4D v11 SQL Release 3 (11.3)
ADDENDUM

Bienvenue dans la release 3 de 4D v11 SQL. Ce document présente les nouveautés et modifications apportées à cette nouvelle version du programme, résumées ci-dessous :

- plusieurs nouveautés concernent les capacités du moteur et du serveur SQL de 4D, avec notamment la prise en charge des **schémas** et les **connexions directes** (sans *driver* ODBC) de bases 4D via le SQL.
- la procédure de **conversion des bases** 4D 2004 en 4D v11 a été modifiée afin de faciliter les conversions successives d'une même structure,
- la **gestion des erreurs** par programmation a été simplifiée : désormais, une seule commande générique, LIRE PILE DERNIERE ERREUR, intercepte toutes les erreurs générées par le moteur de 4D,
- les **échanges de données inter-bases 4D via les services Web** SOAP ont été optimisés,
- les **commandes XML** ont été réorganisées et de nouvelles commandes ont été ajoutées.

Note : 4D v11 SQL r3 s'accompagne d'un nouveau **composant SVG** facilitant l'exploitation du moteur de rendu SVG intégré à 4D. Ce composant fait l'objet d'une documentation séparée.

Moteur SQL

Schémas

Le moteur SQL intégré de 4D v11 SQL implémente le concept de *schémas*. Cette implémentation se traduit par des modifications d'interface et la prise en charge de nouvelles commandes SQL.

La création, la modification et la suppression des schémas s'effectuent via des commandes SQL. Une nouvelle option de l'Inspecteur permet également d'affecter les tables aux schémas.

Présentation

Un schéma est un objet virtuel contenant des tables de la base. Dans le SQL, le concept de schémas a pour but de permettre l'attribution de droits d'accès spécifiques à des ensembles d'objets de la base de données.

Les schémas découpent la base en entités indépendantes dont l'assemblage représente la base entière. Autrement dit, une table appartient toujours à un et un seul schéma.

Lorsqu'une base de données est créée ou convertie avec 4D v11 SQL r3 ou une version ultérieure, un schéma par défaut est créé afin de regrouper toutes les tables de la base. Ce schéma est nommé "DEFAULT_SCHEMA". Il ne peut pas être supprimé ni renommé.

Dans les versions précédentes de 4D, les droits d'accès via le SQL étaient définis globalement pour la base. Désormais, ils seront définis par schémas. Chaque schéma pourra se voir attribuer un type d'accès parmi les suivants :

- Lecture seulement (données)
- Lecture/Ecriture (données)
- Complet (données et structure)

Note : - Lors de la conversion d'une ancienne base en version 11.3 ou suivante, les droits d'accès globaux (tels que définis dans la page SQL des préférences de l'application) sont transférés au schéma par défaut.
- Comme dans les versions précédentes, le contrôle des accès s'applique uniquement aux connexions depuis l'extérieur. Le code SQL exécuté à l'intérieur de 4D via les balises Debut SQL/Fin SQL, SQL EXECUTER, CHERCHER PAR SQL, etc., dispose toujours d'un accès complet.

Seuls le Super_Utilisateur et l'Administrateur de la base peuvent créer, modifier ou supprimer un schéma.

Si le système de gestion des accès de 4D n'est pas activé (c'est-à-dire, si aucun mot de passe n'est assigné au Super_Utilisateur), tous les utilisateurs peuvent créer et modifier des schémas sans restriction.

Création d'un schéma

Les schémas peuvent être créés uniquement par programmation, à l'aide de la commande SQL suivante :

CREATE SCHEMA *Nom_Schema*

Lorsque vous créez un nouveau schéma, par défaut les droits associés sont les suivants :

- Lecture seulement (Données) : Tout le monde
- Lecture/Ecriture (Données) : Tout le monde
- Complet (Données & Structure) : Personne

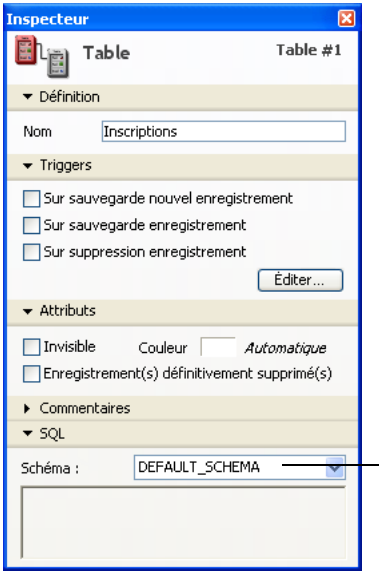
▸ Exemple de création d'un schéma nommé "Droits_Compta" :

```
CREATE SCHEMA Droits_Compta
```

Affectation des tables aux schémas

Chaque table appartient à un seul schéma. Vous pouvez affecter un schéma à une table soit via l'Inspecteur, soit par programmation.

- **Affectation en structure**
  Vous pouvez affecter un schéma à une table dans la zone SQL de l'Inspecteur de table (fenêtre de structure), via un pop up menu listant les schémas définis dans la base :



Menu d'affectation à un schéma

- **Affectation par programmation**
  Il est également possible d'affecter une table à un schéma au moment de sa création à l'aide du langage SQL.

▸ Création d'une table et affectation au schéma MonSchema1 :

CREATE TABLE MonSchema1.MaTable

Si le schéma MONSCHEMA1 n'existe pas, une erreur est retournée et la table est assignée au schéma par défaut.

▸ Création d'une table et affectation au schéma par défaut :

CREATE TABLE MaTable

- **Modification de l'affectation**
  Pour modifier l'affectation courante d'un schéma, vous pouvez utiliser la commande SQL ALTER TABLE :

ALTER TABLE *Nom_Table* SET SCHEMA *Nom_Schema*

▸ Transfert de la table "MaTable" au schéma "MonSchema2" :

ALTER TABLE MaTable SET SCHEMA MonSchema2

Note : Les tables système (_USER_TABLES, _USER_COLUMNS, _USER_INDEXES, _USER_CONSTRAINTS, _USER_IND_COLUMNS et _USER_CONS_COLUMN) sont affectées à un schéma particulier nommé SYSTEM_SCHEMA. Ce schéma ne peut être ni modifié ni supprimé par un utilisateur. Il n'apparaît pas dans la liste des schémas affichée dans l'Inspecteur de table. Il est accessible en mode Lecture seulement à tout utilisateur.

Renommer un schéma

Vous pouvez renommer un schéma à l'aide de la commande SQL ALTER SCHEMA :

ALTER SCHEMA *ancien_nom* RENAME TO *nouveau_nom*

▸ Renommage du schéma "MyFirstSchema" en "MyLastSchema" :

ALTER SCHEMA MyFirstSchema RENAME TO MyLastSchema

Modifier les droits d'accès

Vous pouvez modifier les droits d'accès associés à un schéma à l'aide de la commande SQL GRANT :

GRANT [READ | READ_WRITE | ALL] ON *Nom_Schema* TO *4D_User_Group*

4D_User_Group représente le nom du groupe d'utilisateurs 4D auquel vous souhaitez affecter les droits d'accès au schéma.

Note : 4D permet de définir des noms de groupes comportant des espaces ou des caractères accentués, qui ne sont pas acceptés par le standard SQL. Dans ce cas, vous devez encadrer le nom avec les caractères [et]. Par exemple : GRANT READ ON [le schéma] TO [les admins!]

Les mots-clés READ, READ-WRITE et ALL correspondent aux types d'accès définis dans la page SQL des Préférences :

- READ instaure le mode d'accès Lecture seulement (données)
- READ_WRITE instaure le mode d'accès Lecture/Ecriture (données)
- ALL instaure le mode d'accès complet (données et structure).

▸ Vous souhaitez autoriser l'accès en lecture écriture des données du schéma MonSchema1 au groupe "Power_Users" :

```
GRANT READ_WRITE ON MonSchema1 TO POWER_USERS
```

Supprimer les droits d'accès

Vous pouvez supprimer les droits d'accès spécifiques associés à un schéma à l'aide de la commande SQL REVOKE :

REVOKE [READ | READ_WRITE | ALL] ON *Nom_Schema*

En fait, lorsque vous exécutez cette commande, vous affectez le pseudo-groupe d'utilisateurs *Personne* au droit d'accès défini.

▸ Vous souhaitez supprimer tout droit en lecture écriture au schéma MonSchema1 :

```
REVOKE READ_WRITE ON MonSchema1
```

Suppression des schémas

Il est possible de supprimer tout schéma, à l'exception du schéma par défaut. Lorsque vous supprimez un schéma, toutes les tables qui lui étaient affectées sont transférées au schéma par défaut. Les tables transférées héritent des droits d'accès du schéma par défaut.

La suppression d'un schéma est effectuée à l'aide de la commande SQL DROP SCHEMA :

DROP SCHEMA *Schema_name*

- Vous souhaitez supprimer le schéma MyFirstSchema (auquel sont affectées les tables Table1 et Table2) :

  DROP SCHEMA MyFirstSchema

Après cette opération, les deux tables Table1 et Table2 sont réaffectées au schéma par défaut.

Si vous tentez de supprimer un schéma inexistant ou ayant déjà été supprimé, une erreur est générée.

Intégrité référentielle

4D assure le principe d'intégrité référentielle indépendamment des droits d'accès. Imaginons par exemple que vous disposiez de deux tables Table1 et Table2 reliées par un lien de type N vers 1 (Table2 -> Table1). Table1 appartient au schéma S1 et Table2 au schéma S2.

Un utilisateur disposant des droits d'accès du schéma S1 mais pas de ceux du schéma S2 peut supprimer des enregistrements dans la Table1. Dans ce cas, afin de respecter les principes d'intégrité référentielle, tous les enregistrements de la Table2 liés aux enregistrements supprimés de la Table1 seront également supprimés.

Table système

L'implémentation des schémas SQL dans 4D entraîne l'ajout d'une table système nommée _USER_SCHEMAS :

_USER_SCHEMAS		Décrit les schémas de la base
SCHEMA_ID	INT32	Numéro de schéma
SCHEMA_NAME	VARCHAR	Nom de schéma
READ_GROUP_ID	INT32	Numéro du groupe ayant accès en lecture
READ_GROUP_NAME	VARCHAR	Nom du groupe ayant accès en lecture
READ_WRITE_GROUP_ID	INT32	Numéro du groupe ayant accès en lecture-écriture
READ_WRITE_GROUP_NAME	VARCHAR	Nom du groupe ayant accès en lecture-écriture
ALL_GROUP_ID	INT32	Numéro du groupe ayant un accès complet
ALL_GROUP_NAME	VARCHAR	Nom du groupe ayant un accès complet

Par ailleurs, la colonne SCHEMA_ID a été ajoutée à la table système _USER_TABLES afin de stocker l'appartenance des tables aux schémas.

Importer et exporter des schémas

4D permet d'exporter et d'importer les schémas définis dans une base par l'intermédiaire de tables système. Cette fonction est utile notamment en cas de mise à jour de la structure : il suffit d'importer la définition des schémas dans la nouvelle structure afin qu'elle soit immédiatement opérationnelle.

Pour cela, il suffit d'utiliser la nouvelle table système nommée _USER_SCHEMAS (cf. paragraphe précédent). Les données relatives aux schémas peuvent être stockées dans un BLOB crypté à l'aide d'une méthode du type suivant :

```
Debut SQL
  SELECT * from [_USER_SCHEMAS] INTO Array1,Array2,Array3…;
Fin SQL
VARIABLE VERS BLOB(Array1;SchemaBlob;*)
VARIABLE VERS BLOB(Array2;SchemaBlob;*)
VARIABLE VERS BLOB(Array3;SchemaBlob;*)
…
```

Le BLOB peut ensuite être enregistré dans un fichier texte ou un champ.

L'import des schémas dans une base s'effectuera selon le principe illustré ci-dessous :

```
  `Initialisation des variables tableau
BLOB VERS VARIABLE(SchemaBlob;Array1;Offset)
BLOB VERS VARIABLE(SchemaBlob;Array2;Offset)
BLOB VERS VARIABLE(SchemaBlob;Array3;Offset)
…
  `Remplacement de la table système _USER_SCHEMAS courante par
  `celle qui a été stockée dans le BLOB
$Execute:="INSERT into [_USER_SCHEMAS] ID, SchemaName, ...VALUES
("…)
Debut SQL
  EXECUTE IMMEDIATE $Execute;
Fin SQL
```

Note : Bien entendu, l'utilisateur qui exécute la méthode d'import doit disposer des droits d'accès adéquats (ce doit être le Super_Utilisteur ou l'Administrateur de la base).

INSERT

La commande INSERT bénéficie de deux nouveautés :

- Possibilité d'insérer un contenu de fichier
- Possibilité d'effectuer des insertions multi-lignes

Insertion du contenu de fichiers (INFILE)

La commande INSERT permet désormais d'utiliser le contenu d'un fichier externe pour définir les valeurs d'un nouvel enregistrement. Pour cela, la commande admet le nouveau mot-clé facultatif INFILE :

INSERT INTO {nom_sql | chaîne_sql}
[(ref_colonne, ..., ref_colonne)]
{VALUES([*INFILE*]{expression_arithmétique |NULL}, ...,
[*INFILE*]{expression_arithmétique |NULL}) |sous_requête}

Le mot-clé INFILE doit être utilisé uniquement avec des expressions de type VARCHAR. Lorsque le mot-clé INFILE est passé, la valeur *expression_arithmétique* est évaluée en tant que chemin d'accès de fichier ; si le fichier est trouvé, le contenu du fichier est inséré dans la colonne correspondante. Seuls des champs de type texte ou BLOB peuvent recevoir des valeurs issues d'un INFILE. Le contenu du fichier est transféré sous forme de données brutes, sans interprétation.

Le fichier recherché doit se trouver sur l'ordinateur hébergeant le moteur SQL, même si la requête provient d'un client distant. De même, le chemin d'accès doit être exprimé en respectant la syntaxe du système d'exploitation du moteur SQL. Il peut être absolu ou relatif.

Insertions multi-lignes

Le moteur SQL intégré de 4D admet désormais les insertions multi-lignes de valeurs, ce qui permet d'alléger et d'optimiser le code. L'exécution du code utilisant la syntaxe d'insertion multi-lignes est particulièrement optimisé lors de l'insertion de grandes quantités de données.

La syntaxe des insertions multi-lignes est la suivante :

INSERT INTO {nom_sql | chaîne_sql}
[(ref_colonne, ..., ref_colonne)]
VALUES(expression_arithmétique, ..., expression_arithmétique), ...,
(expression_arithmétique, ..., expression_arithmétique);

Cette syntaxe permet d'éviter la répétition des lignes, nécessaire dans les versions précédentes de 4D. Par exemple :

```
Debut SQL
  INSERT INTO MaTable (Chp1,Chp2,ChpBol,ChpDate,ChpHeure,
         ChpInfo) VALUES (1,1,1,'11/01/01','11:01:01','First row') ;
  INSERT INTO MaTable (Chp1,Chp2,ChpBol,ChpDate,ChpHeure,
         ChpInfo) VALUES (2,2,0,'12/01/02','12:02:02','2nd row'),
.....
  INSERT INTO MaTable (Chp1,Chp2,ChpBol,ChpDate,ChpHeure,
         ChpInfo) VALUES (7,7,1,'17/01/07','17:07:07','7th row');
Fin SQL
```

Désormais, vous pouvez écrire :

```
Debut SQL
  INSERT INTO MaTable
    (Chp1,Chp2,ChpBol,ChpDate,ChpHeure, ChpInfo)
    VALUES
      (1,1,1,'11/01/01','11:01:01','Premiere ligne'),
      (2,2,0,'12/01/02','12:02:02','Deuxième ligne'),
      (3,3,1,'13/01/03','13:03:03','Troisième ligne'),
      .....
      (7,7,1,'17/01/07','17:07:07','Septième ligne');
Fin SQL
```

Vous pouvez également utiliser des variables, par exemple :

```
vid1:=1
vidx1:=1
vbol1:=1
vdate1:= !11/01/01!
vheure1:=?11:01:01?
vtexte1:="Première ligne"
  `Insertion multi-lignes
Debut SQL
  INSERT INTO MaTable
    (Chp1,Chp2,ChpBol,ChpDate,ChpHeure, ChpInfo)
    VALUES
      (:vid1, :vidx1, :vbol1, :vdate1, :vheure1, :vtexte1),
      (2,2,0,'12/01/02','12:02:02','Deuxième ligne'),
      .....
      (7,7,1,'17/01/07','17:07:07','Septième ligne');
Fin SQL
```

Il est également particulièrement intéressant d'utiliser des tableaux avec cette syntaxe :

```
TABLEAU TEXTE(vTabId;0)
TABLEAU TEXTE(vTabIdx;0)
TABLEAU TEXTE(vTabText;0)
TABLEAU BOOLEEN(vTabbol;0)
TABLEAU DATE(vTabdate;0)
TABLEAU ENTIER LONG(vTabL;0)
...
Debut SQL
  INSERT INTO MaTable
    (Chp1,Chp2,ChpBol,ChpDate,ChpHeure, ChpInfo)
    VALUES
      ( :vTabId, :vTabIdx, :vTabbol, :vTabdate, :vTabL, :vTabText);
Fin SQL
```

Note : Vous ne pouvez pas combiner des variables simples et des tableaux dans la même instruction INSERT.

SELECT

La commande SELECT admet désormais des références à des variables 4D dans les clauses LIMIT et OFFSET. Vous pouvez maintenant écrire : SELECT... OFFSET :var1 LIMIT :var2 ...

Serveur SQL

Connexion directe entre deux bases 4D via le SQL

4D v11 SQL r3 permet à une application 4D de se connecter et d'échanger directement des données avec une autre base 4D via le SQL. Dans les versions précédentes de 4D, cette opération nécessitait un transit via le protocole ODBC. Les connexions directes ont pour principal avantage d'accélérer les échanges.

Présentation

Les connexions SQL directes sont basées sur la commande SQL LOGIN[1]. Outre les sources de données ODBC ou la base courante, cette commande permet désormais de désigner directement une base 4D Server comme cible des requêtes SQL effectuées par la suite. La base 4D Server peut être définie via son adresse IP ou son nom de publication.

1. SQL LOGIN est le nouveau nom de la commande existante ODBC LOGIN (cf. paragraphe "Renommage des commandes ODBC", page 13).

Les requêtes SQL concernées sont celles effectuées via la commande SQL EXECUTE ainsi que celles encadrées par les balises Debut SQL / Fin SQL (lorsque le paramètre * est passé). La commande SQL LOGOUT permet ensuite de mettre fin à la session.

Pour une description détaillée de la syntaxe de la commande SQL LOGIN, reportez-vous au paragraphe “SQL LOGIN”, page 15.

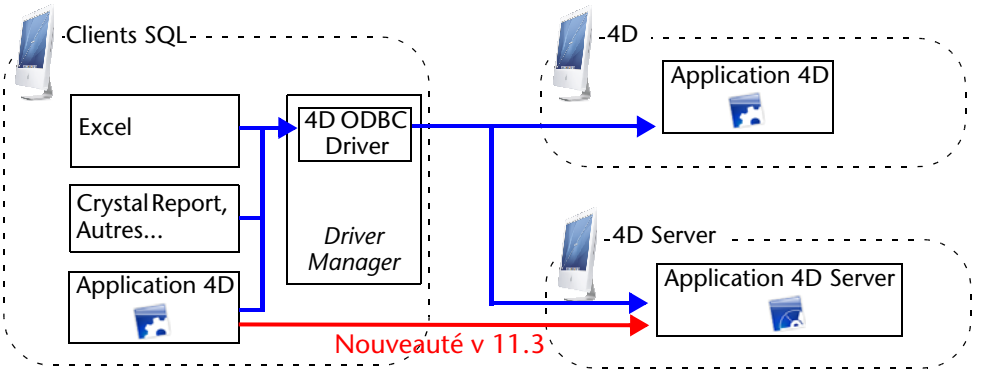
A noter que ces principes ont entraîné une réorganisation des commandes existantes (cf. paragraphe “Renommage des commandes ODBC”, page 13).

Configuration

Seule une application 4D Server v11 SQL peut répondre aux requêtes SQL directes provenant d’autres applications 4D.

De même, seules les applications 4D de la gamme "Professional" peuvent ouvrir une connexion directe vers une autre application 4D.

Les possibilités d’accès externes au serveur SQL de 4D sont désormais les suivantes :



Notes de fonctionnement

Durant une connexion directe, l’échange de données s’effectue automatiquement en mode synchrone, ce qui élimine les questions liées à la synchronisation et à l’intégrité des données.

Une seule connexion est autorisée par process. Si vous souhaitez établir plusieurs connexions simultanément, vous devez créer autant de process que nécessaire.

Les connexions directes sont effectuées via le protocole TCP/IP. Si l’option **Activer SSL** est cochée du côté cible de la connexion (4D Server) dans la page "SQL" des préférences, la communication sera sécurisée.

Les connexions directes ne sont autorisées par 4D Server que si le serveur SQL est démarré.

Encodage des requêtes

L'encodage utilisé par 4D pour les requêtes SQL externes peut désormais être paramétré à l'aide de la commande SQL FIXER OPTION.

SQL FIXER OPTION

SQL FIXER OPTION (option; valeur)

Paramètres	Type		Description
option	Entier long	→	Numéro d'option à définir
valeur	Entier long	→	Nouvelle valeur de l'option

Note : SQL FIXER OPTION est le nouveau nom de la commande ODBC FIXER OPTION (cf. paragraphe "Renommage des commandes ODBC", page 13).

La commande SQL FIXER OPTION permet désormais de définir l'encodage du texte utilisé pour les requêtes envoyées aux sources externes (via le *SQL pass-through*).

Note : La commande SQL LIRE OPTION permet de connaître l'encodage courant.

Pour cela, une nouvelle constante a été ajoutée dans le thème "SQL" : SQL Jeu de caractères (valeur 100). Lorsque vous passez cette constante dans le paramètre *option*, vous devez passer dans le paramètre *valeur* l'identifiant *MIBEnum* du jeu de caractères à utiliser. Les numéros *MIBEnum* sont référencés à l'adresse suivante : http://www.iana.org/assignments/character-sets.

Par exemple, si vous souhaitez utiliser le type d'encodage UTF-7, vous devez exécuter l'instruction suivante :

```
SQL FIXER OPTION(SQL Jeu de caractères;103)
```

Par défaut, 4D utilise en interne l'encodage UTF-8 (valeur 106).

Lorsque vous modifiez l'encodage à l'aide de la commande SQL FIXER OPTION, la modification est effective pour le process courant et la connexion courante.

Si la commande a été exécutée correctement, la variable système OK prend la valeur 1. Sinon, par exemple si vous exécutez la commande SQL FIXER OPTION alors qu'il n'y a pas de connexion externe valide, OK prend la valeur 0.

Renommage des commandes ODBC

L'implémentation des connexions externes directes à des bases 4D Server a entraîné une réorganisation des commandes existantes. En effet, la portée des commandes du thème "Sources de données externes" a été étendue. Un nouveau préfixage et l'abandon de commandes ont été nécessaires.

- Toutes les commandes du thème "Sources de données externes", auparavant préfixées "ODBC", ont été renommées. Elles sont désormais préfixées "SQL". Par exemple, ODBC LOGIN est désormais intitulée SQL LOGIN.
  Cette modification globale a été rendue nécessaire car ces commandes de communication permettent désormais de manipuler des requêtes SQL "directes", n'utilisant pas nécessairement le protocole ODBC.
- Pour plus de clarté, toutes les commandes désormais préfixées "SQL" ont été déplacées dans le thème "SQL". Le thème "Sources de données externes" a été supprimé.
- Les constantes associées à ce thème ont également été préfixées "SQL" et le thème de constantes "Source de données externe" a été renommé "SQL".
- La commande SQL LOGIN va désormais permettre d'aiguiller les requêtes SQL vers la source de donnée requise : base interne, source ODBC ou base 4D Server. La commande SQL LOGOUT, quant à elle, permet de refermer la connexion ouverte précédemment.
  Les commandes UTILISER BASE INTERNE et UTILISER BASE EXTERNE ne sont donc plus nécessaires. Leur utilisation est déconseillée, elles ne seront pas maintenues dans les prochaines versions de 4D.

Le tableau suivant résume ces modifications :

4D v11 SQL	**4D v11 SQL r3**
Commandes	
Thème "Source de données externe"	Supprimé Commandes transférées dans le thème "**SQL**"
ODBC ANNULER CHARGEMENT	**SQL ANNULER CHARGEMENT**
ODBC CHARGER ENREGISTREMENT	**SQL CHARGER ENREGISTREMENT**
ODBC EXECUTER	**SQL EXECUTER**
ODBC EXPORTER	**SQL EXPORTER**
ODBC Fin de selection	**SQL Fin de selection**
ODBC FIXER OPTION	**SQL FIXER OPTION**
ODBC FIXER PARAMETRE	**SQL FIXER PARAMETRE**
ODBC IMPORTER	**SQL IMPORTER**
ODBC LIRE DERNIERE ERREUR	**SQL LIRE DERNIERE ERREUR**
ODBC LIRE OPTION	**SQL LIRE OPTION**
ODBC LOGIN	**SQL LOGIN**
ODBC LOGOUT	**SQL LOGOUT**
Thème "SQL"	
UTILISER BASE INTERNE	Obsolète, utiliser **SQL LOGOUT**
UTILISER BASE EXTERNE	Obsolète, utiliser **SQL LOGIN**
Constantes	
Thème "Source de données externe"	Renommé "**SQL**"
ODBC Asynchrone	**SQL Asynchrone**
ODBC Longueur maxi données	**SQL Longueur maxi données**
ODBC Nombre maxi lignes	**SQL Nombre maxi lignes**
ODBC Paramètre entrée	**SQL Paramètre entrée**
ODBC Paramètre entrée sortie	**SQL Paramètre entrée sortie**
ODBC Paramètre sortie	**SQL Paramètre sortie**
ODBC Timeout connexion	**SQL Timeout connexion**
ODBC Timeout requête	**SQL Timeout requête**
ODBC Tous les enregistrements	**SQL Tous les enregistrements**

SQL LOGIN

SQL LOGIN{(source; utilisateur; motDePasse{;*})}

Paramètres	Type		Description
source	Texte	→	*Nom de la base externe ou* *Adresse IP de la base externe ou* Nom de source de données ODBC ou "" pour afficher le dialogue de sélection
utilisateur	Texte	→	Nom d'utilisateur
motDePasse	Texte	→	Mot de passe de l'utilisateur
*	*	→	*Appliquer à Debut SQL/Fin SQL* *Si omis : non (base interne)* *Si passé : oui*

Note : SQL LOGIN est le nouveau nom de la commande ODBC LOGIN.

La commande SQL LOGIN permet d'ouvrir une connexion avec une source de données SQL, définie dans le paramètre *source*. Elle désigne la cible du code SQL exécuté ultérieurement dans l'application :

- via la commande SQL EXECUTER
- via le code placé à l'intérieur des balises Debut SQL / Fin SQL (si le paramètre * est passé, cf. ci-dessous).

La source de données SQL peut être soit :

- une base 4D Server externe (nouveauté 4D v11 SQL r3),
- une source ODBC externe,
- le moteur SQL interne.

Vous pouvez passer dans *source* l'une des valeurs suivantes :

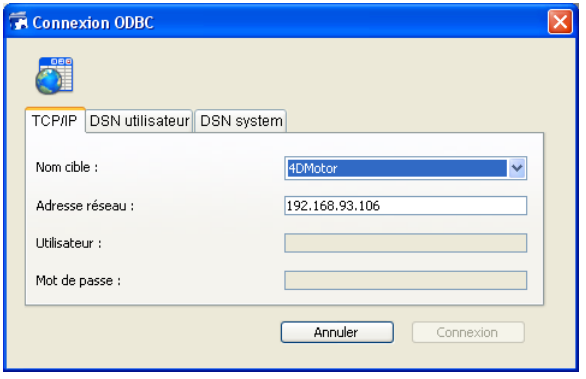
- une **adresse IP**
  Syntaxe : *IP:<Adresse IP>{:<PortTCP>}*
  Dans ce cas, la commande ouvre une connexion directe avec la base 4D Server exécutée sur l'ordinateur ayant l'adresse IP définie. Sur l'ordinateur "cible", le serveur SQL doit être lancé. Si vous passez un numéro de port TCP, il doit avoir été spécifié comme port de publication du serveur SQL dans la base "cible". Si vous ne passez pas de numéro de port TCP, le port par défaut sera utilisé (19812). Le numéro de port TCP du serveur SQL peut être modifié dans la page SQL/Configuration des Préférences de l'application.
  Reportez-vous aux exemples 1 et 2.

- un **nom de publication de base 4D**
  Syntaxe : *4D:<Nom_de_Publication>*
  Dans ce cas, la commande ouvre une connexion directe avec la base 4D Server dont le nom de publication sur le réseau correspond au nom spécifié. Le nom de publication réseau d'une base est défini dans la page Client-Serveur/Configuration des Préférences de l'application. Reportez-vous à l'exemple 4.

Note : Le numéro de port TCP du serveur SQL 4D cible (qui publie la base 4D) et le numéro de port TCP du serveur SQL de l'application 4D ouvrant la connexion doivent être identiques.

- un **nom de source de données ODBC** valide
  Syntaxe : *ODBC:<Ma_DSN>* ou *<Ma_DSN>*
  Dans ce cas, le paramètre *source* contient le nom de la source de données telle qu'elle a été définie dans le gestionnaire du pilote ODBC. Ce principe correspond au fonctionnement précédent de la commande ODBC LOGIN.
  La syntaxe sans le préfixe "ODBC:" a été conservée afin d'assurer la compatibilité avec les versions précédentes de 4D, toutefois pour des raisons de lisibilité du code il est conseillé d'utiliser le préfixe "ODBC:".
  Reportez-vous à l'exemple 4.

- une **chaîne vide**
  Syntaxe : ""
  Dans ce cas la commande provoque l'affichage de la boîte de dialogue de connexion, permettant de désigner manuellement la source de données à laquelle se connecter :



Cette boîte de dialogue comporte plusieurs pages. La page **TCP/IP** se compose des éléments suivants :

- **Nom cible** : ce menu est construit à l'aide de deux listes :
  - la liste des bases ouvertes récemment en connexion directe. Le mécanisme de mise à jour de cette liste est identique à celui de l'application 4D, à la différence près que le dossier contenant les fichiers .4DLink est nommé "Favorites SQL v11" au lieu de "Favorites v11".
  - la liste des applications 4D Server dont le serveur SQL est lancé et dont le port TCP pour les connexions SQL est égal à celui de l'application source. Cette liste est mise à jour dynamiquement à chaque nouvel appel de la commande SQL LOGIN sans le paramètre *source*. Le caractère "^" placé devant un nom de base indique que la connexion est effectuée en mode sécurisé via SSL.
- **Adresse réseau** : cette zone affiche l'adresse IP et éventuellement le port TCP de la base sélectionnée dans le menu Nom cible.
  Vous pouvez également saisir dans cette zone une adresse IP puis cliquer sur le bouton **Connexion** afin de vous connecter à la base 4D Server correspondante. Vous pouvez également spécifier le port TCP, en saisissant deux points (:) puis le numéro du port à la suite de l'adresse. Par exemple : 192.168.93.105:19855
- **Utilisateur** et **Mot de passe** : ces zones permettent de saisir les identifiants de la connexion.

Les pages **DSN utilisateur** et **DSN système** affichent respectivement la liste des sources de données ODBC utilisateur et système définies dans le gestionnaire ODBC de la machine. Ces pages permettent de sélectionner une source de données et de saisir des identifiants afin d'ouvrir une connexion avec une source ODBC externe.

Si la connexion est établie, la variable système OK prend la valeur 1. Sinon, elle prend la valeur 0 et une erreur est générée. Cette erreur peut être interceptée via une méthode de gestion d'erreurs installée par la commande APPELER SUR ERREUR.

- la constante **SQL_INTERNAL**
  Syntaxe : *SQL_INTERNAL*
  Dans ce cas, la commande redirige les requêtes SQL suivantes vers le moteur SQL interne de la base.

Le paramètre facultatif * a été ajouté pour des raisons de compatibilité. Dans les versions précédentes de 4D, la commande ODBC LOGIN n'affectait pas le code SQL inclus dans les balises Debut SQL/Fin SQL (pour cela, il était nécessaire d'utiliser les commandes UTILISER BASE INTERNE et UTILISER BASE EXTERNE). Pour ne pas modifier le fonctionnement des bases existantes, un appel de SQL LOGIN sans le paramètre * ne changera pas la cible du code SQL exécuté au sein des balises Debut SQL/Fin SQL.
Si vous souhaitez que le code placé dans les balises Debut SQL/Fin SQL soit appliqué à la source définie par la commande SQL LOGIN, il est nécessaire de passer le paramètre *.

Note : Dans le cas d'une connexion directe, si vous passez des chaînes vides dans les paramètres *utilisateur* et *motDePasse*, la connexion ne sera acceptée que si les mots de passe 4D ne sont pas activés dans la base cible. Sinon, la connexion est refusée.

Pour refermer la connexion courante et libérer la mémoire, il suffit d'exécuter la commande SQL LOGOUT. Toutes les requêtes SQL sont alors dirigées vers le moteur SQL interne de la base.
Si vous appelez une nouvelle fois SQL LOGIN sans avoir refermé explicitement la connexion courante, elle est automatiquement refermée.

- Exemple 1 : Ouverture d'une connexion directe avec l'application 4D Server v11 SQL exécutée sur le poste ayant l'adresse IP 192.168.45.34 et répondant sur le port TCP par défaut. Les requêtes SQL exécutées via la commande SQL EXECUTE seront redirigées vers cette connexion, les requêtes incluses dans les balises Debut SQL/Fin SQL ne seront pas redirigées.

```
SQL LOGIN("IP:192.168.45.34";"John";"azerty")
```

- Exemple 2 : Ouverture d'une connexion directe avec l'application 4D Server v11 SQL exécutée sur le poste ayant l'adresse IP 192.168.45.34 et répondant sur le port TCP 20150. Les requêtes SQL exécutées via la commande SQL EXECUTE et les requêtes incluses dans les balises Debut SQL/Fin SQL seront redirigées vers cette connexion.

```
SQL LOGIN("IP:192.168.45.34:20150";"John";"azerty";*)
```

- Exemple 3 : Ouverture d'une connexion directe avec l'application 4D Server v11 SQL qui publie sur le réseau local une base dont le nom de publication est "DB_Compta". Le port TCP utilisé pour le serveur SQL des deux bases (défini dans la page SQL/Configuration des Préférences) doit être identique (19812 par défaut). Les requêtes SQL exécutées via la commande SQL EXECUTE seront redirigées vers cette connexion, les requêtes incluses dans les balises Debut SQL/Fin SQL ne seront pas redirigées.

```
SQL LOGIN ("4D:DB_Compta";"John";"azerty")
```

- Exemple 4 : Ouverture d'une connexion via le protocole ODBC avec la source de données externe nommée "MonOracle". Les requêtes SQL exécutées via la commande SQL EXECUTE et les requêtes incluses dans les balises Debut SQL/Fin SQL seront redirigées vers cette connexion.

```
SQL LOGIN ("ODBC:MonOracle";"Scott";"Tiger";*)
```

- Cet exemple illustre les possibilités de connexion offertes par la commande SQL LOGIN :

```
TABLEAU TEXTE (30;aNames)
TABLEAU ENTIER LONG(aAges;0)
SQL LOGIN("ODBC:MonORACLE";"Marc";"azerty")
Si (OK=1)
   `La requête suivante sera redirigée vers la base ORACLE externe
  SQL EXECUTER("SELECT Name, Age FROM PERSONS";aNames; aAges)
   `La requête suivante sera dirigée vers la base 4D locale
  Debut SQL
    SELECT Name, Age
    FROM PERSONS
    INTO :aNames, :aAges;
  Fin SQL
   `La commande SQL LOGIN suivante referme la connexion courante
   `avec la base externe ORACLE et ouvre une nouvelle connexion avec
   `une base externe MySQL
  SQL LOGIN ("ODBC:MySQL";"Jean";"qwerty";*)
  Si (OK=1)
     `La requête suivante sera redirigée vers la base MySQL externe
    SQL EXECUTE("SELECT Name, Age FROM PERSONS";aNames; aAges)
     `La requête suivante sera aussi redirigée vers la base MySQL externe
    Debut SQL
      SELECT Name, Age
      FROM PERSONS
      INTO :aNames, :aAges;
    Fin SQL
```

```
      SQL LOGOUT
        `La requête suivante sera dirigée vers la base 4D locale
      Debut SQL
        SELECT Name, Age
        FROM PERSONS
        INTO :aNames, :aAges;
      Fin SQL
    Fin de si
Fin de si
```

Emplacement des fichiers SSL pour les connexions SQL

Si vous souhaitez que le serveur SQL de 4D utilise le protocole SSL pour traiter les connexions, vous devez désormais :

- cocher l'option **Activer SSL** dans les préférences de l'application (page SSL/Configuration),
- recopier les fichiers *key.pem* et *cert.pem* à l'emplacement suivant : **MaBase/Preferences/SQL** (où "MaBase" représente le dossier/package de la base).

Méthode base Sur Authentification SQL

Le fonctionnement de la méthode base Sur authentification SQL a été modifié afin de renforcer la sécurité du serveur SQL de 4D.

Désormais, pour que la connexion et donc la requête soient acceptées, il est nécessaire que $0 retourne Vrai ET que la commande CHANGER UTILISATEUR COURANT soit appelée et exécutée avec succès.

Dans les versions précédentes de 4D, la connexion était acceptée dès lors que $0 valait Vrai, que la commande CHANGER UTILISATEUR COURANT soit appelée ou non. L'appel de cette commande et son exécution correcte sont désormais obligatoires pour que la requête soit acceptée.

Prise en charge des requêtes Flash Player

La nouvelle option **Autoriser les requêtes Flash Player**, disponible dans la page SQL/Configuration des Préférences, permet d'activer le mécanisme de prise en charge des requêtes Flash Player par le serveur SQL de 4D.

Ce mécanisme est basé sur la présence d'un fichier, nommé "socketpolicy.xml", dans le dossier de préférences de la base (Preferences/SQL/Flash/). Ce fichier est requis par Flash Player afin d'autoriser les connexions *cross-domains* ou les connexions par *sockets* des applications Flex (Web 2.0).

Dans la version précédente de 4D, ce fichier devait être ajouté manuellement. Désormais, l'activation est effectuée via l'option **Autoriser les requêtes Flash Player** : lorsque vous cochez cette option, les requêtes Flash Player sont acceptées et un fichier "socketpolicy.xml" générique est créé si nécessaire pour la base.

Lorsque vous désélectionnez cette option, le fichier "socketpolicy.xml" est désactivé (renommé). Les requêtes Flash Player ultérieures reçues par le serveur SQL sont alors rejetées.

A l'ouverture de la base, l'option est cochée ou non en fonction de la présence d'un fichier "socketpolicy.xml" actif dans le dossier de préférences de la base.

Conversions multiples des bases

Le mécanisme de conversion des bases 4D de la version 200x à la version 11 a été modifié afin de faciliter les conversions successives d'un même fichier de structure (avec une seule conversion du fichier de données).

Ce scénario se produit en cas de développement d'une application en version 2004 avec conversion initiale en v11 puis conversions successives du fichier de structure au cours de la phase de développement, tandis que l'application v11 convertie est en exploitation. Seul le fichier de structure est donc converti par la suite.

Afin de permettre le bon fonctionnement de l'application dans ce contexte, un nouveau fichier, nommé *catalog.xml*, est généré à côté des fichiers v11 au moment des conversions. Ce fichier contient la description de la structure de base convertie ainsi que des nouveaux identifiants internes utilisés en v11. Sa présence est nécessaire pour que vous puissiez effectuer plusieurs conversions de la structure tout en conservant le même fichier de données.

Suppression de la préférence WEDD

Dans les versions précédentes de 4D, il était possible de définir une chaîne de caractères personnalisée afin d'associer un fichier de structure et un fichier de données (option WEDD, "Signature de liaison entre le fichier de structure et le fichier de données" dans les Préférences).

Désormais, ce mécanisme n'est plus utilisé. La liaison entre les fichiers de données et de structure est obligatoire et automatique via un numéro interne. Par conséquent, l'option WEDD a été supprimée des préférences. 4D attribue automatiquement des numéros de liaison internes au moment de la création ou de la conversion des bases.

Gestion des erreurs par programmation

La gestion des erreurs se produisant au cours de l'exécution du code a été optimisée et simplifiée dans 4D v11 SQL r3.

Le principe d'interception des erreurs est inchangé : une méthode installée à l'aide de la commande APPELER SUR ERREUR reçoit et permet de traiter les erreurs. De même, la variable système *Error* permet, à l'intérieur de cette méthode, de connaître le numéro de l'erreur courante.

En revanche, une commande unique permet désormais d'obtenir des informations sur l'erreur interceptée : LIRE PILE DERNIERE ERREUR. Appelée au sein de la méthode de gestion des erreurs, cette commande permet de traiter tous les types d'erreurs, quel que soit le contexte (cf. description ci-dessous).
Dans les versions précédentes de 4D, il était nécessaire d'utiliser plusieurs commandes différentes suivant le contexte d'exécution : LIRE ERREUR XML, LIRE ERREUR XSLT, Lire info erreur web service, etc.

Note de compatibilité : LIRE PILE DERNIERE ERREUR est le nouveau nom de la commande LIRE DERNIERE ERREUR SQL, déjà présente dans les versions précédentes de 4D. Sa portée a été étendue, pour plus de clarté elle a été déplacée dans le thème "Interruptions".

Les autre commandes d'information sur les erreurs sont conservées par compatibilité mais leur intérêt est désormais limité :

- Lire infos erreur Web Service, thème *Web Services (Client)*
- ODBC LIRE DERNIERE ERREUR (renommée SQL LIRE DERNIERE ERREUR), thème *SQL* (cf. paragraphe "Renommage des commandes ODBC", page 13)
- LIRE ERREUR XML, thème *XML Utilitaires*
- LIRE ERREUR XSLT, thème *XML Utilitaires*

LIRE PILE DERNIERE ERREUR

LIRE PILE DERNIERE ERREUR(tabCodes; tabCompInternes; tabLibellés)

Paramètres	Type		Description
tabCodes	Tab Numérique	←	Tableau de numéros d'erreurs
tabCompInternes	Tab Chaîne	←	Tableau de codes de composants internes
tabLibellés	Tab Chaîne	←	Tableau de libellés d'erreurs

Note : LIRE PILE DERNIERE ERREUR est le nouveau nom de la commande LIRE DERNIERE ERREUR SQL.

La commande LIRE PILE DERNIERE ERREUR retourne les informations relatives à la "pile" d'erreurs courante de l'application 4D. Lorsqu'une instruction 4D provoque une erreur, la pile d'erreurs courante contient la description de l'erreur ainsi que les éventuelles erreurs générées en cascade. Par exemple l'erreur du type "disque saturé" entraîne une erreur d'écriture dans le fichier puis une erreur dans la commande de sauvegarde d'enregistrements : la pile contient alors trois erreurs. Si la dernière instruction 4D n'a pas généré d'erreur, la pile d'erreurs courante est vide.

Cette commande générique permet de traiter tous les types d'erreurs pouvant se produire.
A noter toutefois que pour obtenir des informations détaillées relatives aux erreurs générées par une source ODBC, il est nécessaire d'utiliser la commande SQL LIRE DERNIERE ERREUR.

LIRE PILE DERNIERE ERREUR doit être appelée dans le cadre d'une méthode d'appel sur erreur installée par la commande APPELER SUR ERREUR.

Thème : Interruptions

Web services (Client)

Optimisation des Web Services inter-4D

Les échanges de données entre deux applications 4D via les Web Services ont été optimisés dans 4D v11 SQL r3. Cette optimisation se traduit par de nouvelles options pour la commande FIXER OPTION WEB SERVICE ainsi que pour la commande FIXER PARAMETRE BASE (cf. paragraphe "Environnement 4D", page 25).

FIXER OPTION WEB SERVICE

FIXER OPTION WEB SERVICE (option; valeur)

Paramètres	Type		Description
option	Entier long	→	Code de l'option
valeur	Entier long | Texte	→	Valeur de l'option

La commande FIXER OPTION WEB SERVICE vous permet d'activer un mécanisme interne de compression des requêtes SOAP afin d'accélérer les échanges inter-applications 4D.

Pour cela, vous disposez de deux nouvelles constantes, placées dans le thème "Web Services (Client)" : Web Service compression HTTP et Web Service compression deflate.

Pour activer la compression des requêtes SOAP, il vous suffit d'exécuter l'instruction suivante sur le client 4D du Web Service :

FIXER OPTION WEB SERVICE (Web Service compression HTTP; Web Service compression deflate)

Note : "Deflate" est le nom de l'algorithme de compression utilisé en interne par 4D.

Dans ce cas, les données de la prochaine requête SOAP envoyée par le client Web Service seront compressées en utilisant un mécanisme standard HTTP avant leur envoi au serveur SOAP 4D. Le serveur décompressera et analysera la requête puis répondra en utilisant automatiquement le même mécanisme.

Note : Ce mécanisme ne peut pas être utilisé pour des requêtes adressées à un serveur SOAP 4D d'une version antérieure à la 11.3.

Seule la requête suivant l'appel de la commande FIXER OPTION WEB SERVICE est affectée. Vous devez donc appeler cette commande chaque fois que vous voulez utiliser la compression.

Par défaut, 4D ne compresse pas les requêtes HTTP des Web Services.

Afin de vous permettre d'optimiser encore ce fonctionnement, des options supplémentaires configurent le seuil et le taux de compression des requêtes. Ces options sont accessibles via la commande FIXER PARAMETRE BASE (cf. paragraphe "Environnement 4D", page 25).

AUTHENTIFIER WEB SERVICE

AUTHENTIFIER WEB SERVICE (nom; motDePasse{; méthodeAuth}*{;*}*)

Paramètres	Type		Description
nom	Texte	→	Nom
motDePasse	Texte	→	Mot de passe
méthodeAuth	Entier long	→	Méthode d'authentification
*	*	→	*Si passé : authentification par proxy*

La commande AUTHENTIFIER WEB SERVICE admet désormais un astérisque (*) comme dernier paramètre. Lorsque ce paramètre est passé, vous indiquez que les informations d'authentification s'adressent à un proxy HTTP.

Ce paramétrage doit être mis en oeuvre lorsqu'il existe un proxy nécessitant une authentification entre le client du Web Service et le Web Service lui-même. Si le Web Service est lui-même authentifié, une double authentification est requise.

- Authentification auprès d'un Web Service situé derrière un proxy :

```
`Authentification au Web Service en mode DIGEST
AUTHENTIFIER WEB SERVICE("SoapUser";"123";2)
`Authentification au proxy en mode par défaut
AUTHENTIFIER WEB SERVICE("ProxyUser";"456";*)
APPELER WEB SERVICE(...)
```

Environnement 4D

FIXER PARAMETRE BASE, Lire parametre base

Les commandes FIXER PARAMETRE BASE et Lire parametre base admettent deux nouveaux sélecteurs, liés à l'optimisation des échanges Web Services inter-4D (cf. paragraphe "Optimisation des Web Services inter-4D", page 23).

Sélecteur = 50 (Niveau de compression HTTP)

- *Valeurs* : 1 à 9 (1 = plus rapide, 9 = plus compressé) ou -1 = meilleur compromis.
- *Description* : Fixe le niveau de compression pour tous les échanges HTTP compressés effectués pour des Web Services (requêtes client ou réponses serveur). Les échanges compressés sont une optimisation que vous pouvez mettre en oeuvre lorsque vous faites communiquer deux applications 4D via des Web services (cf. commande FIXER OPTION WEB SERVICE, page 24).

Ce sélecteur vous permet d'optimiser les échanges en privilégiant la rapidité d'exécution (compression moindre) ou la quantité de compression (vitesse moindre). Le choix d'une valeur dépend de la taille et de la nature des données échangées. Passez de 1 à 9 dans le paramètre *valeur*, 1 étant la compression la plus rapide et 9 la plus élevée. Vous pouvez également passer -1 pour obtenir un compromis entre vitesse et taux de compression.
Par défaut le niveau de compression est 1 (compression rapide).

Sélecteur = 51 (Seuil de compression HTTP)

- *Valeurs* : Toute valeur de type Entier long
- *Description* : Dans le cadre d'échanges Web Services inter-4D en mode optimisé (cf. ci-dessus), fixe le seuil de taille de requête au-dessous de laquelle les échanges ne doivent pas être compressés. Ce paramétrage est utile pour éviter de perdre du temps machine à compresser les petits échanges.
  Passez dans *valeur* une taille exprimée en octets.
  Par défaut, le seul de compression est fixé à 1024 octets.

XML

Le thème de commandes "XML" a été divisé et des commandes ont été renommées ; en outre, deux nouvelles commandes ont été ajoutées.

Réorganisation des commandes XML

Pour plus de clarté, les commandes du thème "XML" ont été réparties dans trois nouveaux thèmes :

- **XML DOM** : regroupe les commandes DOM
- **XML SAX** : regroupe les commandes SAX
- **XML Utilitaires** : regroupe les commandes XML génériques (commandes XSLT, gestion des erreurs, commandes SVG).

En outre, la commande DOM EXPORTER VERS IMAGE a été renommée SVG EXPORTER VERS IMAGE et placée dans le thème "XML Utilitaires".

Enfin, deux nouvelles commandes XML ont été ajoutées dans 4D v11 SQL r3 (cf. description ci-dessous) :

- DOM Lire element XML racine dans le thème "XML DOM",
- SVG Chercher ID element par coordonnees dans le thème "XML Utilitaires".

DOM Lire element XML racine

DOM Lire element XML racine (refElément) → Chaîne

Paramètres	Type		Description
refElément	Chaîne	→	Référence d'élément XML
Résultat	Chaîne	←	Référence de l'élément racine (16 caractères) ou "" en cas d'erreur

La commande DOM Lire element XML racine retourne une référence vers l'élément racine du document auquel appartient l'élément XML passé en paramètre. Cette référence pourra être utilisée avec les autres commandes d'analyse XML.

Si la commande a été correctement exécutée, la variable système OK prend la valeur 1. Si une erreur se produit, par exemple si la référence de l'élément est invalide, la variable OK prend la valeur 0 et la commande provoque une erreur. La commande retourne dans ce cas une chaîne vide.

Thème : XML DOM

SVG Chercher ID element par coordonnees

SVG Chercher ID element par coordonnees ({*;} objetImage; x; y) → Chaîne

Paramètres	Type		Description
*		→	Si spécifié, objetImage est un nom d'objet (chaîne) Si omis, objetImage est un champ ou une variable
objetImage	Image	→	Nom d'objet (si * spécifié) ou Champ ou Variable (si * omis)
x	Entier long	→	Coordonnée X en pixels
y	Entier long	→	Coordonnée Y en pixels
Résultat	Chaîne	←	ID de l'élément se trouvant à l'emplacement x,y

La commande SVG Chercher ID element par coordonnees retourne l'ID (attribut "id" ou "xml:id") de l'élément XML situé à l'emplacement défini par les coordonnées (*x*,*y*) dans l'image SVG désignée par le paramètre *objetImage*. Cette commande permet notamment de créer des interfaces graphiques interactives utilisant des objets SVG.

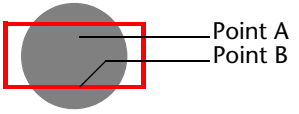
Si vous passez le paramètre optionnel *, vous indiquez que le paramètre *objetImage* est un nom d'objet (une chaîne). Si vous ne passez pas le paramètre, vous indiquez que le paramètre *objetImage* est un champ ou une variable. Dans ce cas, vous ne passez pas une chaîne mais une référence de champ ou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

A noter qu'il n'est pas obligatoire que l'image soit affichée dans un formulaire. Dans ce cas, la syntaxe de type "nom d'objet" n'est pas valide, vous devez passer un nom de champ ou de variable.

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). Dans le contexte d'une image affichée dans un formulaire, 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 et Sur double clic ainsi que (nouveauté v11.3) dans les événements formulaire 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".

Le point pris en compte est le premier point atteint. Par exemple, dans le cas ci-dessous, la commande retournera l'ID du cercle si les coordonnées du point A sont passées et celui du rectangle si les coordonnées du point B sont passées :



Si les coordonnées correspondent à des objets superposés ou composites, la commande retourne l'ID du premier objet disposant d'un attribut ID valide en remontant si nécessaire parmi les éléments parents.

La commande retourne une chaîne vide si :

- la racine est atteinte sans qu'un attribut "id" ait été trouvé,
- le point de coordonnées n'appartient à aucun objet,
- l'attribut "id" est une chaîne vide.

Si *objetImage* ne contient pas une image SVG valide, la commande retourne une chaîne vide et la variable système OK prend la valeur 0. Sinon, si la commande a été exécutée correctement, la variable système OK prend la valeur 1.

Thème : XML Utilitaires

Autres nouveautés

Gestion de la taille des documents

Par défaut, le langage de 4D v11 SQL r3 manipule désormais la taille des documents à l'aide de valeurs de type Réel. Dans les versions précédentes, ces valeurs étaient de type Entier long. Cette nouveauté permet au langage de travailler avec des documents de taille supérieure à 2 Go.

Les commandes suivantes du thème "Documents système" sont concernées par cette modification :

- Taille document (cette commande retourne un réel)
- CHANGER TAILLE DOCUMENT (paramètre *taille*)
- CHANGER POSITION DANS DOCUMENT (paramètre *offset*).

Saisie et affichage multilignes dans les list box sous Mac OS

Il est désormais possible de saisir et d'afficher du texte sur plusieurs lignes au sein d'une même "cellule" de list box sur Mac OS. Dans la version précédente de 4D, cette fonction n'était disponible que sous Windows.

Pour ajouter un retour à la ligne dans une list box sous Mac OS, il suffit d'appuyer sur les touches **Option+Retour chariot**. A noter que la hauteur des lignes n'est pas redimensionnée automatiquement.

Mise à jour de la variable OK par les fonctions statistiques

Lorsque vous exécutez une fonction statistique (par exemple Moyenne) sur une grande sélection de données, un thermomètre de progression apparaît. Ce thermomètre comporte un bouton **Arrêt** permettant à l'utilisateur d'interrompre l'opération.
En conséquence, les fonctions statistiques modifient désormais la variable système OK : si l'opération a été menée à son terme, la variable prend la valeur 1. Si l'utilisateur clique sur le bouton **Arrêt**, la variable OK prend la valeur 0.