Conversion en 4D v17

Bienvenue dans le manuel "Conversion en 4D v17" qui vous présente les différents points à contrôler avant, pendant et après conversion d'une base 4D v16 en 4D v17.

La conversion d'une base 4D v16 doit s'effectuer sans problèmes en 4D v17 à quelques préconisations près, détaillées dans le chapitre **Principes de conversion**. Une fois la conversion faite, il y a toutefois quelques points à vérifier, comme de **Nouvelles options de compatibilité** et des **Changements de comportement** tant au niveau de l'application qu'au niveau des commandes 4D, qu'il faut appréhender pour utiliser au mieux les nouveautés 4D v17.

Et enfin, ce manuel récapitule les **fonctionnalités obsolètes en 4D v17** que le Développeur doit repérer rapidement pour évaluer les temps de mise en place des nouvelles fonctions.

Note : Certaines modifications décrites dans ce manuel ont été introduites durant le programme 4D v16 "R-release".

Pour les conversions de bases plus anciennes, voire très anciennes, il est souvent nécessaire de faire des conversions avec des versions intermédiaires. Et pour les différents points à contrôler, reportez-vous aux documents de conversion des précédentes versions :

- **4D v16** : "Conversion en 4D v16" et "Fonctionnalités obsolètes en 4D v16".
- **4D v15** : "Conversion en 4D v15" et "Fonctionnalités obsolètes en 4D v15".
- **4D v14** : "Conversion en 4D v14" et "Fonctionnalités obsolètes en 4D v14".
- **4D v13** : "Conversion en 4D v13" et "Fonctionnalités obsolètes en 4D v13".
- **4D v12** : "Fonctionnalités obsolètes en 4D v12" (il n'y a pas eu de document "Conversion" pour cette version).
- **4D v11** : "Conversion en 4D v11 SQL".

Principes de conversion

A faire avant de convertir

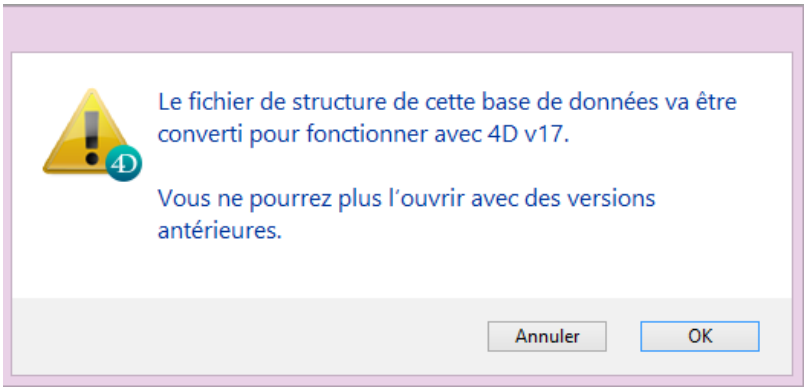
- Vous devez posséder une **version "interprétée**" de la base (fichier xxxx.4DB pour la structure, fichier xxxx.4DD pour les données, fichiers .RSR et .4DR pour les versions antérieures à 4D v11 - voir les documents "Conversion" des versions précédentes), ainsi que le **mot de passe Super-Utilisateur** pour pouvoir faire une conversion ;
- **Faites une copie de votre base** avant conversion ;
- Faites une vérification de votre syntaxe, même si vous ne souhaitez pas compiler. Cette vérification peut vous alerter sur des erreurs ;
- Utilisez le **Centre de Sécurité et de Maintenance** pour vérifier et réparer structure et données ;
- Vérifiez si vous avez des PICTS à l'aide de la commande **LIRE FORMATS IMAGE** et convertissez-les avec la commande **CONVERTIR IMAGE**.
  Dans 4D à compter de la v14, par défaut les codecs QuickTime ne sont plus pris en charge. Par compatibilité, dans une version 32 bits, vous pouvez les réactiver dans votre base à l'aide du sélecteur Prise en charge QuickTime de la commande **FIXER PARAMETRE BASE** . La modification de cette option nécessite le redémarrage de la base. A noter toutefois que la prise en charge de QuickTime sera définitivement supprimée dans les prochaines versions de 4D.
  Pour convertir les images au format obsolète : voir annexe **Conversion des picts en structure**.
- (optionnel) Possibilité de mettre en place des clefs primaire si vous avez besoin de journaliser des données (à partir de la version 14) (cf. **Clé primaire** dans le manuel *Mode Développement*). Il est fortement conseillé de les mettre en place, mais cela peut être fait après la conversion.
- Depuis la version 13.5, vos champs **uniques** doivent obligatoirement être **indexés.** Vous ne serez plus autorisé à créer /modifier un enregistrement d'un champ unique non indexé : la tentative d'enregistrer l'enregistrement génèrera une erreur (-9998 enregistrement unique existe, 1088 L'index est invalide ou manquant).
  Pour créer les index manquants ou générer un fichier disque listant les champs non-indexés, reportez-vous à l'annexe **Pour créer les index manquants**.

Comment convertir

Les bases de données en version 16 de 4D ou 4D Server (ainsi que les bases en v11, v12, v13, v14 et v15) peuvent être converties avec 4D version 17 (fichiers Structure et données). Vous pouvez convertir n'importe quel fichier de Structure interprétée. Pour cela il suffit de lancer 4D v17 et d'ouvrir votre fichier de Structure en interprété, le fichier xxx.4DB.

- **Conversion du fichier de structure (.4DB)**

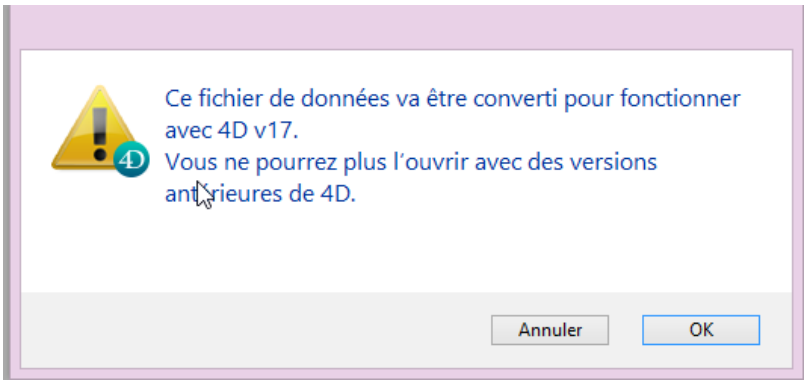
Un dialogue vous avertit de la conversion du fichier de structure et, en fonction de la version de départ, de la conversion du fichier de données :



Votre fichier de structure est converti en 4D v17 et ne pourra plus être ouvert dans une version antérieure.

- **Conversion du fichier de données (.4DD)**

Les données doivent être converties pour des bases en version 4D v14 et antérieures. Dans ce cas, un second dialogue apparaît :

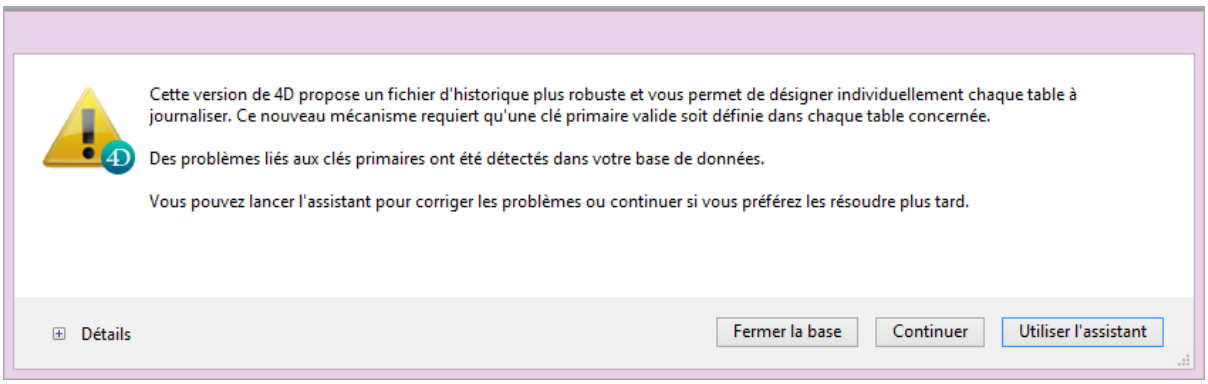


Ce fichier de données est alors converti en version 17 mais pourra toujours être ouvert et utilisé avec 4D v15 ou 4D v16.

Les données n'ont pas besoin d'être converties pour des bases en 4D v15 ou 4D v16.

- **Dialogue de mise en place de l'historique**

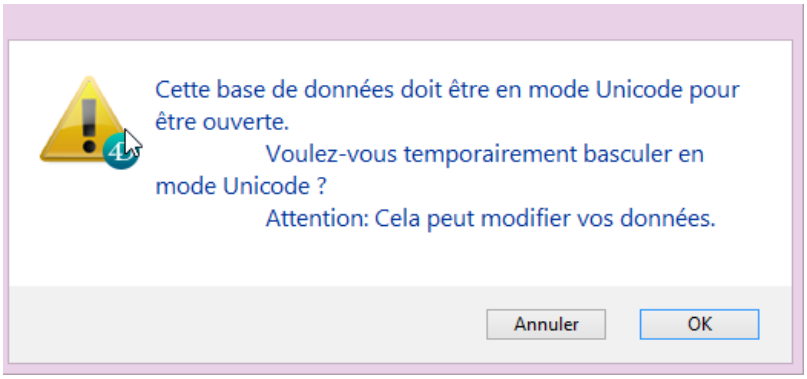
Si les clefs primaires ne sont pas mises en place, 4D vous demandera de le faire en affichant le dialogue ci-dessous :



Il est conseillé de les mettre en place (bouton "Utiliser l'assistant"), mais cette étape peut être faite plus tard (bouton "Continuer"), pour les tables nécessitant une journalisation (pour une sauvegarde). Pour plus d'informations, référez-vous au chapitre **Gestionnaire de clés primaires**.

- **Dialogue de passage temporaire en unicode**

Si vous ouvrez votre application en 4D v17 64 bits, et que votre application de départ n'est pas en unicode, 4D vous proposera de passer temporairement votre base en unicode.



L'unicode permettant un gain très net de rapidité, cette étape aurait dû être franchie depuis plusieurs versions. Si ce n'est pas le cas, faites-le rapidement. Pour plus d'informations, référez-vous au chapitre **Page Compatibilité**.

Après conversion

Utilisez à nouveau le **Centre de Sécurité et de Maintenance** (CSM) pour vérifier et réparer structure et données.

Informations sur la **structure** :

- détection des méthodes orphelines (__Orphan__xxxxx) qui vous sont signalées en **avertissements** dans le compte-rendu du CSM et peuvent être supprimées sans problème à partir de l'Explorateur (après vérification que le code ne peut plus vous servir) ;

- détection des doublons de noms des objets sur les formulaires ne sont plus autorisés : ils vous sont signalés en **avertissements** dans le compte-rendu du CSM. Il suffit de demander une réparation de la base pour que les noms soient modifiés (**attention** dans ce cas à la programmation sur les noms d'objets).

- détection des Images obsolètes (format PICT). **Vérifier l'application** dans le chapitre CSM.
  Voir le paragraphe **Images au format PICT** dans le manuel Fonctionnalités obsolètes ou supprimées. Ces warnings peuvent concerner les images statiques ainsi que les images stockées dans la bibliothèque d'images ou dans les objets de formulaire.

  Note : Pour convertir ces images : voir annexe **Conversion des picts en structure**.

- **Caractères indésirables dans les noms (".", "[", and "]")**
  A compter de 4D v16 R4, l'utilisation de points (.) et/ou de crochets ([ ]) est déconseillée dans les éléments suivants : noms de variables
  - noms de table
  - noms de champs
  - noms de méthodes projet

  Pour aider les développeurs à mettre leur application en conformité avec cette règle, l'action **Vérifier l'application** recherche automatiquement la présence

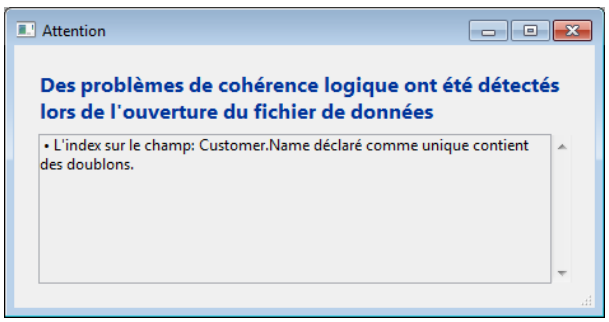
de ces caractères indésirables dans les noms de variables, de tables, de champs et de méthodes. Si ces caractères sont détectés, des "anomalies" sont signalées par le CSM et le fichier de compte rendu contient les avertissements adéquats :

Dans ce cas, il est fortement recommandé de renommer ces éléments dans votre application.

Informations sur les **données** :

- détection de doublons dans des champs uniques :

  Des informations complémentaires sont fournies : Lors de l'utilisation du CSM ou d'une commande telle **VERIFIER FICHIER DONNEES**, les fichiers d'historique générés contiennent désormais les noms des tables et des champs en cause, ainsi que chaque valeur dupliquée.
  A noter : Lors de la saisie de données, la boîte de dialogue d'erreur "Clé dupliquée" contient désormais les noms de la table et du champ concernés, ainsi que la valeur dupliquée, et la commande **LIRE PILE DERNIERE ERREUR** contient également des informations détaillées sur les éventuels doublons.
  Lorsque 4D ouvre un fichier de données, s'il est nécessaire de construire (ou de reconstruire) un index, les doublons sont désormais automatiquement détectés dans le ou les champ(s) associé(s) déclaré(s) unique(s). Dans ce cas, une boîte de dialogue d'alerte spécifique est affichée avant l'ouverture de la base, fournissant à l'utilisateur les informations nécessaires pour identifier et supprimer les doublons :



Reconstruction des index

La conversion d'une base v16 en v17 ne nécessite pas la reconstruction des index (hormis pour une version japonaise de 4D).

Mais si la mise à jour se fait d'une version plus ancienne (notamment si la mise à jour de la bibliothèque Unicode - ICU - International Components for Unicode - est nécessaire), tous les index des champs de type texte et de mots-clés de 4D peuvent être reconstruits. Cette opération s'effectue automatiquement à la première ouverture de la base (attention : la durée de l'opération peut être conséquente).

Note : depuis 4D v16, nous avons optimisé de façon importante l'algorithme de réindexation globale de la base de données. Tout le processus a été revu, et l'opération peut s'effectuer désormais jusqu'à deux fois plus rapidement. Une réindexation globale est nécessaire, par exemple, après une réparation de la base de données ou lorsque le fichier .4dindx a été supprimé**.**

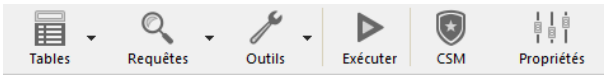
Note de compatibilité

Il est possible d'ouvrir une base de données en version 16 Rx avec une version 4D v16.x et vice versa. Simplement le code utilisant les nouvelles fonctionnalités ne fonctionnera pas et devra être désactivé. Mais une base convertie en v17 ne peut plus être ouverte en 4D v16.

Note : De même, une base v17 Rx pourra également être rouverte avec une version 4D v17.x.

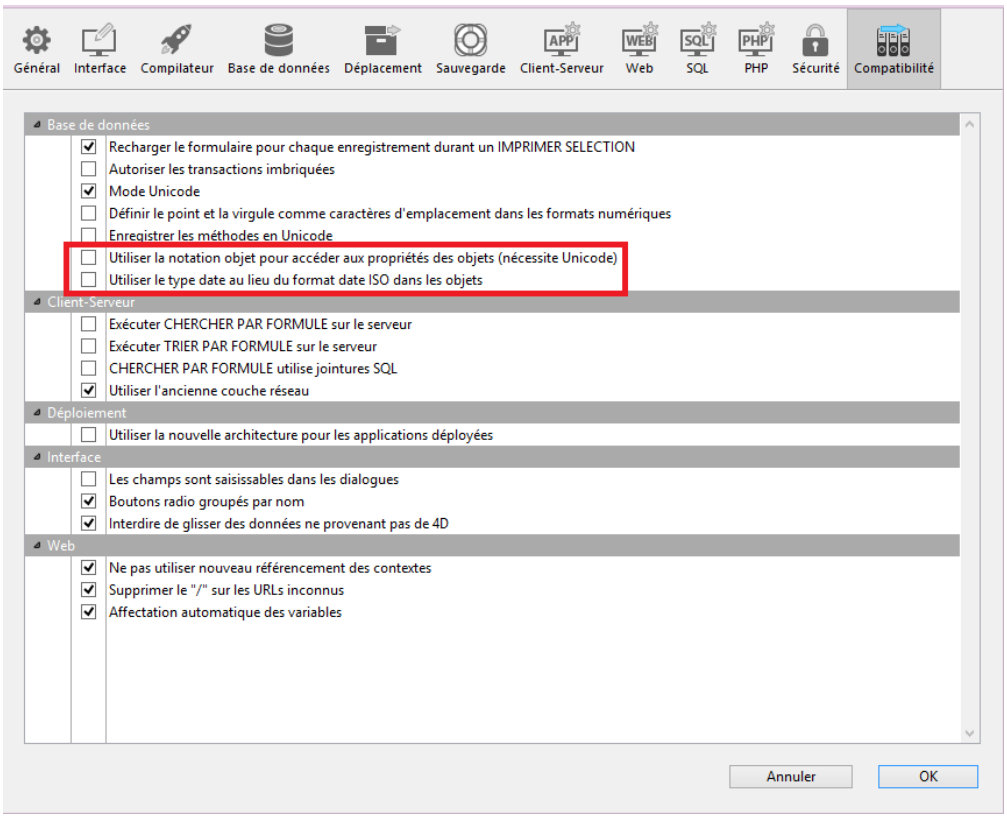
Dialogue Compatibilité

Pour obtenir ce dialogue, il suffit de cliquer, à partir du bandeau principal, sur l'icône "Propriété" :



puis sur l'onglet "Compatibilité".

Deux nouvelles options ont été ajoutées en v17 :



Utiliser la notation objet pour accéder aux propriétés des objets

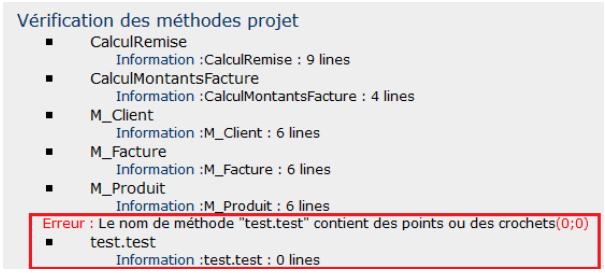
Si vous souhaitez bénéficier de la notation objet (voir section **Utiliser la notation objet**), cette option permet l'utilisation des caractères ".", "[" et "]" comme symboles de notation objet, pour accéder à des propriétés tokenisées -- dans les **noms de variables**, de **tables**, de **champ** et de **méthodes-projet**.

Activer cette option est obligatoire dans une base de données créée dans une version antérieure à 4D v17, puisque les signes ".", "[" et "]" étaient auparavant autorisés dans les noms de ces éléments dans les versions précédentes de 4D et sont désormais déconseillés.

Attention :

- cette option nécessite d'être en unicode ;

- en activant cette option, Il est recommandé de vérifier la conformité de votre code existant grâce au CSM (voir **Vérifier l'application**). Pour aider les développeurs à mettre leur application en conformité avec cette règle, l'action **Vérifier l'application** recherche automatiquement la présence de ces caractères indésirables dans les noms de variables, de tables, de champs et de méthodes. Si ces caractères sont détectés, des erreurs sont signalées par le CSM :



Dans ce cas, il faut renommer ces éléments dans votre application.

Utiliser le type date au lieu du format date ISO dans les objets

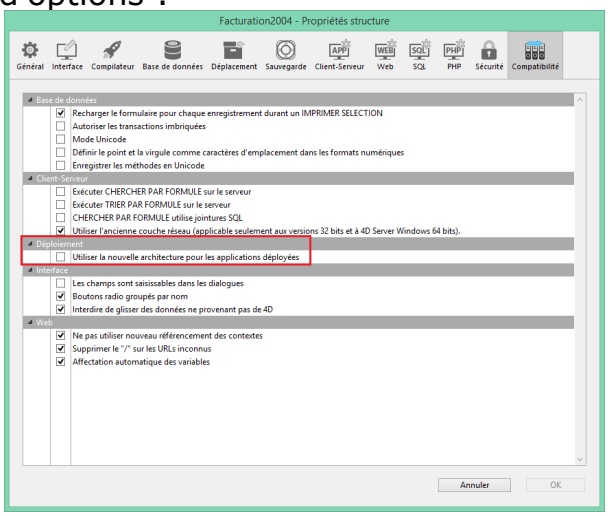
Dans un objet, les dates peuvent maintenant être définies comme des dates et non comme des chaînes, rendant l'utilisation des dates dans les objets plus facile et plus intuitive. Ainsi **OB FIXER** et **OB Lire** peuvent être utilisées sans nécessiter la constante Est une date, et la notation objet peut leur être appliquée comme n'importe quel autre objet. Pour utiliser les dates dans les objets dans une base convertie, cochez simplement cette option "Utiliser le type date au lieu du format date ISO dans les objets". Voir **Page Compatibilité**.

Option supprimée

L'option "**Utiliser les commentaires 4DVAR au lieu des crochets**" est supprimée du dialogue **Compatibilité** depuis 4D v16.

Anciennes options de compatibilité

Plus votre base a été créée dans une ancienne version, plus vous visualiserez d'options :



Pour plus d'informations sur ces options, cf. **Page Compatibilité**.

Page Compatibilité

La page Compatibilité regroupe les paramètres relatifs au maintien de la compatibilité avec les versions précédentes de 4D. A noter que le nombre d'options affichées dépend de la version de 4D avec laquelle la base de données a été créée à l'origine (2004.x, v11, v12...) ainsi que des paramétrages modifiés dans cette base.

Note : Cette page n'apparaît pas pour les bases de données créées avec une version courante de 4D (bases non converties).

- **Les champs sont saisissables dans les dialogues** : dans les versions précédentes de 4D, il n'était pas possible de saisir des valeurs par l'intermédiaire de champs dans des boîtes de dialogue (affichées par exemple à l'aide de la commande **DIALOGUE**). Cette limitation a été supprimée depuis 4D 2004. Vous pouvez toutefois souhaiter conserver le fonctionnement précédent, notamment si votre base de données utilise des champs dans des boîtes de dialogue pour afficher des données. Par défaut, cette option est cochée pour les anciennes bases de données converties en version 2004 et non cochée pour les bases de données créées à partir de la version 2004.
- **Boutons radio groupés par nom** : dans les versions précédentes de 4D, le fonctionnement coordonné d'un groupe de boutons radio était obtenu en donnant une première lettre identique aux variables associées aux boutons (par exemple *m_bouton1*, *m_bouton2*, *m_bouton3*, etc.). Ce principe a été remplacé depuis 4D 2004 par le suivant : pour fonctionner de manière coordonnée, un ensemble de boutons radio doit simplement être groupé dans l'éditeur de formulaires. Pour plus d'informations sur ce point, reportez-vous à la section **Boutons radio et boutons radio image**.
  Ce nouveau mode est valide pour les boutons radio, les boutons radio 3D et les boutons radio image. Pour des raisons de compatibilité, l'ancien mode est conservé par défaut dans les bases de données converties. Vous pouvez cependant forcer l'utilisation du nouveau mode en désélectionnant cette option. Les bases de données créées à compter de la version 2004 utilisent le nouveau mode.
- **Recharger le formulaire pour chaque enregistrement durant un IMPRIMER SELECTION** : dans les versions précédentes de 4D, le formulaire utilisé lors d'une impression lancée à l'aide de la commande **IMPRIMER SELECTION** était rechargé pour chaque enregistrement. Ce principe permettait de réinitialiser automatiquement tous les paramètres des objets que le développeur avait pu modifier par le langage dans l'événement formulaire Sur impression corps.
  Afin d'optimiser les performances, ce mécanisme a été supprimé à compter de 4D 2004. Le développeur 4D doit désormais réinitialiser lui-même les paramètres qu'il souhaite dans la méthode formulaire — ce fonctionnement est identique à celui des formulaires listes avec l'événement Sur affichage corps. Vous pouvez toutefois conserver l'ancien mécanisme à l'aide de cette option. Les bases de données créées à compter de la version 2004 utilisent le nouveau mode.

- **Ne pas utiliser nouveau référencement des contextes** : lorsque cette option n'est pas cochée (valeur par défaut), le serveur Web 4D place le numéro de contexte dans l'URL de base des documents HTML envoyés.
  Avec l'ancien système (option sélectionnée), le serveur Web 4D envoie au navigateur le numéro du contexte pour chaque élément d'une page, ce qui ralentit les traitements. Cette option peut cependant être cochée pour des raisons de compatibilité. Notez qu'après l'avoir modifiée, vous devez redémarrer la base afin que le nouveau fonctionnement soit effectif.
- **Supprimer le "/" sur les URLs inconnus** : dans les anciennes versions de 4D, les URLs inconnus (URLs ne correspondant à aucune page ni URLs spéciaux) étaient retournés dans les méthodes bases Sur authentification Web et Sur connexion Web ($1) sans débuter par le caractère "/". Cette particularité a été supprimée depuis 4D 2004. Toutefois, si vous aviez mis en place des mécanismes basés sur ce fonctionnement et souhaitez le conserver, vous pouvez désélectionner cette option.
- **Interdire de glisser des données ne provenant pas de 4D** : à compter de la v11, 4D permet le glisser-déposer de sélections, d'objets ou de fichiers provenant de l'extérieur de 4D, comme par exemple des fichiers image, en mode Application.
  Cette possibilité doit être prise en charge par le code de la base. Dans les bases de données converties depuis une version précédente de 4D, ce principe peut entraîner des dysfonctionnements si le code existant n'est pas adapté.
  Cette option permet d'anticiper ces éventuels dysfonctionnements. Lorsqu'elle est cochée, le déposer d'objets externes est refusé dans les formulaires 4D. A noter toutefois que l'insertion d'objets externes reste possible dans les objets disposant de l'option **Déposer automatique**, lorsque l'application peut interpréter automatiquement les données déposées (texte ou image). Pour plus d'informations, reportez-vous à la section **Glisser Déposer**.
- **Exécuter CHERCHER PAR FORMULE sur le serveur** et **Exécuter TRIER PAR FORMULE sur le serveur** : à compter de 4D v11, pour des raisons d'optimisation, les commandes de recherches et de tris "par formule" sont exécutées sur le serveur ; seul le résultat est retourné au poste client distant. Il s'agit des commandes **CHERCHER PAR FORMULE**, **CHERCHER PAR FORMULE DANS SELECTION** et **TRIER PAR FORMULE**. En cas d'appel direct de variables dans la formule, la requête est calculée avec la valeur de la variable sur le poste client. Par exemple,

  ```
  CHERCHER PAR FORMULE([latable];[latable]lechamp=lavariable)
  ```

  sera exécuté sur le serveur mais avec le contenu de la variable lavariable du client. En revanche, ce principe n'est pas appliqué pour les formules utilisant des méthodes qui, elles-mêmes, font appel à des variables : dans ce cas la valeur des variables est évaluée sur le serveur.
  Dans les bases de données converties, ce fonctionnement peut affecter les algorithmes existants. Par conséquent, par défaut dans ce contexte ces commandes continuent d'être exécutées sur le poste client. Si vous souhaitez bénéficier du nouvel algorithme dans une base convertie, il vous suffit de cocher ces options.
  Note : Cette option peut être définie par process via la commande **FIXER PARAMETRE BASE**.
- **CHERCHER PAR FORMULE utilise jointures SQL** : à compter de 4D v11, les commandes **CHERCHER PAR FORMULE** et **CHERCHER PAR FORMULE DANS SELECTION** effectuent des "jointures" à la manière du SQL. Avec ce principe, il n'est pas nécessaire qu'un lien automatique structurel existe entre la table A et

la table B pour pouvoir utiliser une formule contenant [Table_A]champ_X= [Table_B]champ_Y.
Ce mécanisme pouvant générer des dysfonctionnements dans les applications existantes, il est désactivé par défaut dans les bases de données converties. Il est conseillé de l'activer (après contrôle du code de la base) en cochant cette option afin de bénéficier de l'optimisation des commandes de recherche par formule.
Notes :
  - Lorsque le mode "jointure SQL" est activé, les commandes **CHERCHER PAR FORMULE** et **CHERCHER PAR FORMULE DANS SELECTION** utilisent toutefois les liens automatiques définis dans l'éditeur de structure dans les cas suivants :
    - si la formule ne peut se pas décomposer en éléments de la forme {champ ;comparateur ;valeur}
    - si deux champs de la même table sont comparés.
  - Cette option peut également être définie par process via la commande **FIXER PARAMETRE BASE**.
- **Autoriser les transactions imbriquées** : active la prise en charge des transactions multi-niveaux. A compter de la v11, 4D accepte les transactions imbriquées sur un nombre de niveaux illimité. Ce fonctionnement pouvant générer des dysfonctionnements dans les bases développées avec des versions précédentes de 4D, il est désactivé par défaut dans les bases converties (les transactions restent limitées à un seul niveau). Pour en bénéficier dans une base convertie, vous devez cocher cette option. Cette option n'est pas cochée par défaut. Elle est spécifique à chaque base de données.
Note : Cette option n'a pas d'effet sur les transactions effectuées dans le moteur SQL de 4D. Les transactions SQL sont toujours multi-niveaux.
- **Mode Unicode** : permet d'activer ou de désactiver le mode Unicode pour la base courante. En mode Unicode, le moteur de la base de données, le langage et les menus manipulent nativement les chaînes de caractères en Unicode. En mode non-Unicode (aussi appelé mode compatibilité ASCII), le jeu de caractères ASCII est utilisé.
Cette option permet de préserver la compatibilité des bases converties. Elle est cochée par défaut pour les bases créées avec 4D v11 et suivantes et désélectionnée dans les bases converties.
Notes :
  - Cette option est spécifique à chaque base de données. Il est donc possible de faire cohabiter une base Unicode avec des composants non Unicode (ou inversement) en mode interprété.
  - Il est également possible de configurer le mode Unicode à l'aide de la commande **FIXER PARAMETRE BASE**.
  - Vous devez redémarrer l'application afin que la modification de cette option soit prise en compte.
  - Les versions 64 bits de 4D ne prennent pas en charge le mode compatibilité ASCII.

  Les spécificités de la prise en charge d'Unicode dans 4D sont détaillées dans le manuel *Langage*. Pour plus d'informations, reportez-vous à la section **EXPORTER TEXTE**.
- **Définir le point et la virgule comme caractères d'emplacement dans les formats numériques** : à compter de la version 11, 4D utilise les paramètres système régionaux pour les formats d'affichage numériques (cf. "Formats numériques" dans la section **Formats d'affichage**). 4D substitue les caractères "," et "." dans les formats d'affichage des numériques par, respectivement, le séparateur des milliers et le séparateur décimal définis dans le système d'exploitation. Le point et la virgule sont alors considérés comme

des caractères d'emplacement, à l'instar du 0 ou du #. Dans les versions précédentes de 4D, les formats d'affichage des numériques ne tenaient pas compte des paramètres régionaux du système. Par exemple, le format "###,##0.00" est un format valide pour un système américain. Toutefois, lorsqu'il est appliqué à une valeur numérique affichée sur un système français ou suisse, le résultat est incorrect.
Dans les bases de données converties, par souci de compatibilité, le nouveau mécanisme n'est pas actif. Pour en bénéficier, vous devez cocher cette option.

- **Affectation automatique des variables** : dans les versions précédentes de 4D, un mécanisme standard du serveur Web permettait de recopier automatiquement dans des variables process 4D la valeur des variables postées via un formulaire HTTP ou un URL de type GET. En mode interprété, la valeur de toute variable reçue était directement copiée dans une variable process 4D du même nom ; en mode compilé, les variables devaient être pré-déclarées dans une méthode projet *COMPILER_WEB*.
A compter de 4D v13.4, ce mécanisme est obsolète et n'est plus disponible dans les nouvelles bases de données. Par compatibilité, il est maintenu dans les bases de données converties mais il est alors possible de le désactiver en désélectionnant cette option de compatibilité. Il est désormais recommandé d'utiliser les commandes dédiées **WEB LIRE VARIABLES** ou **WEB LIRE PARTIE CORPS**.
- **Utiliser l'ancienne couche réseau** : à compter de 4D v15, les applications 4D proposent une nouvelle couche réseau, nommée *ServerNet*, chargée de gérer les communications entre 4D Server et les postes 4D distants (clients). L'ancienne couche réseau devient obsolète, mais est conservée pour assurer la compatibilité des bases existantes. A l'aide de cette option, vous pouvez activer ou désactiver à tout moment l'ancienne couche réseau dans vos applications 4D Server converties en fonction de vos besoins, par exemple au moment de la phase de migration des applications clientes (cf. section **Options réseau et Client-serveur**). *ServerNet* est automatiquement utilisé dans les nouvelles bases, et désactivé dans les bases converties (option cochée).
A noter qu'en cas de modification de l'option, vous devez redémarrer le serveur pour que le changement soit pris en compte. Toute application cliente qui était connectée doit également être redémarrée afin de pouvoir se connecter avec la nouvelle couche réseau (la version minimale du client pour utiliser la couche *ServerNet* est 4D v14 R4, cf. section **Options réseau et Client-serveur**).
Note : Cette option peut également être gérée par programmation via la commande **FIXER PARAMETRE BASE**.
- **Enregistrer les méthodes en Unicode :** permet d'enregistrer les chaînes des méthodes 4D en Unicode. Dans les versions de 4D antérieures à la v15, les chaînes présentes dans le code des méthodes (formules, noms de variables et de méthodes, commentaires, etc.) étaient stockées en utilisant les paramètres d'encodage locaux. Ce principe pouvait entraîner des difficultés, en particulier lorsque le code 4D était partagé entre de développeurs situés dans différents pays. Par exemple, si un développeur français écrivait du code contenant des caractères accentués puis envoyait la base à un développeur anglais, les accents étaient perdus. De même, l'échange de code écrit avec des versions japonaises pouvait également être problématique. L'enregistrement des méthodes en Unicode solutionne tous ces écueils et permet l'échange transparent de code 4D contenant des caractères locaux. Nous recommandons fortement l'activation du mode Unicode pour l'enregistrement des méthodes dès que possible dans vos bases converties, en particulier si vous travaillez dans un environnement international.
Notes :

    - Cette fonctionnalité s'applique au langage lui-même et à son interprétation. Certains éditeurs, comme la Liste des propriétés, utilisent toujours l'encodage local et par conséquent peuvent afficher incorrectement certaines chaînes. Cependant, cela n'affecte pas l'exécution du code.
    - Vous pouvez cocher ou désélectionner l'option à tout moment, elle ne s'applique qu'aux méthodes enregistrées ultérieurement.
    - Les versions 64 bits de 4D ne prennent pas en charge le mode compatibilité ASCII.
- **Utiliser le type date au lieu du format date ISO dans les objets :** vous permet de stocker les dates dans les attributs d'objets avec le type date plutôt qu'en texte au format ISO. Dans les versions précédentes de 4D, les dates dans les attributs d'objets pouvaient uniquement être stockées sous forme de chaîne de caractères. A compter de 4D v16 R6, les dates dans les attributs d'objets peuvent être stockées avec le type date. Vous pouvez activer ce fonctionnement en cochant cette option (les dates précédemment stockées au format texte ISO ne seront pas modifiées, mais les nouvelles dates seront stockées avec le type date). Ce paramètre peut également être géré par programmation pour chaque process à l'aide de la commande **FIXER PARAMETRE BASE** et du sélecteur Dates dans les objets.
- **Utiliser la nouvelle architecture pour les applications déployées :** cette option est disponible pour toutes les applications à compter de 4D v15 R4. Elle permet d'activer ou de désactiver de nouveaux mécanismes liés au déploiement des applications 4D (elle doit être définie sur le poste qui génère l'application finale). Les mécanismes contrôlés par cette option sont décrits dans les paragraphes **Dernier fichier de données ouvert**, **Gestion du fichier de données dans les applications finales** et **Gestion de la connexion des applications clientes**. Cette option est désélectionnée par défaut dans les applications converties. Pour bénéficier des nouveaux mécanismes, il est nécessaire de la cocher explicitement.
- **Utiliser la notation objet pour accéder aux propriétés des objets (nécessite Unicode)** : cette option permet l'utilisation des caractères ".", "[" et "]" comme symboles de notation objet, utilisés pour accéder à des propriétés tokenisées -- et non comme partie de nom de table, champ, méthode ou variable. Activer cette option est obligatoire dans une base de données créée dans une version antérieure à 4D v17 si vous souhaitez bénéficier de la notation objet, puisque les signes ".[ ]" étaient autorisés dans les noms de ces éléments dans les versions précédentes de 4D. En activant cette option, vous déclarez que le code de votre base est en conformité avec la notation objet. Il est recommandé de vérifier cette conformité au préalable à l'aide du CSM (voir **Page Vérification**). Pour plus d'informations sur la notation objet, veuillez vous reporter à la section **Utiliser la notation objet**.
  Note : La notation objet requiert que la base fonctionne en mode Unicode (les options **Mode Unicode** et **Enregistrer les méthodes en Unicode** ne doivent pas être désélectionnées).

Changement de comportement

Objet lire action

Les commandes **OBJET Lire action** et **OBJET FIXER ACTION** utilisent désormais des constantes avec valeurs de type chaîne, et non plus de type entier long (quand elles sont utilisées sur des objets du formulaire).

Ancien code :

```
TABLEAU TEXTE($arrObjects;0)
FORM LIRE OBJETS($arrObjects)
Boucle($i;1;Taille tableau($arrObjects))
    Si(OBJET Lire action(*;$arrObjects{$i})=Objet sans action standard)
        OBJET FIXER ACTION(*;$arrObjects{$i};Objet action Annuler)
    Fin de si
Fin de boucle
```

Code après conversion en v17 :

```
TABLEAU TEXTE($arrObjects;0)
FORM LIRE OBJETS($arrObjects)
Boucle($i;1;Taille tableau($arrObjects))
    Si(_o_OBJET Lire action(*;$arrObjects{$i})=_o_Objet sans action standard)
        OBJET FIXER ACTION(*;$arrObjects{$i};_o_Objet action Annuler)
    Fin de si
Fin de boucle
```

Note : Seule la fonction _o_Objet Lire action a été déclarée obsolète car en cas de compilation, il est nécessaire de fixer à l'avance le type du résultat. Ceci a été fait pour que vous pensiez à changer votre code (voir ci-dessous) sinon la compilation serait impossible.

Nouveau code : commande et constantes obsolètes à remplacer par :

```
TABLEAU TEXTE($arrObjects;0)
FORM LIRE OBJETS($arrObjects)
Boucle($i;1;Taille tableau($arrObjects))
    Si(OBJET Lire action(*;$arrObjects{$i})=ak none)
        OBJET FIXER ACTION(*;$arrObjects{$i};ak cancel)
    Fin de si
Fin de boucle
```

Heures exprimées en secondes

A compter de la v17, les heures 4D (type **C_HEURE**) manipulées via les propriétés d'objet sont converties en **nombre de secondes**. Dans les versions précédentes, elles étaient converties en nombre de millisecondes.

Ce changement s'applique à toutes les heures converties depuis et vers les objets ou les collections à l'aide de notation objet, des commandes telles que **OB FIXER** et **OB Lire**, **CHERCHER PAR ATTRIBUT**, ou des commandes JSON telles que **JSON Stringify** et **JSON Parse**. Il impacte également les conversions heures/numérique dans les fonctionnalités suivantes de 4D :

- zones Web (via JavaScript),
- 4D Mobile,
- SQL (fonction **CAST**)

Pour faciliter la migration des bases 4D (en particulier si des heures ont été stockées dans des attributs de champs objet), une nouvelle option de compatibilité permet de rétablir l'ancien fonctionnement pour la session :

```
//réactiver le précédent fonctionnement
//à placer dans la méthode base sur ouverture /sur démarrage serveur
FIXER PARAMETRE BASE(Heures dans les objets;Heures en millisecondes)
```

Modifications XML

- **Résolution des entités externes**
  A partir de 4D v16 R3, pour des raisons de sécurité, la résolution des entités externes n'est pas activée par défaut dans les documents XML (la déclaration d'une entité externe génère une erreur d'analyse). Pour rétablir le fonctionnement précédent, utilisez le nouveau sélecteur XML résolution des entités externes avec la valeur XML activé de la commande **XML FIXER OPTIONS**.

- **Sensibilité à la casse**
  A partir de 4D v16R4, les commandes **DOM Lire element XML** et **DOM Compter elements XML** sont sensibles à la casse par défaut. Un nouveau sélecteur vous permet de revenir à l'ancien fonctionnement : XML DOM sensibilité à la casse avec comme valeurs possibles :
  - XML casse sensible (valeur par défaut) : les commandes tiennent compte de la casse.
  - XML casse insensible : les commandes ne tiennent pas compte de la casse.

4D Write Pro et attributs image

Les attributs "wk image", "wk list style image" et "wk background image" utilisés avec **WP LIRE ATTRIBUTS** ou **OB Lire** retournent l'image elle-même et non son url lorsque la variable image n'est pas déclarée.

Pour retrouver l'url de l'image, utilisez les nouveaux attributs "wk image url", "wk list style image url" et "wk background image url".

Changement de nom ou de thème

4D Write Pro

- WP Lire paragraphes a été renommée **WP Creer plage paragraphes**.
- WP Lire images a été renommée **WP Creer plage images**.

Constantes Actions Standard

Les anciennes constantes numériques du thème "**Action standard**" sont déclarées obsolètes (préfixées "_o_"). A compter de 4D v16 R3, il est recommandé d'utiliser les constantes de type texte, préfixées "ak...".

Fonctionnalités obsolètes

Commandes obsolètes

Pour retrouver toutes les commandes obsolètes dans vos bases, faites simplement une recherche "_o_" dans le **Chercher dans le développement** du Menu Edition ou dans la zone **Chercher** de la barre d'outils. Il est conseillé de les remplacer par les nouvelles commandes ou fonctionnalités. Sachez toutefois que si ces commandes ne sont pas indiquées comme "désactivées", elles continuent *temporairement* à fonctionner.

Pour obtenir la liste de toutes les commandes obsolètes : **Langage : commandes obsolètes et/ou supprimées**

Commandes et constantes obsolètes en 4D v17

- **Commandes 4D**

Nom précédent	Nouveau nom	Commentaires	Obsolète depuis	Statut actuel
Objet lire action	**OBJET Lire action** (ou **_o_OBJET Lire action** en cas de conversion), avec changement de paramètre (on passe d'un paramètre avec valeur de type entier long à un paramètre avec valeur de type chaîne)	En cas de conversion la commande _o_Objet lire action vous indique un changement de comportement (voir le chapitre "Changement de comportement")	v16R3	Obsolète
Image vers gif	**_o_IMAGE VERS GIF**	**IMAGE VERS BLOB**	v16R5	Obsolète
Type document	**_o_Type document**	utilisez la commande **Chemin vers objet**	v16R6	Obsolète
Createur document	**_o_Createur document**	utilisez la commande **Chemin vers objet**	v16R6	Obsolète
CHANGER TYPE DOCUMENT	**_o_Type document**	utilisez la commande **Objet vers chemin**	v16R6	obsolète
CHANGER CREATEUR DOCUMENT	**_o_CHANGER CREATEUR DOCUMENT**	Utilisez la commande **Objet vers chemin**	v16R6	Obsolète
ASSOCIER TYPES FICHIER	**_o_ASSOCIER TYPES FICHIER**	Utiliser UTIs et info.plist	v16R6	Obsolète
Gestalt	**_o_Gestalt**	à remplacer par **Lire information systeme** / **Sur macOS** / **Sur Windows**	v17	Obsolète
PROPRIETES PLATE FORME	**_o_PROPRIETES PLATE FORME**	à remplacer par **Lire information systeme** / **Sur macOS** / **Sur Windows**	v17	Obsolète

- **Constantes 4D**

Nom précédent	Nouveau nom	Commentaires	Obsolète depuis	Statut actuel
lk affichage barre déf hor	_o_lk affichage barre déf hor	**OBJET LIRE BARRES DEFILEMENT**	v16R3	Obsolète
lk affichage barre déf ver	_o_lk affichage barre déf ver	**OBJET LIRE BARRES DEFILEMENT**	v16R3	Obsolète
lk position barre déf hor	_o_lk position barre déf hor	**OBJET LIRE DEFILEMENT**	v16R3	Obsolète
lk position barre déf ver	_o_lk position barre déf ver	**OBJET LIRE DEFILEMENT**	v16R3	Obsolète
lk hauteur pied	_o_lk hauteur pied	**LISTBOX Lire hauteur pieds**	v16R3	Obsolète
lk hauteur entête	_o_lk hauteur entête	**LISTBOX Lire hauteur entetes**	v16R3	Obsolète
_o_Objet xxx	ak xxx	Constantes portant sur les actions standard. Voir notamment les commandes [#cmd id="1457"/] et **OBJET FIXER ACTION**	v16R3	Obsolète

- **Commandes 4D Internet Commands**

Nom précédent	Nouveau nom	Commentaires	Désactivée depuis	Statut actuel
FTP_Progress	retourne une erreur si appelée	Le paramètre *progression* n'est plus pris en charge par les commandes FTP_Append, FTP_Receive, FTP_Send	v16R3	Désactivée

Fonctionnalités désactivées

- Le plug-in **4D Pack** est supprimé à partir de 4D v16 R2.
- **FTP_Progress** : dans les 4D Internet Commands, cette commande est désactivée à partir de 4D v16 R2. Elle retourne désormais une erreur -2201 si elle est appelée.
  Note : le paramètre *progression* n'est plus pris en charge par les commandes :
  FTP_Append
  FTP_Receive
  FTP_Send

Passage de 32 bits à 64 bits

4D v17 fonctionne en 32-bit ou 64-bit, plate-forme macOS ou Windows.

Passer d"une application 4D 32 bits à une version 64 bits exige un travail de préparation.

Toute la ligne de produits 4D 64 bits ne dépend maintenant plus du tout de la librairie *Mac2Win* d'Altura**. Mac2Win d'Altura est désormais totalement supprimé des versions 64 bits de 4D Developer Edition et 4D Volume Desktop, permettant à ces produits d'utiliser pleinement les API Windows modernes :**

- 4D Developer Edition et 4D Volume Desktop Windows 64 bits en v17 (depuis v16R2)
- 4D Server Windows 64 bits en 4D v17 (depuis v16 R4)

Note : Les versions Windows 32 bits de 4D utilisent toujours Mac2Win.

4D Developer Edition 64 bits intègre les nouveaux éditeurs d'étiquettes, d'états rapides et d'import-export, modernes, intuitifs et faciles à utiliser ! Il est possible de lancer plusieurs instances de 4D Developer Edition 64 bits sur son ordinateur sans devoir installer l'application deux fois.

Si votre application fonctionne avec un 4D Server 64 bits Windows ou OS X, la plupart du travail est déjà fait. Les applications monopostes 64 bits peuvent nécessiter des étapes supplémentaires. Cette section fournit une liste à contrôler point par point pour vous aider à vérifier toute les étapes nécessaires, avant et après la mise à niveau.

Plusieurs fonctionnalités ont été mises à jour, désactivées ou même déclarées en obsolescence avec la migration en 64 bits de nos produits. Tous les détails sont listés dans le paragraphe de la section **Utiliser 4D Developer Edition 64 bits**.

Note : Comme pour tout processus de mise à niveau, il est recommandé d'utiliser le CSM et de lancer une vérification avant chaque étape importante, pour vous assurer que les données et la structure sont valides.

Vérifier les plug-ins

La première étape consiste à mettre à jour vos plug-ins (si vous en avez) dans leur version 64 bits :

- **Plug-ins 4D :**
  Tous les plug-ins existent déjà en versions 64 bits, exceptés 4D Write et 4D View.
  - Si votre application utilise 4D Write, vous devez envisager de migrer votre code vers 4D Write Pro. Une bonne pratique est de garder votre code 32 bits existant et de commencer, à part, un nouveau module basé sur 4D Write Pro 64 bits. Si, pendant une période transitoire, vous souhaitez avoir

à la fois vos documents 4D Write et 4D Write Pro, vous devez utiliser 4D v17 32-bit.
    - Si votre application utilise 4D View, vous devrez utiliser 4D View Pro ou d'autres alternatives.
- **Plug-ins tierce partie :**
  Contactez votre éditeur pour obtenir la version 64 bits.

Préparer la mise à niveau dans la version 32 bits

1. Mettez à jour votre application dans la version 32 bits la plus récente, par exemple 4D v17 32 bits.
2. Vérifiez que le mode Unicode est activé.
3. Convertissez toutes vos images PICT/cicn/QuickTime.
   Pour détecter les formats d'image obsolètes dans vos données, vous pouvez utiliser la commande **LIRE FORMATS IMAGE**.
   Vous devez aussi remplacer tous les formats d'image non pris en charge dans la structure de votre base. Une vérification à l'aide du CSM vous permettra de détecter toutes les images dont le format est obsolète dans les fichiers de ressources pour les boutons images, les boutons 3D ainsi que pour les images statiques.
4. Remplacez toutes les fonctionnalités basées sur du XSLT (commandes **_o_XSLT APPLIQUER TRANSFORMATION**, **_o_XSLT FIXER PARAMETRE** ou **_o_XSLT LIRE ERREUR**), avec la commande **TRAITER BALISES 4D** par exemple.
5. Remplacez tout votre code avec **_o_Numéro de police** par du code avec les noms des polices.
6. Supprimez tout code qui crée ou modifie des fichiers de ressources.

A ce stade, vous êtes prêt(e) pour ouvrir votre application avec une version 64 bits de 4D.

Ouvrir et vérifier la base en version 64 bits

1. Ouvrez votre application avec une version 64 bits de 4D Developer Edition.
2. Si vous utilisiez le moteur de rendu Web Kit intégré pour vos zones Web, vérifiez-les car elles sont automatiquement basculées sur le moteur de rendu système (l'accès aux méthodes 4D avec $4d est toujours possible).
3. Si votre code utilise l'Option mode impression Mac de la commande **FIXER OPTION IMPRESSION**, remplacez-la par un appel à **FIXER IMPRIMANTE COURANTE** avec la constante Driver PDF générique.
4. Vérifiez les appels et l'utilisation de l'éditeur d'étiquettes (référez-vous à **Editeur d'étiquettes (64 bits)**).
5. Vérifiez les appels et l'utilisation de l'éditeur d'états rapides (référez-vous à **Etats rapides (64 bits)**).

Votre application est désormais pleinement compatible 64 bits et vous pouvez bénéficier de toutes les nouvelles fonctionnalités de 4D 64 bits.

Bénéficier des fonctionnalités 64 bits

En particulier :

- **L'architecture 64 bits** repousse les limites de la mémoire cache de la base de données. Améliorez les performances de votre base simplement en utilisant un cache plus important.
  Adoptez les **puissantes fonctionnalités** 64 bits telles que les process préemptifs, les objets de formulaire animés, ou de nouvelles fonctionnalités d'impression
  Construisez vos applications avec 4D Runtime Volume License 64 bits.
  Utilisez 4D Server 64 bits Win et Mac OS en version finale - veuillez vous référer aux sections **Utiliser 4D Server 64 bits (OS X)** et **Utiliser 4D Server 64 bits (Windows)**

- **Nouvel éditeur d'états rapides**, compatible avec les fichiers d'états créés avec les versions précédentes. Voir **Editeur d'étiquettes (64 bits)**.

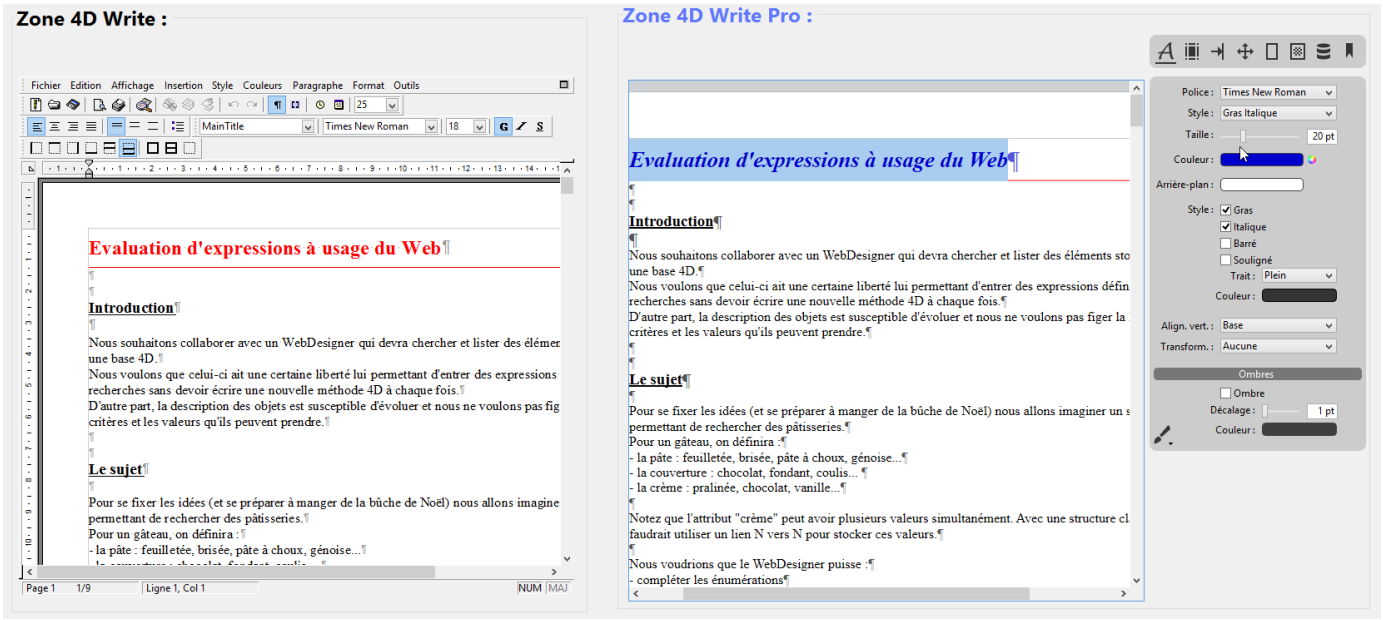
- **Nouvel éditeur d'étiquettes**, compatible avec les fichiers d'étiquettes créés en version précédente. Voir **Etats rapides (64 bits)**.

- Créez des **graphes** en utilisant un paramètre de type Objet avec la commande **GRAPHE**.

Convertir les documents 4D Write en 4D Write Pro

Convertir un document 4D Write

4D Write Pro peut ouvrir et convertir des documents 4D Write en conservant la quasi-totalité de leurs propriétés spécifiques :



Sur l'image ci-dessus, nous avons une zone 4D Write à gauche et une zone 4D Write Pro à droite (créée avec le nouvel objet de la bibliothèque d'objet - voir ci-dessous). Le contenu de la zone 4D Write a été récupéré simplement avec la commande **WP Nouveau** :

```
// on récupère dans la zone 4D Write Pro le contenu de la zone 4D Write
[ZONESWRITE]ZoneNTWP:=WP Nouveau([ZONESWRITE]ZoneNT_)
```

Mais 4D Write n'étant utilisable qu'avec 4D 32 bits, il faut donc que vous convertissiez vos documents 4D Write avant votre passage en 64 bits.

A la différence de 4D Write, 4D Write Pro n'est pas un plug-in mais est intégré à 4D lui-même. Notez que 4D Write Pro utilise la même licence que 4D Write, et donc que celle-ci doit être installée dans votre application pour que la fonctionnalité soit activée.

Les objets 4D Write Pro permettent de récupérer des documents 4D Write de deux manières :

- Pour les fichiers 4D Write stockés sur disque, vous pouvez utiliser la commande **WP Importer document**. Les fichiers 4D Write : .4w6, .4w7 et .4wt (modèle 4D Write), doivent être convertis en fichiers 4D Write Pro (.4wp).

```
// Convertir d'abord les fichiers .4w6 en .4w7 (avec des commandes 4D Write)
$offscreen:=WR Hors ecran
WR OUVRIR DOCUMENT($offscreen;"monFichier.4w6";"4WR6")
WR SAUVER DOCUMENT($offscreen;"monFichier.4w7";"4WR7")
WR DETRUIRE HORS ECRAN($offscreen)
```

```
// puis convertir les .4w7 en .4wp (avec des commandes 4D Write Pro)
C_OBJET($docWritePro)
$docWritePro:=WP Importer document("monFichier.4w7")
WP EXPORTER DOCUMENT($docWritePro;"MonFichier.4wp")
```

- Pour les documents 4D Write stockés dans le fichier de données, vous pouvez utiliser la commande **WP Nouveau**. Les fichiers 4D Write (dans un champ Image ou un champ BLOb) doivent être transférés dans un champ Objet.

```
// d'un champ Image vers un champ Objet, en passant par un BLOb
// [DocWRITE]ZoneWriteImage_ étant un champ Image
// $Blob étant un BLOb
// [DocWRITE]ZoneWritePro étant un champ Objet
$offscreen:=WR Hors ecran
WR IMAGE VERS ZONE($offscreen;[DocWRITE]ZoneWriteImage_)
$Blob:=WR Zone vers blob($offscreen;1)
[DocWRITE]ZoneWritePro:=WP Nouveau($Blob)
WR DETRUIRE HORS ECRAN($offscreen)
```

```
// d'un champ BLOb vers un champ Objet
// [DocWRITE]ZoneWriteBlob_ étant un champ BLOb
// [DocWRITE]ZoneWritePro étant un champ Objet
[DocWRITE]ZoneWritePro:=WP Nouveau([DocWRITE]ZoneWriteBlob_)
```

Notes de compatibilité :

- Seuls les documents 4D Write de dernière génération ("4D Write v7") sont pris en charge.
- Contrôler les fonctionnalités et objets importables en consultant : **Quelles propriétés 4D Write sont importées ?**
- Le copier-coller d'un document 4D Write vers une zone 4D Write Pro n'est pour le moment pas pris en charge. L'importation d'un document 4D Write peut être uniquement effectuée via les commandes du langage de 4D Write Pro.
- Sous Windows, les fonctionnalités de 4D Write Pro s'appuient sur Direct2D. Avec des machines sous Windows 7 ou Windows Server 2008, assurez-vous

que le composant *Platform Update for Windows* a bien été installé, afin de bénéficier de la version requise de Direct2D.

Filtrage des expressions 4D

Ce filtrage n'était pas activé pour les documents 4D Write Pro dans les versions précédentes. Si vos documents 4D Write Pro référençaient des méthodes 4D, elles ne seront plus évaluées correctement une fois les documents convertis en 4D v16 ou plus. Des messages "## Error # 48" seront affichés à la place.
Dans ce cas, vous devrez ajouter les méthodes dans la liste des méthodes autorisées en utilisant la commande **FIXER METHODES AUTORISEES**.

Commandes modifiées

De nouvelles fonctionnalités ont été ajoutées et des commandes existantes ont évolué pour fonctionner avec 4D Write Pro :

- Règle horizontale : pour ajuster marges, retraits et tabulations.
- Barre d'outils personnalisée : extension du mécanisme des actions standard.
- Mise à jour de la commande **Pop up menu dynamique** : pour concevoir votre propre menu contextuel 4D Write Pro basé sur les actions standard.
- Gestion des tableaux : **WP Inserer tableau**, **WP Tableau ajouter ligne**, **WP Tableau lire lignes**, **WP Tableau lire colonnes**, **WP Tableau lire cellules**
- Hyperliens : nouvel attribut wk link url en utilisant les commandes **WP FIXER ATTRIBUTS** et **WP LIRE ATTRIBUTS**.
- Insertion d'image : commande **WP Ajouter image**, et image de fond en pleine page avec la commande **WP FIXER ATTRIBUTS** (attribut wk paper box).
- gestion des en-têtes et pieds de page : **WP Lire entete**, **WP Lire pied**, **WP Lire corps**.
- Commandes pour déplacer le curseur, **WP FIXER CADRE**, et le lire **WP Lire cadre**.
- Gestion des points de suite pour les tabulations (avec la commande **WP FIXER ATTRIBUTS**).

Convertir des documents 4D View en 4D View Pro

Conversion

4D v16 R6 a introduit la toute première étape vers la conversion de vos documents 4D View en 4D View Pro en ***pré-version.*** Grâce à la nouvelle commande **VP Convert from 4D View** la plupart des propriétés et informations stockées dans les documents 4D View sont automatiquement converties, y compris la structure du document, les valeurs, les formats, les styles, les bordures et les formules.

Ces points sont détaillés dans ces pages de la documentation de 4D View Pro :

- **Conversion des documents 4D View**, **Procédure de conversion**
- **Précisions sur la conversion**.

4D View Pro area

Vous disposez d'un nouvel objet de formulaire 4D View Pro (**Objet 4D View Pro**), et de nouvelles commandes associées :

- Créez un nouveau document avec **VP NEW DOCUMENT**,
- Sauvegardez-le sur disque avec **VP EXPORT DOCUMENT** ou dans la base de données en utilisant **VP Export to object**,
- Ré-ouvrez-le ensuite avec **VP IMPORT DOCUMENT** ou **VP IMPORT FROM OBJECT**.

Annexe : Release Notes 4D v16 Rx

Au cours du cycle des versions R-release, diverses bibliothèques et composants utilisés par 4D sont mis à jour et des problèmes connus sont répertoriés. Ces informations, publiées initialement via les "Release Notes" de chaque version, sont résumées dans cette page.

4D v16 R2

4D Developer et 4D Volume Desktop pour Windows 64 bits sont en version finale à partir de 4D v16 R2.

Le plug-in 4D Pack n'est plus proposé dans les installeurs. Notez que 4D Pack pour 4D v16.x est compatible avec 4D v16 R.x

4D v16 R3

L'ancienne librairie Altura a été supprimée de 4D Developer Edition et 4D Volume Desktop Windows 64-bit.

XML : Modification de la résolution des entités externes, nouvelle option pour **XML FIXER OPTIONS** (cf. **Modifications XML**).

4D v16 R4

Xerces : Mise à jour en version 3.1.4 (librairie "Open Source" utilisée en interne pour gérer le XML).

L'ancienne librairie Altura a été supprimée de 4D Server Windows 64 bits. Désormais, toute la ligne de produit 64 bits devient "Altura-free".

XML : A partir de 4D v16 R4, les commandes **DOM Lire element XML** et **DOM Compter elements XML** sont sensibles à la casse par défaut. Le nouveau sélecteur XML DOM sensibilité à la casse de la commande **XML FIXER OPTIONS** permet de revenir à l'ancien fonctionnement (cf. **Modifications XML**).

4D v16 R5

CEF : Mise à jour en version 301. Chromium Embedded Framework (CEF).

libldap : Mise à jour en version 2.45.

libsasl : mise à jour en version 2.1.17. (Simple Authentication Security Layer)

libzip : mise à jour en version 1.2.

zlib : Mise à jour en version 1.2.11.(bibliothèque logicielle de compression de données).

- Ecran HDPI sous Windows 10 et anciens plug-ins :

Les anciens plug-ins 4D Write et 4D View ne sont pas certifiés pour écrans High DPI sous Windows 10 (1709). Nous vous recommandons l'utilisation de la ligne de produits 64 bits (4D Write Pro et 4D View Pro) dans ce contexte.

- Zone web dans 4D 32-bit et macOS 10.13 :

La zone web peut ne pas retrouver le focus en cas de passage de l'arrière-plan vers l'avant-plan. Nous vous recommandons l'utilisation de la ligne de produits 64 bits dans ce contexte.

- Rendu pdf dans une zone web en utilisant 4D 32-bit sous macOS 10.13 :

La zone web échoue parfois à afficher un fichier pdf en utilisant le plugin PDF viewer sous macOS 10.13. Nous vous recommandons l'utilisation de la ligne de produits 64-bit dans ce contexte.

4D v16 R6

Hunspell : Mise à jour en version 1.6.2.

Les attributs "wk image", "wk list style image" et "wk background image" utilisés avec **WP LIRE ATTRIBUTS** ou **OB Lire** retournent l'image elle-même et non son url lorsque la variable image n'est pas déclarée. Pour retrouver l'url de l'image, utilisez les nouveaux attributs "wk image url", "wk list style image url" et "wk background image url".

WP Lire paragraphes a été renommée **WP Creer plage paragraphes**.
WP Lire images a été renommée **WP Creer plage images**.

4D v17

Open SSL : Mise à jour en version v1.0.2n.

CEF : Mise à jour 3282.

SpreadJS : Mise à jour en version v11.

Annexe : Méthodes utiles pour la conversion

Pour générer un fichier disque listant les champs uniques non-indexés

Techtip pour générer un fichier disque listant les champs uniques non-indexés : #command_32, (en anglais) ou ci-dessous code en français :

```
C_ENTIER LONG($maxTableNumber_l;$currentTable_l)
C_ENTIER LONG($maxFieldCount_l;$currentField_l)
C_ENTIER LONG($dontCare_l)   // pour les valeurs du LIRE PROPRIETES CHAMP qui ne sont pas utilisées
C_BOOLEEN($dontCare_f;$isIndexed_f;$isUnique_f)
C_TEXTE($logHeader_t;$logRecord_t;$logfile_t)
C_TEXTE($delim_t;$lf_t)
C_HEURE($logfile_h)
C_TEXTE($tableName_t;$fieldName_t;$note_t)

$delim_t:=Caractere(Tabulation)
$lf_t:=Caractere(Retour chariot)+Caractere(Retour à la ligne)

$logHeader_t:="Champs uniques sans index :"+$lf_t
$logfile_t:=Dossier 4D(Dossier Logs)+"UniqueNonIndex.txt"
$logfile_h:=Creer document($logfile_t)
Si(OK=1)
    ENVOYER PAQUET($logfile_h;$logHeader_t)
    $maxTableNumber_l:=Lire numero derniere table
    Boucle($currentTable_l;1;$maxTableNumber_l)
        Si(Est un numero de table valide($currentTable_l))
            $maxFieldCount_l:=Lire numero dernier champ(Table($currentTable_l))
            Boucle($currentField_l;1;$maxFieldCount_l)
                Si(Est un numero de champ valide($currentTable_l;$currentField_l))
                    LIRE PROPRIETES CHAMP($currentTable_l;$currentField_l;$dontCare_l;\
                    $dontCare_l;$isIndexed_f;$isUnique_f;$dontCare_f)
                    Si(($isUnique_f) & (Non($isIndexed_f)))
                        $tableName_t:=Nom de la table(Table($currentTable_l))
                        $fieldName_t:=Nom du champ(Champ($currentTable_l;$currentField_l))
                        $logRecord_t:="["+$tableName_t+"]"+$fieldName_t+$lf_t
                        ENVOYER PAQUET($logfile_h;$logRecord_t)
                    Fin de si
```

```
                Fin de si
            Fin de boucle
        Fin de si
    Fin de boucle
Fin de si
FERMER DOCUMENT($logfile_h)
```

Pour créer les index manquants

TechTip pour créer les index manquants sur les champs déclarés "Unique" mais non-indexés : #command_33 (en anglais), ou ci-dessous code en français :

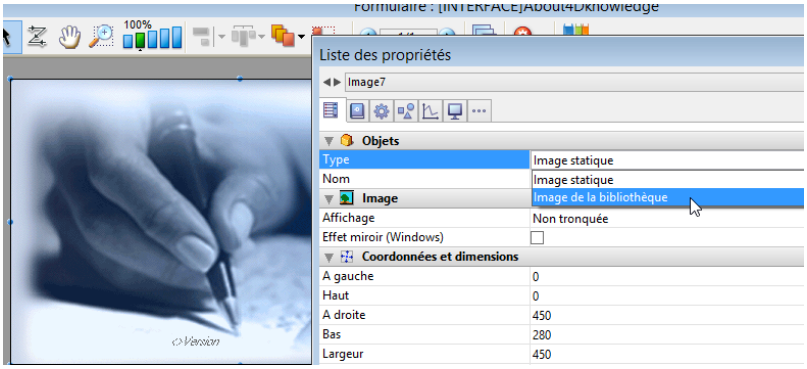
```
C_ENTIER LONG($maxTableNumber_l;$currentTable_l)
C_ENTIER LONG($maxFieldCount_l;$currentField_l)
C_ENTIER LONG($dontCare_l)  // pour les valeurs du LIRE PROPRIETES CHAMP qui ne sont pas utilisées
C_BOOLEEN($dontCare_f;$isIndexed_f;$isUnique_f)
C_TEXTE($logHeader_t;$logRecord_t;$logfile_t)
C_TEXTE($delim_t;$lf_t)
C_TEXTE($tableName_t;$fieldName_t;$note_t)

$maxTableNumber_l:=Lire numero derniere table

Boucle($currentTable_l;1;$maxTableNumber_l)
    Si(Est un numero de table valide($currentTable_l))
        $maxFieldCount_l:=Lire numero dernier champ(Table($currentTable_l))
        boucle($currentField_l;1;$maxFieldCount_l)
            Si(Est un numero de champ valide($currentTable_l;$currentField_l))
                LIRE PROPRIETES CHAMP($currentTable_l;$currentField_l;$dontCare_l;\
                $dontCare_l;$isIndexed_f;$isUnique_f;$dontCare_f)

                Si($isUnique_f) & (Non($isIndexed_f))
                    $tablePtr:=Table($currentTable_l)
                    $fieldPtr:=Champ($currentTable_l;$currentField_l)
                    $tableName_t:=Nom de la table($tablePtr)
                    $fieldName_t:=Nom du champ($fieldPtr)
                    $indexName_t:="["+$tableName_t+"]"+$fieldName_t+" indexé car unicité"
                    TABLEAU POINTEUR($fieldsArray_p;1)
                    $fieldsArray_p{1}:=$fieldPtr
                    CREER INDEX($tablePtr->;$fieldsArray_p;Index BTree standard;$indexName_t;*)
                Fin de si
            Fin de si
        Fin de boucle
```

```
    Fin de si
Fin de boucle
```

Conversion des picts en structure

A faire en version 32-bit (donc avant passage en 64 bits).

1 - transférer vos images statiques sur les formulaires dans la bibliothèque d'images :



2 - une fois vos images tranférées, convertissez-les en .png ou .jpeg :

```
C_ENTIER LONG($i;$SOA;$RIS;$PictRef)
C_TEXTE($PictName)
C_IMAGE($Pict)
 //---------- initialisation tableaux ----------
TABLEAU ENTIER LONG($aL_PictRef;0)
TABLEAU TEXTE($aT_PictName;0)
TABLEAU TEXTE($at_Codecs;0)
LISTE IMAGES DANS BIBLIOTHEQUE($aL_PictRef;$aT_PictName)
$SOA:=Taille tableau($aL_PictRef)
 //---------- conversion des images pict en png ----------
Si($SOA>0)
   Boucle($i;1;$SOA)   // pour chaque image
      $PictRef:=$aL_PictRef{$i}
      $PictName:=$aT_PictName{$i}
      LIRE IMAGE DANS BIBLIOTHEQUE($aL_PictRef{$i};$Pict)
      LIRE FORMATS IMAGE($Pict;$at_Codecs)
      Boucle($j;1;Taille tableau($at_Codecs))
         Si($at_Codecs{$j}=".pict")   // si le format est obsolète
            CONVERTIR IMAGE($Pict;".png")   // conversion en png
 // et stockage dans la bibliothèque
            ECRIRE IMAGE DANS BIBLIOTHEQUE($Pict;$PictRef;$PictName)
         Fin de si
      Fin de boucle
   Fin de boucle
Sinon
   ALERTE("La bibliothèque d'images est vide.")
Fin de si
 //---------- fin de la méthode ----------
```

Applications fusionnées : utilisation des paramètres régionaux

---

Voir **Langue des commandes et des constantes**. Dans le contexte du déploiement d'applications fusionnées, pour utiliser les paramètres régionaux vous pouvez éditer le contenu du fichier de préférences de 4D v1x sur chaque machine locale et fixer la clé "**use_localized_language**" à "**true**". Par exemple :

```
$UserPreference:=Dossier 4D(Dossier 4D actif)+"4D Preferences v17.4DPreferences"
$ref:=DOM Analyser source XML($UserPreference;Vrai)
$refElem:=DOM Chercher element XML($ref;"preferences/com.4d/method_editor/options";arrElementRefs)   // On lit la valeur courante
DOM LIRE ATTRIBUT XML PAR NOM($refElem;"use_localized_language";$attrValue)
Si($attrValue="false")        // Retour au fonctionnement <v15
   DOM ECRIRE ATTRIBUT XML($refElem;"use_localized_language";"true")
Fin de si
DOM EXPORTER VERS FICHIER($ref;$UserPreference)
DOM FERMER XML($ref)
```