installation/1_2**.**

Extensions JELIX

1 **Téléchargez (**http://jelix.org/articles/en/download/stable**) et décompactez le framework JELIX.**

2 **Ajoutez le dossier "jelix" résultant dans le "include_path" défini dans le fichier "php.ini".**

3 **Lisez la documentation des composants du framework JELIX :** http://jelix.org/articles/en/manual-1.1/components

Composants eZ

1 **Téléchargez (**http://www.ezcomponents.org/download**) et décompactez les composants eZ dans un dossier "ez".**

2 **Ajoutez le dossier "ez" dans le "include_path" défini dans le fichier "php.ini".**

3 **Lisez la documentation des composants eZ :** http://www.ezcomponents.org/docs/api/latest

B Balises de style

Cette annexe liste les balises de style prises en charge par 4D v12 dans les zones de texte riche (zones disposant de l'attribut "Multistyle"). Pour une description de cet attribut, reportez-vous au paragraphe "Zones de texte riche", page 96.

Vous pouvez utiliser ces balises pour mettre en place une gestion personnalisée des styles. Seules les balises listées ci-dessous sont prises en charge par 4D pour les variations de style.

Nom de police

<SPAN STYLE="font-family: DESDEMONA"> ... </SPAN>

Taille de police

<SPAN STYLE="font-size: 20pt"> ... </SPAN>

Style de police

- **Gras**

  <SPAN STYLE="font-weight: bold"> ... </SPAN>

- **Italique ou normal**

  <SPAN STYLE="font-style: italic"> ... </SPAN>
  <SPAN STYLE="font-style: normal"> ... </SPAN>

- **Souligné**

  <SPAN STYLE="text-decoration: underline"> ... </SPAN>

- **Barré**

  <SPAN STYLE="text-decoration:line-through">...</SPAN>

Note Le style "barré" n'est pas pris en charge sous Mac OS. La balise correspondante peut toutefois être gérée par programmation.

Couleurs de police

<SPAN STYLE="color:green"> ... </SPAN>

ou

<SPAN STYLE="color:#006CCC">...</SPAN>

Couleurs de fond (Windows uniquement)

<SPAN STYLE="background-color:green"> ... </SPAN>

ou

<SPAN STYLE="background-color:#006CCC">...</SPAN>

Note Sous Mac OS, cet attribut est ignoré. Il est supprimé en cas de modification de l'objet.

Valeurs de couleurs

Pour les attributs de couleur de police et couleur de fond, la valeur de couleur peut être soit le code hexadécimal des couleurs RVB soit le nom d'une des 16 couleurs HTML définies pour le standard CSS par le W3C :

Color								
Name	Aqua	Black	Blue	Fuchsia	Gray	Green	Lime	Maroon
RGB	#00FFFF	#000000	#0000FF	#FF00FF	#808080	#008000	#00FF00	#800000

Color								
Name	Navy	Olive	Purple	Red	Silver	Teal	White	Yellow
RGB	#000080	#808000	#800080	#FF0000	#C0C0C0	#008080	#FFFFFF	#FFFF00

C Constantes de métadonnées

Cette annexe présente les constantes présentes dans les nouveaux thèmes "Noms des métadonnées images" et "Valeurs des métadonnées images" de 4D v12. Pour plus d'informations sur l'utilisation de ces constantes, reportez-vous à la description des commandes FIXER METADONNEES IMAGE et LIRE METADONNEES IMAGE.

Noms des métadonnées images

EXIF

Constantes du thème "Noms des métadonnées images" (paramètre *nomMéta*)	**Valeur**	**Types de valeurs possibles ou constantes associées du thème "Valeurs des métadonnées images" (paramètre *contenu*)**
EXIF Aperture Value	EXIF/ApertureValue	*Réel (valeur APEX)*
EXIF Brightness Value	EXIF/BrightnessValue	*Réel (valeur APEX)*
EXIF Color Space	EXIF/ColorSpace	EXIF Adobe RGB
		EXIF s RGB
		EXIF Uncalibrated
EXIF Components Configuration	EXIF/ComponentsConfiguration	EXIF B
		EXIF Cb
		EXIF Cr
		EXIF G
		EXIF R
		EXIF Unused
		EXIF Y
EXIF Compressed Bits Per Pixel	EXIF/CompressedBitsPerPixel	*Réel*

EXIF Contrast	EXIF/Contrast	EXIF High
		EXIF Low
		EXIF Normal
EXIF Custom Rendered	EXIF/CustomRendered	EXIF Normal
		EXIF Custom
EXIF Date Time Digitized	EXIF/DateTimeDigitized	*Date ou Texte (Datetime XML)*
EXIF Date Time Original	EXIF/DateTimeOriginal	*Date ou Texte (Datetime XML)*
EXIF Digital Zoom Ratio	EXIF/DigitalZoomRatio	*Réel*
EXIF Exif Version	EXIF/ExifVersion	*Tableau Entier (4 valeurs)*
EXIF Exposure Bias Value	EXIF/ExposureBiasValue	*Réel*
EXIF Exposure Index	EXIF/ExposureIndex	*Réel*
EXIF Exposure Mode	EXIF/ExposureMode	EXIF Auto
		EXIF Auto Bracket
		EXIF Manual
EXIF Exposure Program	EXIF/ExposureProgram	EXIF Manual
		EXIF Action
		EXIF Aperture Priority AE
		EXIF Creative
		EXIF Landscape
		EXIF Exposure Portrait
		EXIF Program AE
		EXIF Shutter Speed Priority AE
EXIF Exposure Time	EXIF/ExposureTime	*Réel*
EXIF File Source	EXIF/FileSource	EXIF Digital Camera
		EXIF Film Scanner
		EXIF Reflection Print Scanner
EXIF Flash	EXIF/Flash	EXIF Auto Mode
		EXIF Compulsory Flash Firing
		EXIF Compulsory Flash Suppression
		EXIF Unknown
		EXIF Detected
		EXIF No Detection Function
		EXIF Not Detected
		EXIF Reserved
EXIF Flash Energy	EXIF/FlashEnergy	*Réel*
EXIF Flash Pix Version	EXIF/FlashPixVersion	*Tableau Entier (4 valeurs)*
EXIF Flash Fired	EXIF/Flash/Fired	*Booléen*
EXIF Flash Function Present	EXIF/Flash/FunctionPresent	*Booléen*
EXIF Flash Mode	EXIF/Flash/Mode	EXIF Auto Mode
		EXIF Compulsory Flash Firing
		EXIF Compulsory Flash Suppression
		EXIF Unknown
EXIF Flash Red Eye Reduction	EXIF/Flash/RedEyeReduction	*Booléen*

EXIF Flash Return Light	EXIF/Flash/ReturnLight	EXIF Detected
		EXIF No Detection Function
		EXIF Not Detected
		EXIF Reserved
EXIF F Number	EXIF/FNumber	*Réel*
EXIF Focal Len In 35 mm Film	EXIF/FocalLenIn35mmFilm	*Entier long*
EXIF Focal Length	EXIF/FocalLength	*Réel*
EXIF Focal Plane Resolution Unit	EXIF/FocalPlaneResolutionUnit	*Entier long*
EXIF Focal Plane X Resolution	EXIF/FocalPlaneXResolution	*Réel*
EXIF Focal Plane Y Resolution	EXIF/FocalPlaneYResolution	*Réel*
EXIF Gain Control	EXIF/GainControl	EXIF High Gain Down
		EXIF High Gain Up
		EXIF Low Gain Down
		EXIF Low Gain Up
		EXIF None
EXIF Gamma	EXIF/Gamma	*Réel*
EXIF Image Unique ID	EXIF/ImageUniqueID	*Texte*
EXIF ISO Speed Ratings	EXIF/ISOSpeedRatings	*Entier long ou Tableau Entier long*
EXIF Light Source	EXIF/LightSource	EXIF Unknown
		EXIF Cloudy
		EXIF Cool White Fluorescent
		EXIF D50
		EXIF D55
		EXIF D65
		EXIF D75
		EXIF Daylight
		EXIF Daylight Fluorescent
		EXIF Day White Fluorescent
		EXIF Fine Weather
		EXIF Flash
		EXIF Light Fluorescent
		EXIF ISOStudio Tungsten
		EXIF Other
		EXIF Shade
		EXIF Standard Light A
		EXIF Standard Light B
		EXIF Standard Light C
		EXIF Tungsten
		EXIF White Fluorescent
EXIF Maker Note	EXIF/MakerNote	*Texte*
EXIF Max Aperture Value	EXIF/MaxApertureValue	*Réel*

EXIF Metering Mode	EXIF/MeteringMode	EXIF Other
		EXIF Average
		EXIF Center Weighted Average
		EXIF Multi Segment
		EXIF Multi Spot
		EXIF Partial
		EXIF Spot
EXIF Pixel X Dimension	EXIF/PixelXDimension	*Entier long*
EXIF Pixel Y Dimension	EXIF/PixelYDimension	*Entier long*
EXIF Related Sound File	EXIF/RelatedSoundFile	*Texte*
EXIF Saturation	EXIF/Saturation	EXIF High
		EXIF Low
		EXIF Normal
EXIF Scene Capture Type	EXIF/SceneCaptureType	EXIF Scene Landscape
		EXIF Night
		EXIF Scene Portrait
		EXIF Standard
EXIF Scene Type	EXIF/SceneType	*Entier long*
EXIF Sensing Method	EXIF/SensingMethod	EXIF Color Sequential Area
		EXIF Color Sequential Linear
		EXIF Not Defined
		EXIF One Chip Color Area
		EXIF Three Chip Color Area
		EXIF Trilinear
		EXIF Two Chip Color Area
EXIF Sharpness	EXIF/Sharpness	EXIF High
		EXIF Low
		EXIF Normal
EXIF Shutter Speed Value	EXIF/ShutterSpeedValue	*Réel*
EXIF Spectral Sensitivity	EXIF/SpectralSensitivity	*Texte*
EXIF Subject Area	EXIF/SubjectArea	*Tableau Entier long (2, 3 ou 4 valeurs)*
EXIF Subject Dist Range	EXIF/SubjectDistRange	EXIF Unknown
		EXIF Close
		EXIF Distant
		EXIF Macro
EXIF Subject Distance	EXIF/SubjectDistance	*Réel*
EXIF Subject Location	EXIF/SubjectLocation	*Tableau Entier long (2 valeurs)*
EXIF User Comment	EXIF/UserComment	*Texte*
EXIF White Balance	EXIF/WhiteBalance	EXIF Auto
		EXIF Manual

GPS

Constantes du thème "Noms des métadonnées images" (paramètre *nomMéta*)	**Valeur**	**Types de valeurs ou constantes du thème "Valeurs des métadonnées images" (paramètre *contenu*)**
GPS Altitude	GPS/Altitude	GPS Above Sea Level
		GPS Below Sea Level
GPS Altitude Ref	GPS/AltitudeRef	GPS Above Sea Level
		GPS Below Sea Level
GPS Area Information	GPS/AreaInformation	*Texte*
GPS Date Time	GPS/DateTime	*Date ou Texte (Datetime XML)*
GPS Dest Bearing	GPS/DestBearing	*Texte (1 caractère)*
GPS Dest Bearing Ref	GPS/DestBearingRef	*Texte (1 caractère)*
GPS Dest Distance	GPS/DestDistance	*Texte (1 caractère)*
GPS Dest Distance Ref	GPS/DestDistanceRef	*Texte (1 caractère)*
GPS Dest Latitude	GPS/DestLatitude	*Texte*
GPS Dest Latitude Deg	GPS/DestLatitude/Deg	*Réel*
GPS Dest Latitude Dir	GPS/DestLatitude/Dir	*Texte (1 caractère)*
GPS Dest Latitude Min	GPS/DestLatitude/Min	*Réel*
GPS Dest Latitude Sec	GPS/DestLatitude/Sec	*Réel*
GPS Dest Longitude	GPS/DestLongitude	*Texte*
GPS Dest Longitude Deg	GPS/DestLongitude/Deg	*Réel*
GPS Dest Longitude Dir	GPS/DestLongitude/Dir	*Texte (1 caractère)*
GPS Dest Longitude Min	GPS/DestLongitude/Min	*Réel*
GPS Dest Longitude Sec	GPS/DestLongitude/Sec	*Réel*
GPS Differential	GPS/Differential	GPS Correction Applied
		GPS Correction Not Applied
GPS DOP	GPS/DOP	*Réel*
GPS Img Direction	GPS/ImgDirection	GPS Magnetic north
		GPS True north
GPS Img Direction Ref	GPS/ImgDirectionRef	GPS Magnetic north
		GPS True north
GPS Latitude	GPS/Latitude	GPS North
		GPS South
GPS Latitude Deg	GPS/Latitude/Deg	*Réel*
GPS Latitude Dir	GPS/Latitude/Dir	GPS North
		GPS South
GPS Latitude Min	GPS/Latitude/Min	*Réel*
GPS Latitude Sec	GPS/Latitude/Sec	*Réel*
GPS Longitude	GPS/Longitude	GPS West
		GPS East
GPS Longitude Deg	GPS/Longitude/Deg	*Réel*
GPS Longitude Dir	GPS/Longitude/Dir	GPS West
		GPS East

GPS Longitude Min	GPS/Longitude/Min	*Réel*
GPS Longitude Sec	GPS/Longitude/Sec	*Réel*
GPS Map Datum	GPS/MapDatum	*Texte*
GPS Measure Mode	GPS/MeasureMode	GPS 2D
		GPS 3D
GPS Processing Method	GPS/ProcessingMethod	*Texte*
GPS Satellites	GPS/Satellites	*Texte*
GPS Speed	GPS/Speed	GPS km h
		GPS miles h
		GPS knots h
GPS Speed Ref	GPS/SpeedRef	GPS km h
		GPS miles h
		GPS knots h
GPS Status	GPS/Status	GPS Measurement in progress
		GPS Measurement Interoperability
GPS Track	GPS/Track	*Réel (0.00..359.99)*
GPS Track Ref	GPS/TrackRef	*Texte (1 caractère)*
GPS Version ID	GPS/VersionID	*Tableau Entier long (4 caractères)*

IPTC

Constantes du thème "Noms des métadonnées images" (paramètre *nomMéta*)	**Valeur**	**Types de valeurs ou constantes du thème "Noms des métadonnées images" (paramètre *contenu*)**
IPTC Byline	IPTC/Byline	*Texte ou Tableau Texte*
IPTC Byline Title	IPTC/BylineTitle	*Texte ou Tableau Texte*
IPTC Caption Abstract	IPTC/CaptionAbstract	*Texte*
IPTC Category	IPTC/Category	*Texte*
IPTC City	IPTC/City	*Texte*
IPTC Contact	IPTC/Contact	*Texte ou Tableau Texte*
IPTC Content Location Code	IPTC/ContentLocationCode	*Texte ou Tableau Texte*
IPTC Content Location Name	IPTC/ContentLocationName	*Texte ou Tableau Texte*
IPTC Copyright Notice	IPTC/CopyrightNotice	*Texte*
IPTC Country Primary Location Code	IPTC/CountryPrimaryLocation-Code	*Texte*
IPTC Country Primary Location Name	IPTC/CountryPrimaryLocation-Name	*Texte*
IPTC Credit	IPTC/Credit	*Texte*
IPTC Date Time Created	IPTC/DateTimeCreated	*Date ou Texte (Datetime XML)*
IPTC Digital Creation Date Time	IPTC/DigitalCreationDateTime	*Date ou Texte (Datetime XML)*
IPTC Edit Status	IPTC/EditStatus	*Texte*
IPTC Expiration Date Time	IPTC/ExpirationDateTime	*Date ou Texte (Datetime XML)*
IPTC Fixture Identifier	IPTC/FixtureIdentifier	*Texte*

IPTC Headline	IPTC/Headline	*Texte*
IPTC Image Orientation	IPTC/ImageOrientation	*Texte*
IPTC Image Type	IPTC/ImageType	*Texte*
IPTC Keywords	IPTC/Keywords	*Texte ou Tableau Texte*
IPTC Language Identifier	IPTC/LanguageIdentifier	*Texte*
IPTC Object Attribute Reference	IPTC/ObjectAttributeReference	*Texte*
IPTC Object Cycle	IPTC/ObjectCycle	*Texte*
IPTC Object Name	IPTC/ObjectName	*Texte*
IPTC Original Transmission Reference	IPTC/OriginalTransmissionReference	*Texte*
IPTC Originating Program	IPTC/OriginatingProgram	*Texte*
IPTC Program Version	IPTC/ProgramVersion	*Texte*
IPTC Province State	IPTC/ProvinceState	*Texte*
IPTC Release Date Time	IPTC/ReleaseDateTime	*Date ou Texte (Datetime XML)*
IPTC Scene	IPTC/Scene	IPTC Action
		IPTC Aerial View
		IPTC Close Up
		IPTC Couple
		IPTC Exterior View
		IPTC Full Length
		IPTC General View
		IPTC Group
		IPTC Half Length
		IPTC Headshot
		IPTC Interior View
		IPTC Movie Scene
		IPTC Night Scene
		IPTC Off Beat
		IPTC Panoramic View
		IPTC Performing
		IPTC Posing
		IPTC Profile
		IPTC Rear View
		IPTC Satellite
		IPTC Single
		IPTC Symbolic
		IPTC Two
		IPTC Under Water
IPTC Source	IPTC/Source	*Texte*
IPTC Special Instructions	IPTC/SpecialInstructions	*Texte*
IPTC Star Rating	IPTC/StarRating	*Entier long*
IPTC Sub Location	IPTC/SubLocation	*Texte*
IPTC Subject Reference	IPTC/SubjectReference	*Entier long ou Tableau Entier long*

IPTC Supplemental Category	IPTC/SupplementalCategory	*Texte ou Tableau Texte*
IPTC Urgency	IPTC/Urgency	*Entier long*
IPTC Writer Editor	IPTC/WriterEditor	*Texte ou Tableau Texte*

TIFF

Constantes du thème "Noms des métadonnées images" (paramètre *nomMéta*)	**Valeur**	**Types de valeurs ou constantes du thème "Valeurs des métadonnées images" (paramètre *contenu*)**
TIFF Artist	TIFF/Artist	*Texte*
TIFF Compression	TIFF/Compression	TIFF Adobe Deflate
		TIFF CCIRLEW
		TIFF CCITT1D
		TIFF DCS
		TIFF Deflate
		TIFF Epson ERF
		TIFF IT8BL
		TIFF IT8CTPAD
		TIFF IT8LW
		TIFF IT8MP
		TIFF JBIG
		TIFF JBIGB&W
		TIFF JBIGColor
		TIFF JPEG
		TIFF JPEG2000
		TIFF JPEGThumbs Only
		TIFF Kodak262
		TIFF Kodak DCR
		TIFF Kodak KDC
		TIFF LZW
		TIFF MDIBinary Level Codec
		TIFF MDIProgressive Transform Codec
		TIFF MDIVector
		TIFF Next
		TIFF Nikon NEF
		TIFF Pack Bits
		TIFF Pentax PEF
		TIFF Pixar Film
		TIFF Pixar Log
		TIFF SGILog
		TIFF SGILog24
		TIFF Sony ARW
		TIFF T4Group3Fax

		TIFF T6Group4Fax
		TIFF Thunderscan
		TIFF Uncompressed
TIFF Copyright	TIFF/Copyright	*Texte*
TIFF Date Time	TIFF/DateTime	*Date ou Texte (Datetime XML)*
TIFF Document Name	TIFF/DocumentName	*Texte*
TIFF Host Computer	TIFF/HostComputer	*Texte*
TIFF Image Description	TIFF/ImageDescription	*Texte*
TIFF Make	TIFF/Make	*Texte*
TIFF Model	TIFF/Model	*Texte*
TIFF Orientation	TIFF/Orientation	TIFF Horizontal
		TIFF Mirror Horizontal
		TIFF Mirror Horizontal And Rotate270CW
		TIFF Mirror Horizontal And Rotate90CW
		TIFF Mirror Vertical
		TIFF Rotate180
		TIFF Rotate270CW
		TIFF Rotate90CW
TIFF Photometric Interpretation	TIFF/PhotometricInterpretation	TIFF Black Is Zero
		TIFF CIELab
		TIFF CMYK
		TIFF Color Filter Array
		TIFF ICCLab
		TIFF ITULab
		TIFF Linear Raw
		TIFF Pixar Log L
		TIFF Pixar Log Luv
		TIFF RGB
		TIFF RGBPalette
		TIFF Transparency Mask
		TIFF White Is Zero
		TIFF YCb Cr
TIFF Resolution Unit	TIFF/ResolutionUnit	TIFF CM
		TIFF Inches
		TIFF MM
		TIFF None
		TIFF UM
TIFF Software	TIFF/Software	*Texte*
TIFF XResolution	TIFF/XResolution	*Réel*
TIFF YResolution	TIFF/YResolution	*Réel*

Valeurs des métadonnées images

Cette liste présente les constantes du thème "Valeurs des métadonnées images" ainsi que leur valeur. Ces constantes peuvent être utilisées dans le paramètre *contenu* des commandes FIXER METADONNEES IMAGE et LIRE METADONNEES IMAGE, en fonction du paramètre *nomMéta* :

Constante	Type	Valeur
EXIF Adobe RGB	Entier long	2
EXIF s RGB	Entier long	1
EXIF Uncalibrated	Entier long	-1
EXIF B	Entier long	6
EXIF Cb	Entier long	2
EXIF Cr	Entier long	3
EXIF G	Entier long	5
EXIF R	Entier long	4
EXIF Unused	Entier long	0
EXIF Y	Entier long	1
EXIF High	Entier long	2
EXIF Low	Entier long	1
EXIF Normal	Entier long	0
EXIF Custom	Entier long	1
EXIF Auto	Entier long	0
EXIF Auto Bracket	Entier long	2
EXIF Manual	Entier long	1
EXIF Action	Entier long	6
EXIF Aperture Priority AE	Entier long	3
EXIF Creative	Entier long	5
EXIF Landscape	Entier long	8
EXIF Exposure Portrait	Entier long	7
EXIF Program AE	Entier long	2
EXIF Shutter Speed Priority AE	Entier long	4
EXIF Digital Camera	Entier long	3
EXIF Film Scanner	Entier long	1
EXIF Reflection Print Scanner	Entier long	2
EXIF Auto Mode	Entier long	3
EXIF Compulsory Flash Firing	Entier long	1
EXIF Compulsory Flash Suppression	Entier long	2
EXIF Unknown	Entier long	0
EXIF Detected	Entier long	3
EXIF No Detection Function	Entier long	0
EXIF Not Detected	Entier long	2
EXIF Reserved	Entier long	1
EXIF High Gain Down	Entier long	4
EXIF High Gain Up	Entier long	2

EXIF Low Gain Down	Entier long	3
EXIF Low Gain Up	Entier long	1
EXIF None	Entier long	0
EXIF Cloudy	Entier long	10
EXIF Cool White Fluorescent	Entier long	14
EXIF D50	Entier long	23
EXIF D55	Entier long	20
EXIF D65	Entier long	21
EXIF D75	Entier long	22
EXIF Daylight	Entier long	1
EXIF Daylight Fluorescent	Entier long	12
EXIF Day White Fluorescent	Entier long	13
EXIF Fine Weather	Entier long	9
EXIF Flash	Entier long	4
EXIF Light Fluorescent	Entier long	2
EXIF ISOStudio Tungsten	Entier long	24
EXIF Other	Entier long	255
EXIF Shade	Entier long	11
EXIF Standard Light A	Entier long	17
EXIF Standard Light B	Entier long	18
EXIF Standard Light C	Entier long	19
EXIF Tungsten	Entier long	3
EXIF White Fluorescent	Entier long	15
EXIF Average	Entier long	1
EXIF Center Weighted Average	Entier long	2
EXIF Multi Segment	Entier long	5
EXIF Multi Spot	Entier long	4
EXIF Partial	Entier long	6
EXIF Spot	Entier long	3
EXIF Scene Landscape	Entier long	1
EXIF Night	Entier long	3
EXIF Scene Portrait	Entier long	2
EXIF Standard	Entier long	0
EXIF Color Sequential Area	Entier long	5
EXIF Color Sequential Linear	Entier long	8
EXIF Not Defined	Entier long	1
EXIF One Chip Color Area	Entier long	2
EXIF Three Chip Color Area	Entier long	4
EXIF Trilinear	Entier long	7
EXIF Two Chip Color Area	Entier long	3
EXIF Close	Entier long	2
EXIF Distant	Entier long	3
EXIF Macro	Entier long	1

GPS Above Sea Level	Entier long	0

GPS Below Sea Level	Entier long	1
GPS Correction Applied	Entier long	1
GPS Correction Not Applied	Entier long	0
GPS Magnetic north	Texte	M
GPS True north	Texte	T
GPS North	Texte	N
GPS South	Texte	S
GPS West	Texte	W
GPS East	Texte	E
GPS 2D	Entier long	2
GPS 3D	Entier long	3
GPS km h	Texte	K
GPS miles h	Texte	M
GPS knots h	Texte	K
GPS Measurement in progress	Texte	A
GPS Measurement Interoperability	Texte	V

IPTC Action	Entier long	11900
IPTC Aerial View	Entier long	11200
IPTC Close Up	Entier long	11800
IPTC Couple	Entier long	10700
IPTC Exterior View	Entier long	11600
IPTC Full Length	Entier long	10300
IPTC General View	Entier long	11000
IPTC Group	Entier long	10900
IPTC Half Length	Entier long	10200
IPTC Headshot	Entier long	10100
IPTC Interior View	Entier long	11700
IPTC Movie Scene	Entier long	12400
IPTC Night Scene	Entier long	11400
IPTC Off Beat	Entier long	12300
IPTC Panoramic View	Entier long	11100
IPTC Performing	Entier long	12000
IPTC Posing	Entier long	12100
IPTC Profile	Entier long	10400
IPTC Rear View	Entier long	10500
IPTC Satellite	Entier long	11500
IPTC Single	Entier long	10600
IPTC Symbolic	Entier long	12200
IPTC Two	Entier long	10800
IPTC Under Water	Entier long	11300

TIFF Adobe Deflate	Entier long	8
TIFF CCIRLEW	Entier long	32771
TIFF CCITT1D	Entier long	2

TIFF DCS	Entier long	32947
TIFF Deflate	Entier long	32946
TIFF Epson ERF	Entier long	32769
TIFF IT8BL	Entier long	32898
TIFF IT8CTPAD	Entier long	32895
TIFF IT8LW	Entier long	32896
TIFF IT8MP	Entier long	32897
TIFF JBIG	Entier long	34661
TIFF JBIGB&W	Entier long	9
TIFF JBIGColor	Entier long	10
TIFF JPEG	Entier long	7
TIFF JPEG2000	Entier long	34712
TIFF JPEGThumbs Only	Entier long	6
TIFF Kodak262	Entier long	262
TIFF Kodak DCR	Entier long	65000
TIFF Kodak KDC	Entier long	32867
TIFF LZW	Entier long	5
TIFF MDIBinary Level Codec	Entier long	34718
TIFF MDIProgressive Transform Codec	Entier long	34719
TIFF MDIVector	Entier long	34720
TIFF Next	Entier long	32766
TIFF Nikon NEF	Entier long	34713
TIFF Pack Bits	Entier long	32773
TIFF Pentax PEF	Entier long	65535
TIFF Pixar Film	Entier long	32908
TIFF Pixar Log	Entier long	32909
TIFF SGILog	Entier long	34676
TIFF SGILog24	Entier long	34677
TIFF Sony ARW	Entier long	32767
TIFF T4Group3Fax	Entier long	3
TIFF T6Group4Fax	Entier long	4
TIFF Thunderscan	Entier long	32809
TIFF Uncompressed	Entier long	1
TIFF Horizontal	Entier long	1
TIFF Mirror Horizontal	Entier long	2
TIFF Mirror Horizontal And Rotate270CW	Entier long	5
TIFF Mirror Horizontal And Rotate90CW	Entier long	7
TIFF Mirror Vertical	Entier long	4
TIFF Rotate180	Entier long	3
TIFF Rotate270CW	Entier long	8
TIFF Rotate90CW	Entier long	6
TIFF Black Is Zero	Entier long	1
TIFF CIELab	Entier long	8
TIFF CMYK	Entier long	5
TIFF Color Filter Array	Entier long	32803

TIFF ICCLab	Entier long	9
TIFF ITULab	Entier long	10
TIFF Linear Raw	Entier long	34892
TIFF Pixar Log L	Entier long	32844
TIFF Pixar Log Luv	Entier long	32845
TIFF RGB	Entier long	2
TIFF RGBPalette	Entier long	3
TIFF Transparency Mask	Entier long	4
TIFF White Is Zero	Entier long	0
TIFF YCb Cr	Entier long	6
TIFF CM	Entier long	3
TIFF Inches	Entier long	2
TIFF MM	Entier long	4
TIFF None	Entier long	1
TIFF UM	Entier long	5