

Installation

Bienvenue dans 4D ! Cette page regroupe toutes les informations nécessaires sur l'installation et le lancement de votre produit 4D.

Configuration requise

La page [Téléchargements](#) du site de 4D fournit des informations sur les pré-requis macOS / Windows nécessaires à la gamme 4D.

Des détails techniques supplémentaires sont disponibles sur la [page Ressources](#) du site web de 4D.

Installation sur disque

Les produits 4D sont installés à partir du site web de 4D :

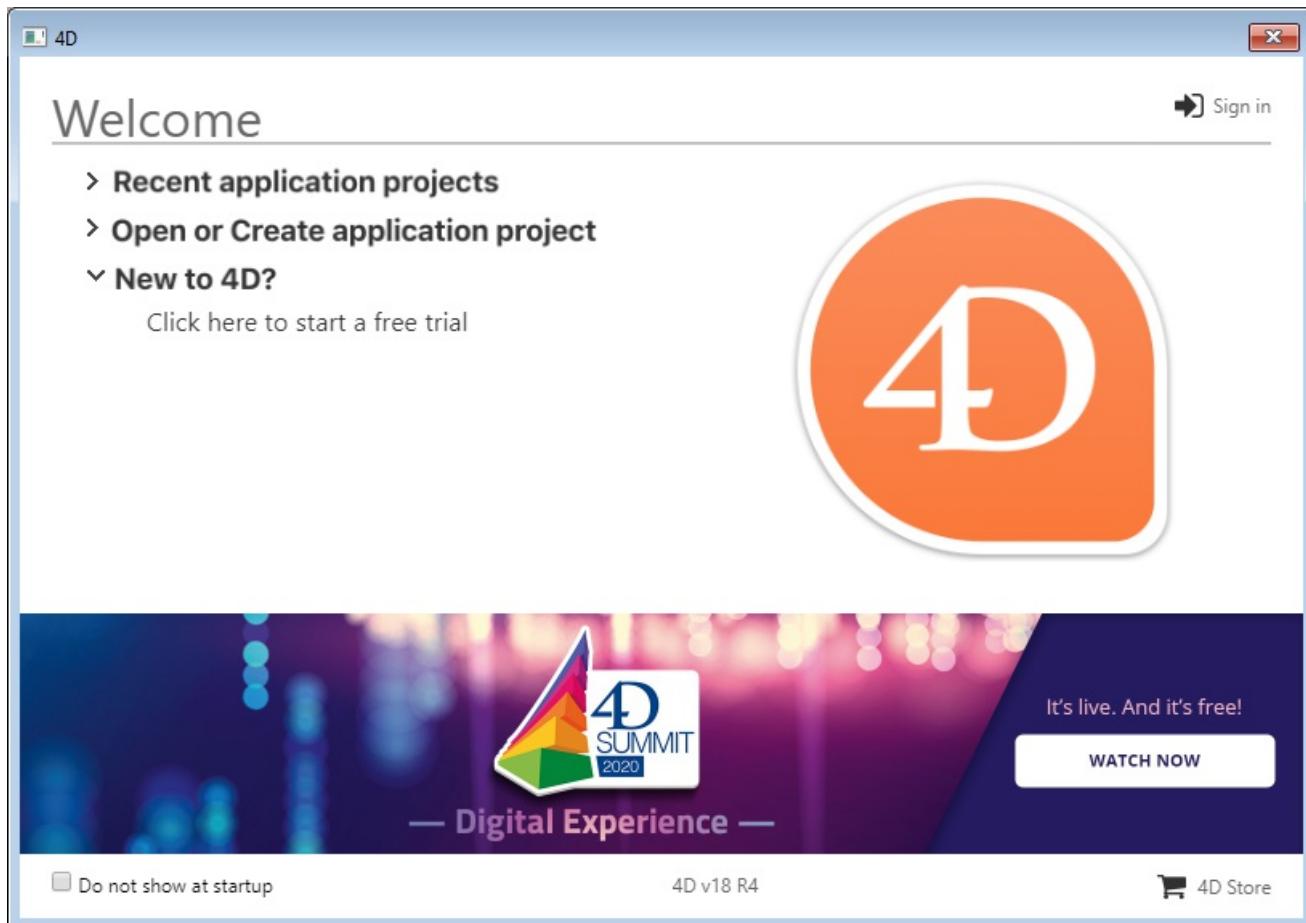
1. Connectez-vous au site internet de 4D puis consultez la page [Téléchargements](#).
2. Cliquez sur le lien de téléchargement du produit 4D de votre choix, puis suivez les instructions qui s'affichent à l'écran.

Connexion

Une fois l'installation terminée, vous pouvez lancer 4D et vous connecter. Pour ce faire, double-cliquez sur l'icône du produit 4D.



Apparaît ensuite l'assistant de bienvenue :



- Si vous souhaitez découvrir et explorer 4D, cliquez sur le lien [essai gratuit](#). Il vous sera demandé de vous connecter ou de créer un compte 4D.
- Si vous possédez déjà un compte 4D, cliquez sur le lien [Se connecter](#) sur le côté supérieur droit de l'assistant d'installation et saisissez les informations relatives à votre compte. Toute licence 4D préalablement enregistrée est automatiquement mise à jour (ou les packs d'extension supplémentaires sont chargés) sur votre machine.

Élargissez la zone Ouvrir ou créer un projet d'application et sélectionnez l'action que vous souhaitez exécuter :

- Connectez-vous au 4D Server - utilisez 4D en tant que client et connectez-vous à une application déjà chargée par 4D Server.
- Ouvrez un projet d'application local - chargez un projet d'application existant stocké sur votre disque.
- Créez un nouveau projet d'application - créez un nouveau projet d'application vide sur votre disque.

Nous vous souhaitons une excellente expérience 4D !

Architecture of a project

Un projet 4D est constitué de plusieurs fichiers et dossiers, stockés dans un seul dossier parent de l'application (dossier package). Par exemple :

- MonProjet
 - Composants
 - Données
 - Logs
 - Settings
 - Documentation
 - Plugins
 - Project
 - DerivedData
 - Sources
 - Trash
 - Resources
 - Settings
 - userPreferences.jSmith
 - WebFolder

Si votre projet a été converti depuis une base binaire, des dossiers supplémentaires peuvent être présents. Voir "Conversion de bases en projets" sur [doc.4d.com](#).

Dossier Project

La hiérarchie du dossier Project se présente généralement comme suit :

- fichier `<applicationName>.4DProject`
- Sources
 - Classes
 - DatabaseMethods
 - Méthodes
 - Formulaires
 - TableForms
 - Triggers
- DerivedData
- Trash (le cas échéant)

fichier `<applicationName>.4DProject`

Le fichier de développement de projet, utilisé pour désigner et lancer le projet. Ce fichier peut être ouvert par :

- 4D
- 4D Server (read-only, see [Opening a remote project](#))

Dans les projets 4D, le développement est réalisé avec 4D et le développement multi-utilisateurs est géré par des outils de contrôle de version. 4D Server peut ouvrir des fichiers .4DProject à des fins de test.

Ce fichier texte peut également contenir des clés de configuration, notamment "`tokenizedText" : false`".

Sources

Contenu	Description	Format
catalog.4DCatalog	Définit des tables et des champs	XML
folders.json	Définitions des dossiers de l'Explorateur	JSON
menus.json	Définit les menus	JSON
settings.4DSettings	<p>Propriétés de la base <i>Structure</i>. Elles ne sont pas prises en compte si les <i>paramètres utilisateur</i> ou les <i>paramètres utilisateur des données</i> sont définis.</p> <p>Attention : dans les applications compilées, les paramètres de structure sont stockés dans le fichier .4dz (lecture seule). Pour le déploiement, il est nécessaire d'utiliser les <i>paramètres utilisateur</i> ou les <i>paramètres utilisateur pour les données</i> afin de définir des paramètres personnalisés.</p>	XML
tips.json	Définit les messages d'aide	JSON
lists.json	Listes définies	JSON
filters.json	Filtres définis	JSON
styleSheets.css	Feuilles de style CSS	CSS
styleSheets_mac.css	Feuilles de style css sur Mac (à partir d'une base binaire convertie)	CSS
styleSheets_windows.css	Feuilles de style css sur Windows (à partir d'une base binaire convertie)	CSS

DatabaseMethods

Contenu	Description	Format
databaseMethodName.4dm	Méthodes base définies dans le projet. Un fichier par méthode base	Texte

Méthodes

Contenu	Description	Format
methodName.4dm	Méthodes projet définies dans le projet. Un fichier par méthode	Texte

Classes

Contenu	Description	Format
className.4dm	Méthode de définition de classe utilisateur, permettant d'instancier des objets spécifiques. Un fichier par classe, le nom du fichier est le nom de la classe	Texte

Formulaires

Contenu	Description	Format
formName/form.4DForm	Description du formulaire projet	json
formName/method.4dm	Méthode formulaire projet	Texte
formName/Images/pictureName	Image statique du formulaire projet	picture
formName/ObjectMethods/objectName.4dm	Méthodes objet. Un fichier par méthode objet	Texte

TableForms

Contenu	Description	Format
<i>n/Input/formName/form.4DForm</i>	Description du formulaire d'entrée de la table (n étant le numéro de table)	json
<i>n/Input/formName/Images/pictureName</i>	Images statiques du formulaire d'entrée de la table	picture
<i>n/Input/formName/method.4dm</i>	Méthode du formulaire d'entrée de la table	Texte
<i>n/Input/formName/ObjectMethods/objectName.4dm</i>	Méthodes objet du formulaire d'entrée. Un fichier par méthode objet	Texte
<i>n/Output/formName/form.4DForm</i>	Description du formulaire de sortie de la table (n étant le numéro de table)	json
<i>n/Output/formName/Images/pictureName</i>	Images statiques du formulaire de sortie de la table	picture
<i>n/Output/formName/method.4dm</i>	Méthode du formulaire de sortie de la table	Texte
<i>n/Output/formName/ObjectMethods/objectName.4dm</i>	Méthodes objet du formulaire de sortie. Un fichier par méthode objet	Texte

Triggers

Contenu	Description	Format
<i>table_n.4dm</i>	Méthodes trigger définies dans le projet. Un fichier de trigger par table (n étant le numéro de table)	Texte

Note : L'extension de fichier .4dm est un format de fichier texte contenant le code d'une méthode 4D. Il est compatible avec les outils de contrôle de version.

Trash

Le dossier Trash contient des méthodes et des formulaires qui ont été supprimés du projet (le cas échéant). Il peut contenir les dossiers suivants :

- Méthodes
- Formulaires
- TableForms

Dans ces dossiers, les noms des éléments supprimés sont entre parenthèses, par exemple. "(myMethod).4dm". L'organisation des dossiers est identique à celle du dossier [Sources](#).

DerivedData

Le dossier DerivedData contient des données en cache utilisées en interne par 4D pour optimiser le traitement. Il est automatiquement créé ou recréé si nécessaire. Vous pouvez ignorer ce dossier.

Libraries

Ce dossier n'est utilisé que sur macOS.

Le dossier Librairies contient le fichier résultant d'une compilation avec le [compilateur Silicon](#) sur macOS.

Resources

Le dossier Resources contient tous les fichiers et dossiers de ressources personnalisés du projet. Dans ce dossier, vous pouvez placer tous les fichiers nécessaires à la traduction ou à la personnalisation de l'interface de l'application (fichiers

image, fichiers texte, fichiers XLIFF, etc.). 4D utilise des mécanismes automatiques pour manipuler le contenu de ce dossier, notamment pour le traitement des fichiers XLIFF et des images statiques. Pour l'utilisation en mode distant, le dossier Resources vous permet de partager des fichiers entre le serveur et tous les ordinateurs clients. Voir le *Manuel 4D Server - Référence*.

Contenu	Description	Format
<i>item</i>	Fichiers et dossiers de ressources de la base	variés
Images/Library/ <i>item</i>	Images de la bibliothèque d'images sous forme de fichiers séparés(*). Les noms de ces éléments deviennent des noms de fichiers. Si un élément dupliqué existe, un numéro est ajouté au nom.	picture

(*) uniquement si le projet a été exporté depuis une base binaire .4db.

Données

Le dossier Data contient le fichier de données ainsi que tous les fichiers et dossiers relatifs aux données.

Contenu	Description	Format
data.4dd(*)	Fichier de données contenant les données saisies dans les enregistrements et toutes les données appartenant aux enregistrements. Lorsque vous ouvrez un projet 4D, l'application ouvre par défaut le fichier de données courant. Si vous modifiez le nom ou l'emplacement de ce fichier, la boîte de dialogue <i>Ouvrir un fichier de données</i> apparaît alors pour vous permettre de sélectionner le fichier de données à utiliser ou d'en créer un nouveau	binaire
data.journal	Créé uniquement lorsque la base de données utilise un fichier journal. Le fichier journal est utilisé pour assurer la sécurité des données entre les sauvegardes. Toutes les opérations effectuées sur les données sont enregistrées séquentiellement dans ce fichier. Par conséquent, chaque opération sur les données entraîne deux actions simultanées : la première sur les données (l'instruction est exécutée normalement) et la seconde dans le fichier journal (une description de l'opération est enregistrée). Le fichier journal est construit indépendamment, sans perturber ni ralentir le travail de l'utilisateur. Une base de données ne peut fonctionner qu'avec un seul fichier journal à la fois. Le fichier journal enregistre des opérations telles que des ajouts, des modifications ou des suppressions d'enregistrements, des transactions, etc. Il est généré par défaut lors de la création d'une base de données.	binaire
data.match	(interne) UUID correspondant au numéro de la table	XML

(*) Lorsque le projet est créé depuis une base binaire .4b, le fichier de données demeure inchangé. Ainsi, il peut être nommé différemment et placé dans un autre emplacement.

Settings

Ce dossier contient des fichiers de propriétés utilisateur des données utilisés pour l'administration de l'application.

Ces paramètres ont la priorité sur les [fichiers de propriétés utilisateur](#) et les fichiers de [propriétés structure](#).

Contenu	Description	Format
directory.json	Description des groupes et utilisateurs 4D et de leurs droits d'accès lorsque l'application est lancée avec ce fichier de données.	JSON
Backup.4DSettings	Paramètres de sauvegarde de la base de données, utilisés pour définir les options de sauvegarde lorsque la base est lancée avec ce fichier de données. Les clés concernant la configuration de la sauvegarde sont décrites dans le manuel <i>Sauvegarde des clés XML 4D</i> .	XML
settings.4DSettings	Propriétés de la base personnalisées pour ce fichier de données.	XML

Logs

Le dossier Logs contient tous les fichiers journaux utilisés par le projet. Les fichiers journaux comprennent notamment :

- conversion de base de données,
- requêtes de serveur Web,
- journal des activités de sauvegarde/restitution (*Journal de sauvegarde[xxx].txt*, voir [Journal de sauvegarde](#))
- débogage de commandes,
- Requêtes 4D Server (générées sur les postes clients et sur le serveur).

Un dossier Logs supplémentaire est disponible dans le dossier des préférences utilisateur du système (dossier 4D actif, voir la commande [Get 4D folder](#)) pour les fichiers journaux de maintenance et dans les cas où le dossier de données est en lecture seule.

Settings

Ce dossier contient des fichiers de propriétés utilisateur utilisés pour l'administration de l'application.

Ces paramètres ont la priorité sur les fichiers de [propriétés structure](#). Toutefois, si un [fichier de paramètres utilisateur](#) existe, il est prioritaire par rapport au fichier des propriétés utilisateur.

Contenu	Description	Format
directory.json	Description des groupes et utilisateurs 4D pour l'application, ainsi que leurs droits d'accès	JSON
Backup.4DSettings	Paramètres de sauvegarde de la base de données, utilisés pour définir les options de sauvegarde) à chaque lancement de sauvegarde. Ce fichier peut également être utilisé pour lire ou définir des options supplémentaires, telles que la quantité d'informations stockées dans le <i>journal de sauvegarde</i> . Les clés concernant la configuration de la sauvegarde sont décrites dans le manuel <i>Sauvegarde des clés XML 4D</i> .	XML
BuildApp.4DSettings	Fichier de paramètres de génération, créé automatiquement lors de l'utilisation de la boîte de dialogue du générateur d'applications ou de la commande <code>BUILD APPLICATION</code>	XML
settings.4DSettings	Paramètres personnalisés pour ce projet (tous les fichiers de données)	XML

userPreferences.<userName>

This folder contains files that memorize user configurations, e.g. break point or window positions. Vous pouvez simplement ignorer ce dossier. Il contient par exemple :

Contenu	Description	Format
methodPreferences.json	Préférences de l'éditeur de méthodes de l'utilisateur courant	JSON
methodWindowPositions.json	Position de la fenêtre de l'utilisateur courant pour les méthodes	JSON
formWindowPositions.json	Position de la fenêtre de l'utilisateur courant pour les formulaires	JSON
workspace.json	Liste de fenêtres ouvertes : sous macOS, ordre des fenêtres à onglets	JSON
debuggerCatches.json	Appels vers commandes	JSON
recentTables.json	Liste ordonnée de tables	JSON
preferences.4DPreferences	Current data path and main window positions	XML
CompilerIntermediateFiles	Fichiers intermédiaires résultant de la compilation Apple Silicon	Folder

Composants

This folder contains the components to be available in the application project. Il doit être stocké au même niveau que le dossier Project.

Une application projet peut être elle-même un composant : - à des fins de développement : insérer un alias du fichier .4dproject dans le dossier Components du projet hôte. - for deployment: [build the component](#) and put the resulting .4dz file in a .4dbase folder in the Components folder of the host application.

Plugins

This folder contains the plug-ins to be available in the application project. Il doit être stocké au même niveau que le dossier Project.

Documentation

Ce dossier contient tous les fichiers de documentation (.md) créés pour les éléments du projet, tels que les classes, les méthodes ou les formulaires. Les fichiers de documentation sont gérés et affichés dans l'Explorateur 4D.

Pour plus d'informations, reportez-vous à [Documenter un projet](#).

WebFolder

Il s'agit du dossier racine par défaut du serveur Web 4D pour les pages, les images, etc. Il est automatiquement créé lors du premier lancement du serveur Web.

Fichier `.gitignore` (optionnel)

Fichier qui spécifie les fichiers qui seront ignorés par git. Vous pouvez inclure un fichier gitignore dans vos projets en utilisant l'option Créer un fichier `.gitignore` sur la page Général des préférences. Pour configurer le contenu de ce fichier, voir [Créer un fichier `.gitignore`](#).

Documenter un projet

In application projects, you can document your methods as well as your forms, tables, or fields. Creating documentation is particularly appropriate for projects being developed by multiple programmers and is generally good programming practice. Documentation can contain a description of an element as well as any information necessary to understand how the element functions in the application.

Les éléments de projet suivants acceptent la documentation :

- Méthodes (méthodes base, méthodes composant, méthodes projet, méthodes formulaire, méthodes 4D Mobile, triggers et classes)
- Formulaires
- Tables et champs

Vos fichiers de documentation sont écrits dans la syntaxe Markdown (fichiers .md) à l'aide de n'importe quel éditeur prenant en charge le Markdown. Ils sont stockés en tant que fichiers indépendants dans votre dossier Project.

La documentation s'affiche dans la zone d'aperçu (panneau de droite) de l'Explorateur :

The screenshot shows the 4D Explorer window with the title "HDI_4DW_P_GetSetFormulas - Explorer". The left sidebar has icons for Home, Tables, Forms, Methods (which is selected), Commands, Constants, Plug-ins, and Trash. The main pane shows a tree view under "Methods" with "Project Methods" expanded, showing items like 00_Start, Compiler_Arrays, Compiler_Methods, Compiler_Variables, Compiler_Variables_Inter, GetEmptyString, GetFullAddress, GetLogo, GetPage, GetString, initHDI, keep, m_Quit, Method15, and zz. Below this is a list of other method categories: Component Methods, Database Methods, Classes, Triggers, Project Form Methods, Table Form Methods, and 4D Mobile Methods. On the right, there is a table with four columns: Parameter, Type, in/out, and Description. The table contains two rows: one for 'size' (Longint, in, Logo style selector (1 to 5)) and one for 'logo' (Picture, out, Selected logo). Below the table is a large text area titled "Description" which reads: "This method returns a logo of a specific size, depending on the value of the size parameter value. 1 = smallest size, 5 = largest size." At the bottom of this area is a code example in 4D language:

```
C_PICTURE($logo)
C_LONGINT($size)

//Get the Largest Logo
$logo:=GetLogo(5)
```

At the bottom of the window are buttons for "+", "-", "*", and a search bar, followed by "Preview" and "Documentation" tabs, with "Documentation" being the active tab.

Il peut également être partiellement exposé en tant que [conseils de l'éditeur de code](#).

Fichiers documentation

Nom du fichier de documentation

Les fichiers de documentation ont le même nom que l'élément auquel ils sont rattachés, avec l'extension ".md". Par exemple, le fichier de documentation rattaché à la méthode projet `myMethod.4dm` sera nommé `myMethod.md`.

Dans l'Explorateur, 4D affiche automatiquement le fichier de documentation avec le même nom que l'élément sélectionné (voir ci-dessous).

Architecture des fichiers de documentation

Tous les fichiers de documentation sont stockés dans le dossier `Documentation`, situé au premier niveau du dossier `Package`.

L'architecture du dossier `Documentation` est la suivante :

- `Documentation`
 - `Classes`
 - `myClass.md`
 - `DatabaseMethods`
 - `onStartup.md`
 - ...
 - `Formulaires`
 - `loginDial.md`
 - ...
 - `Méthodes`
 - `myMethod.md`
 - ...
 - `TableForms`
 - `1`
 - `input.md`
 - ...
 - ...
 - `Triggers`
 - `table1.md`
 - ...
- Un formulaire projet et sa méthode de formulaire projet partagent le même fichier de documentation pour le formulaire et la méthode.
- Un formulaire table et sa méthode de formulaire table partagent le même fichier de documentation pour le formulaire et la méthode.

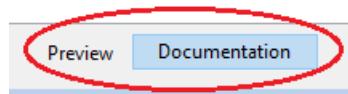
Renommer ou supprimer un élément documenté dans votre projet renomme ou supprime également le fichier Markdown associé à l'élément.

Documentation dans l'Explorateur

Visualiser la documentation

Pour afficher la documentation dans la fenêtre de l'Explorateur :

1. Assurez-vous que la zone d'aperçu est affichée.
2. Sélectionnez l'élément documenté dans la liste de l'Explorateur.
3. Cliquez sur le bouton `Documentation` situé sous la zone d'aperçu.



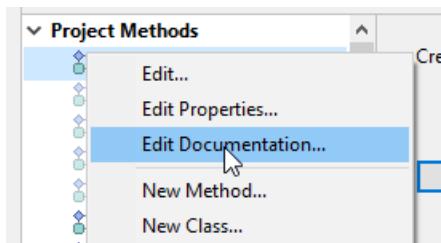
- Si aucun fichier de documentation n'a été trouvé pour l'élément sélectionné, un bouton `Créer` s'affiche (voir ci-dessous).
- Sinon, s'il existe un fichier de documentation pour l'élément sélectionné, le contenu est affiché dans la zone. Le contenu n'est pas directement modifiable dans le volet.

Modifier le fichier documentation

Vous pouvez créer et/ou modifier un fichier de documentation Markdown à partir de la fenêtre de l'Explorateur pour l'élément sélectionné.

S'il n'y a pas de fichier de documentation pour l'élément sélectionné, vous pouvez :

- cliquez sur le bouton `Créer` dans le volet `Documentation` ou,
- choisissez l'option `Modifier la documentation...` dans le menu contextuel ou le menu d'options de l'Explorateur.

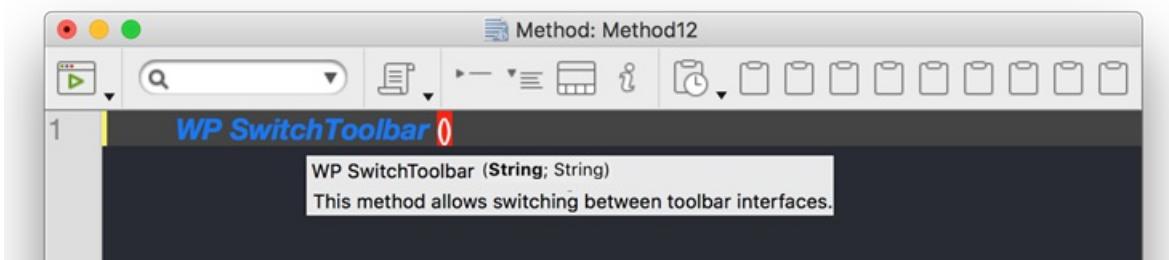


4D crée automatiquement un fichier `.md` nommé correctement avec un modèle de base à l'emplacement approprié et l'ouvre avec votre éditeur Markdown par défaut.

Si un fichier de documentation existe déjà pour l'élément sélectionné, vous pouvez l'ouvrir avec votre éditeur Markdown en choisissant l'option `Modifier la documentation...` dans le menu contextuel ou le menu d'options de l'Explorateur.

Visualiser la documentation dans l'éditeur de code

L'éditeur de code 4D affiche une partie de la documentation d'une méthode dans son info-bulle.



Si un fichier nommé "`<MethodName>.md`" existe dans le dossier "`<package>/documentation`", l'éditeur de code affiche (par priorité) :

- Tout texte saisi dans une balise de `commentaire HTML` (`<!-- commande documentation -->`) en haut du fichier markdown.
- Ou, si aucune balise de `commentaire html` n'est utilisée, la première phrase après une balise `# Description` du fichier markdown.

Dans ce cas, la première ligne contient le prototype de la méthode, généré automatiquement par le parseur du code 4D.

Sinon, l'éditeur de code affiche [le bloc de commentaire en haut du code de la méthode](#).

Définition du fichier de documentation

4D utilise un modèle de base pour créer de nouveaux fichiers de documentation. Ce modèle propose des fonctionnalités spécifiques qui vous permettent [d'afficher des informations dans l'éditeur de code](#).

Cependant, vous pouvez utiliser toutes les [balises Markdown prises en charge](#).

De nouveaux fichiers de documentation sont créés avec les contenus par défaut suivants :

```

Method15.md
1 <!-- Type here your summary -->
2 ## Description
3 ~
4 ## Example
5 ~
6 ````4d
7 Type here your example
8 ````
```

Ligne	Description
"<!-- Type your summary here -->"	Commentaire HTML. Utilisé en priorité comme description de méthode dans les astuces de l'éditeur de code
## Description	Titre de niveau 2 en Markdown. La première phrase qui suit cette balise est utilisée comme description d'une méthode dans les astuces de l'éditeur de code si le commentaire HTML n'est pas utilisé
## Example	Titre de niveau 2, vous pouvez utiliser cette zone pour afficher un exemple de code
` 4D Insérez votre exemple ici \`	Utilisé pour formater des exemples de code 4D (utilise la bibliothèque highlight.js)

Prise en charge du markdown

- La balise de titre est prise en charge :

```
# Title 1
## Title 2
### Title 3
```

- Les balises de style (italique, gras, barré) sont prises en charge :

```
_italic_
**bold**
**_bold/italic_**
~~strikethrough~~
```

- La balise du bloc de code (` ``4d ... `` `) est supportée avec le surlignage du code 4D :

```
` 4d C_TEXT($txt) $txt:="Hello world!" \`
```

- La balise de tableau est prise en charge :

Parameter	Type	Description
-----	-----	-----
wpArea	String	Write pro area
toolbar	String	Toolbar name

- La balise de lien est prise en charge :

```
// Case 1
The [documentation](https://doc.4d.com) of the command ....

// Case 2
[4D blog] [1]

[1]: https://blog.4d.com
```

- Les balises d'image sont prises en charge :

```
![image info](pictures/image.png)

![logo 4D](https://blog.4d.com/wp-content/uploads/2016/09/logoOriginal-1.png "4D blog logo")

![logo 4D blog with link](https://blog.4d.com/wp-content/uploads/2016/09/logoOriginal-1.png "4D blog log")
```



Pour plus d'informations, consultez le [guide Markdown de GitHub](#).

Exemple

Dans le fichier `WP_SwitchToolbar.md`, vous pouvez entrer le code suivant :

Parameter	Type	in/out	Description
size	Longint	in	Logo style selector (1 to 5)
logo	Picture	out	Selected logo

```
## Description

Cette méthode retourne un logo de taille spécifique, selon la valeur du paramètre *size*.  

1 = plus petite taille, 5 = plus grande taille.

## Exemple

C_PICTURE($logo)
C_LONGINT($size)

//Obtenir le plus grand logo
$logo:=GetLogo(5)
```

- Vue de l'Explorateur :

HDL_4DWP_GetSetFormulas - Explorer

Methods

Project Methods

- 00_Start
- Compiler_Arrays
- Compiler_Methods
- Compiler_Variables
- Compiler_Variables_Inter
- GetEmptyString
- GetFullAddress
- GetLogo**
- GetPage
- GetString
- initHDI
- keep
- m_Quit
- Method15
- zz

Parameter	Type	in/out	Description
size	Longint	in	Logo style selector (1 to 5)
logo	Picture	out	Selected logo

Description

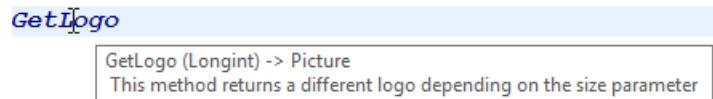
This method returns a logo of a specific size, depending on the value of the `size` parameter value. 1 = smallest size, 5 = largest size.

Example

```
C_PICTURE($logo)
C_LONGINT($size)

//Get the largest logo
$logo:=GetLogo(5)
```

- Vue de l'Éditeur de code :



À propos du Langage 4D

Grâce à son langage 4D intégré, qui comprend plus de 1300 commandes, 4D est un outil de développement puissant pour les applications web, mobile ou desktop. Ce langage peut être utilisé pour effectuer plusieurs types de tâches allant de la réalisation de simples calculs à la création d'interfaces utilisateur personnalisées et complexes. Vous pouvez par exemple :

- Accéder par programmation à tous les éditeurs de gestion des enregistrements (tris, recherches, etc.),
- Créer et imprimer des états et des étiquettes complexes avec les données de la base,
- Communiquer avec d'autres systèmes d'information,
- Envoyer des e-mails,
- Gérer des documents et des pages web,
- Importer et exporter des données entre des applications 4D et d'autres applications,
- Incorporer des procédures écrites dans d'autres langages que celui de 4D.

La flexibilité et la puissance du langage de 4D en font l'outil idéal pour exécuter toute une gamme de fonctions de gestion de l'information. Les utilisateurs novices peuvent exécuter des calculs rapidement, les utilisateurs plus expérimentés peuvent personnaliser leurs applications sans connaissances préalables en programmation et les développeurs chevronnés peuvent utiliser ce langage puissant pour ajouter à leurs applications des fonctions sophistiquées, allant jusqu'au transfert de fichiers, aux communications et au suivi. Quant aux développeurs ayant une maîtrise de la programmation dans d'autres langages, ils peuvent ajouter leurs propres commandes au langage de 4D.

Qu'est-ce qu'un langage ?

Le langage de 4D n'est pas très différent de celui que nous utilisons tous les jours. C'est une forme de communication qui permet d'exprimer des idées, d'informer ou de donner des instructions. Tout comme un langage parlé, celui de 4D se compose d'un vocabulaire, d'une grammaire et d'une syntaxe, que vous employez pour indiquer au programme comment gérer votre application et vos données.

Il n'est pas nécessaire de connaître entièrement le langage. Pour pouvoir vous exprimer en anglais, vous n'êtes pas obligé de connaître la totalité de la langue anglaise ; en réalité, un peu de vocabulaire suffit. Il en va de même pour 4D — il vous suffit de connaître un minimum du langage pour être productif, et vous en apprendrez de plus en plus au fur et à mesure de vos besoins.

Pourquoi utiliser un langage ?

À première vue, un langage de programmation dans 4D peut sembler inutile. En effet, 4D propose des outils puissants et entièrement paramétrables en mode Développement, permettant d'exécuter une grande variété de tâches de gestion des données sans programmation. Les opérations fondamentales telles que la saisie de données, les requêtes, les tris et la création d'états s'effectuent facilement. Il existe aussi de nombreuses autres possibilités, telles que les filtres de validation des données, les aides à la saisie, la représentation graphique et la génération d'étiquettes.

Alors, pourquoi avons-nous besoin d'un langage 4D ? Voici quelques-uns de ses principaux rôles :

- Automatiser les tâches répétitives : par exemple la modification de données, la génération d'états complexes et la réalisation de longues séries d'opérations ne nécessitant pas l'intervention d'un utilisateur.
- Contrôler l'interface utilisateur : vous pouvez gérer les fenêtres et les menus, contrôler les formulaires et les objets d'interface.
- Gérer les données de manière très fine : cette gestion inclut le traitement de transactions, les systèmes de validation complexes, la gestion multi-utilisateurs, l'utilisation des ensembles et des sélections temporaires.
- Contrôler l'ordinateur : vous pouvez contrôler les communications par le port série, la gestion des documents et des erreurs.
- Créer des applications : vous pouvez créer des applications simples, personnalisées et autonomes.
- Ajouter des fonctionnalités au serveur Web 4D intégré : par exemple, vous pouvez créer et mettre à jour des pages web dynamiques contenant vos données.

Le langage vous permet de contrôler totalement la structure et le fonctionnement de votre application. 4D propose de

puissants éditeurs "génériques", mais le langage vous offre la possibilité de personnaliser votre application autant que vous voulez.

Prendre le contrôle de vos données

Le langage de 4D vous permet de prendre le contrôle total de vos données, d'une manière à la fois puissante et élégante. Il reste d'un abord facile pour un débutant, et est suffisamment riche pour un développeur d'applications. Il permet de passer par étapes d'une simple exploitation des fonctions intégrées de 4D à une application entièrement personnalisée.

Les commandes du langage de 4D vous donnent la possibilité d'accéder aux éditeurs standard de gestion des enregistrements. Par exemple, lorsque vous utilisez la commande `QUERY`, vous accédez à l'Editeur de quêtes (accessible en mode Développement via la commande Chercher... dans le menu Enregistrements). Vous pouvez, si vous le voulez, indiquer explicitement à la commande `Query` ce qu'elle doit rechercher. Par exemple, `QUERY([People]; [People]Last Name="Dupont")` trouvera toutes les personnes nommées Dupont dans votre base.

Le langage 4D est très puissant — une commande remplace souvent des centaines, voire des milliers de lignes de code écrites dans les langages informatiques traditionnels. Et cette puissance s'accompagne d'une réelle simplicité : les commandes sont écrites en français. Par exemple, vous utilisez la commande `QUERY` pour effectuer une recherche ; pour ajouter un nouvel enregistrement, vous appelez la commande `ADD RECORD`.

Le langage est conçu de telle manière que vous puissiez facilement accomplir n'importe quelle tâche. L'ajout d'un enregistrement, le tri des enregistrements, la recherche de valeurs et les autres opérations de même type sont définies par des commandes simples et directes. Mais les routines peuvent également contrôler les ports série, lire des documents sur le disque, traiter des systèmes de transactions complexes, et plus encore.

Même les opérations les plus compliquées se définissent de manière relativement simple. Il serait inimaginable de les réaliser sans le langage de 4D. Cependant, même à l'aide de la puissance des commandes du langage, certaines tâches peuvent se révéler complexes et difficiles. Ce n'est pas l'outil lui-même qui fait le travail ; une tâche peut représenter en soi une difficulté, l'outil ne fait que faciliter son traitement. Par exemple, un logiciel de traitement de texte permet d'écrire un livre plus rapidement et plus facilement, mais il ne l'écrira pas à votre place. Le langage de 4D vous permet de gérer plus facilement vos données et d'appréhender les tâches ardues en toute confiance.

Est-ce un langage informatique "traditionnel" ?

Si vous êtes familier avec les langages informatiques traditionnels, ce paragraphe peut vous intéresser. Si ce n'est pas le cas, vous pouvez passer directement au paragraphe suivant.

Le langage 4D n'est pas un langage informatique traditionnel. C'est un des langages les plus souples et les plus innovants que l'on puisse trouver aujourd'hui sur micro-ordinateur. Le langage a été conçu pour s'adapter à vous, et non l'inverse.

Pour utiliser les langages traditionnels, vous devez avant tout réaliser des maquettes détaillées. C'est même l'une des principales phases d'un développement. 4D vous permet de commencer à utiliser le langage à tout moment et dans n'importe quelle partie de votre projet. Vous pouvez commencer par ajouter une méthode objet dans un formulaire, puis plus tard une ou deux méthodes formulaires. Comme votre projet devient plus sophistiquée, vous pouvez ajouter une méthode projet contrôlée par un menu. Vous êtes totalement libre d'utiliser une petite partie du langage ou une partie plus étendue. Ce n'est pas "tout ou rien", comme c'est le cas dans beaucoup d'autres bases de données.

Les langages traditionnels vous obligent à définir et à pré-déclarer vos objets d'interface sous une forme syntaxique rigide. Dans 4D, créez simplement un objet, tel qu'un bouton, et utilisez-le. 4D se charge de la gestion de l'objet. Par exemple, pour utiliser un bouton, il suffit de le dessiner dans un formulaire et de lui donner un nom. Lorsque l'utilisateur clique sur le bouton, le langage déclenche automatiquement vos méthodes.

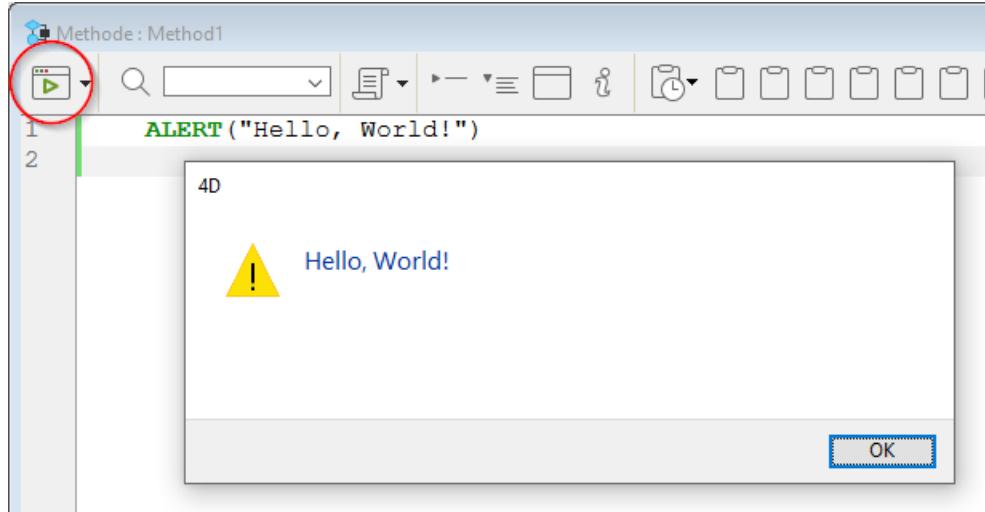
Les langages traditionnels sont souvent rigides et inflexibles, et exigent que les commandes soient saisies dans un style très formel et contraignant. Le langage de 4D rompt avec la tradition, au grand bénéfice de l'utilisateur.

Tour d'horizon

En utilisant le langage 4D, le traditionnel "Hello, world!" peut s'afficher à l'écran de plusieurs manières. Le plus simple est probablement d'écrire la ligne suivante dans une méthode de projet :

```
ALERT("Hello, World!")
```

Ce code affichera une boîte de dialogue d'alerte standard contenant le message "Hello, World!" et un bouton OK. Pour exécuter le code, il vous suffit de cliquer sur le bouton d'exécution dans l'éditeur de méthode :



Vous pouvez également associer ce code à un bouton de formulaire et exécuter le formulaire. Dans ce cas, en cliquant sur le bouton, vous afficherez la boîte de dialogue d'alerte. Dans tous les cas, vous venez d'exécuter votre première ligne de code 4D !

Assigner des valeurs

Vous pouvez donner des valeurs aux variables, aux champs, aux éléments de tableaux et/ou récupérer leur valeur. Donner une valeur à une variable s'appelle assigner une valeur (ou affecter une valeur) et s'effectue à l'aide de l'opérateur d'assignation (:=). L'opérateur d'assignation est également utilisé pour assigner des valeurs aux champs ou aux éléments de tableaux.

```
$MyNumber:=3 //assigne 3 à la variable MyNumber  
[Products]Size:=$MyNumber //assigne la variable MyNumber au champ [Products]Size  
arrDays{2}:="Mardi" //assigne la chaîne "Mardi" au 2ème élément arrDays  
MyVar:=Length("Acme") //assigne le résultat de la fonction (4) à MyVar  
$myDate:=!2018/01/21! //assigne une date littérale  
$myHour:=?08:12:55? //assigne une heure littérale
```

Vous devez impérativement distinguer l'opérateur d'affectation := des autres opérateurs. Plutôt que de combiner des expressions dans une nouvelle expression, l'opérateur d'affectation copie la valeur de l'expression à droite de l'opérateur d'affectation dans la variable ou le champ situé à gauche de l'opérateur.

Important : Ne confondez pas l'opérateur d'assignation (:=) avec le signe égal (=). Un opérateur d'affectation différent (et non pas =) a été choisi délibérément pour éviter les problèmes et la confusion qui surviennent souvent avec == ou === dans d'autres langages de programmation. De telles erreurs sont souvent difficiles à reconnaître pour le compilateur et conduisent à un dépannage fastidieux.

Variables

Le langage 4D est fortement typé, bien qu'une certaine flexibilité soit autorisée dans de nombreux cas. Par exemple, pour créer une variable du type date, vous pouvez écrire : Par exemple, pour créer une variable du type date, vous pouvez écrire :

```
var MyDate : Date
```

Le mot-clé `var` permet de déclarer des variables objet d'un type de classe défini, par exemple :

```
var myPerson : cs.Person  
//variable de la classe utilisateur Person
```

Même si cela est généralement déconseillé, vous pouvez déclarer des variables simplement en les utilisant; il n'est pas obligatoire de les déclarer formellement. Par exemple, si vous voulez créer une variable qui contient la date du jour plus 30 jours, il vous suffit d'écrire dans 4D :

```
MyOtherDate:=Current date+30
```

La ligne de code indique "MyOtherDate obtient la date actuelle plus 30 jours." Cette ligne crée la variable, lui attribue à la fois le type de date (temporaire) et un contenu. Une variable déclarée par affectation est interprétée comme étant non typée, c'est-à-dire qu'elle peut être affectée à d'autres types dans d'autres lignes, puis modifie le type de manière dynamique. Une variable typée avec `var` ne peut pas changer le type. En revanche, en [mode compilé](#), le type ne peut jamais être modifié, quelle que soit la façon dont la variable a été déclarée.

Commandes

Les commandes 4D sont des méthodes intégrées qui permettent d'effectuer une action. Toutes les commandes 4D, telles que `CREATE RECORD` ou `ALERT`, sont décrites dans le *Manuel Langage de 4D*, et sont regroupées par thème. Les commandes sont souvent utilisées avec des paramètres qui sont passés entre parenthèses () et séparés par des points-virgules (;). Exemple :

```
COPY DOCUMENT("dossier1\\nom1";"dossier2\\" ; "nouveau")
```

Certaines commandes sont reliées à des collections ou à des objets, auquel cas ce sont des méthodes nommées qui sont utilisées à l'aide de la notation en point. Par exemple :

```
$c:=New collection(1;2;3;4;5)  
$nc:=$c.slice(0;3) // $nc=[1,2,3]  
  
$lastEmployee:=$employee.last()
```

Vous pouvez utiliser des plug-ins ou des composants 4D qui ajoutent de nouvelles commandes à votre environnement de développement 4D.

Il existe de nombreux plug-ins proposés par la communauté des utilisateurs de 4D ou des développeurs tiers. Par exemple, en utilisant les pages [4d-plugin-pdf-pages](#) sur macOS :

```
PDF REMOVE PAGE(path;page)
```

4D SVG est un exemple de composant utilitaire qui augmente les capacités de votre application :

```
//faire un dessin  
svgRef:=SVG_New  
objectRef:=SVG_New_arc(svgRef;100;100;90;90;180)
```

4D SVG est inclus dans 4D.

Constantes

4D propose un large ensemble de constantes prédéfinies, dont les valeurs sont accessibles par un nom. Elles permettent d'écrire un code plus lisible. Par exemple, `XML DATA` est une constante (valeur 6).

```
vRef:=Open document("PassFile";"TEXTE";Read Mode) // ouvrir le doc en mode lecture seule
```

Par défaut, les constantes prédéfinies sont soulignées dans l'éditeur de méthodes 4D.

Méthodes

4D propose un grand nombre de méthodes (ou de commandes) intégrées, mais vous permet également de créer vos propres méthodes de projet. Les méthodes de projet sont des méthodes définies par l'utilisateur qui contiennent des commandes, des opérateurs et d'autres parties du langage. Les méthodes projet sont des méthodes génériques, mais il existe d'autres types de méthodes : les méthodes objet, les méthodes formulaire, les méthodes table (Triggers) et les méthodes base.

Une méthode est composée de plusieurs lignes d'instructions. Une ligne d'instructions effectue une action. Cette ligne d'instruction peut être simple ou complexe.

Par exemple, la ligne de code suivante est une instruction qui affichera une boîte de dialogue de confirmation :

```
CONFIRM("Souhaitez-vous vraiment clore ce compte ?";"Oui";"Non")
```

Une méthode contient également des tests et des boucles qui gèrent le flux d'exécution. Les méthodes 4D prennent en charge les structures `If...Else...End if` et `Case of...Else...End case` ainsi que les boucles : `While...End while`, `Repeat...Until`, `For...End for`, et `For each...End for each`:

L'exemple suivant permet d'examiner chaque caractère du texte `vtSomeText` :

```
For($vlChar;1;Length(vtSomeText))  
    //Faire quelque chose avec le caractère s'il s'agit d'une tabulation  
    If(Character code(vtSomeText[[$vlChar]])=Tab)  
        //...  
    End if  
End for
```

Une méthode projet peut en appeler une autre avec ou sans les paramètres (arguments). Les paramètres sont passés à la méthode entre parenthèses, à la suite du nom de la méthode. Chaque paramètre est séparé par des points virgule (;). Les paramètres sont passés à la méthode appelée en tant que variables locales numérotées séquentiellement : \$1, \$2, ..., \$n. Une méthode peut retourner une seule valeur dans le paramètre \$0. Lorsque vous appelez une méthode, vous saisissez simplement son nom :

```

$myText:="hello"
$myText:=Do_Something($myText) //Appelle la méthode Do_Something
ALERT($myText) //"HELLO"

//Voici le code de la méthode Do_Something
$0:=Uppercase($1)

```

Types de données

De nombreux types de données peuvent être manipulés via le langage 4D. Il existe des types de données élémentaires (chaîne, numérique, date, heure, booléen, image, pointeur, tableau), ainsi que des types de données composites (BLOBs, objets, collections).

A noter que les données de type chaîne et numérique peuvent être associées à plusieurs types de champ. Lorsque des données sont placées dans un champ, le langage les convertit automatiquement dans le type du champ. Par exemple, si un champ de type entier est utilisé, les valeurs qu'il contient sont automatiquement traitées en tant que numériques. En d'autres termes, vous n'avez pas à vous préoccuper du mélange de champs de types semblables lorsque vous programmez avec 4D ; le langage le gère pour vous.

Cependant, il est important, lorsque vous utilisez le langage, de ne pas mélanger les différents types de données. Tout comme il est absurde de stocker la valeur "ABC" dans un champ de type Date, il est absurde de donner la valeur "ABC" à une variable utilisée pour des dates. Dans la plupart des cas, 4D est très tolérant et tentera d'utiliser de manière logique ce que vous faites. Par exemple, si vous additionnez un nombre x et une date, 4D déduira que vous voulez ajouter x jours à la date, mais si vous tentez d'ajouter une chaîne à une date, 4D vous préviendra que cette opération est impossible.

Certains cas nécessitent que vous stockiez des données dans un type et que vous les utilisiez dans un autre. Le langage contient un ensemble complet de commandes vous permettant de convertir des types de données en d'autres types. Par exemple, si vous voulez créer un numéro de matricule commençant par des chiffres et se terminant par des lettres, telles que "abc", vous pouvez écrire : vous pouvez écrire :

```
[Produits]Matricule:=String(Numéro)+"abc"
```

Si Numéro vaut 17, [Produits]Matricule prendra la valeur "17abc".

Les types de données sont détaillés dans la section [Types de données](#).

Objets et collections

Vous pouvez gérer les objets et collections du langage 4D à l'aide de la notation objet pour lire ou définir leurs valeurs. Par exemple :

```
employee.name:="Smith"
```

Vous pouvez également utiliser de crochets, comme dans l'exemple ci-dessous :

```
$vName:=employee["nom"]
```

Comme la valeur d'une propriété d'objet peut elle-même être un objet ou une collection, la notation objet requiert une séquence de symboles pour accéder aux sous-propriétés, par exemple :

```
$vAge:=employee.children[2].age
```

A noter que si la valeur de la propriété de l'objet est un objet qui encapsule une méthode (une formule), vous devez ajouter des parenthèses () au nom de la propriété pour exécuter la méthode :

```

$f:=New object
$f.message:=New formula(ALERT("Hello world!"))
$f.message() //affiche "Hello world!"
$f.message() //affiche "Hello world!"

```

Pour accéder à un élément de collection, vous devez passer le numéro de l'élément situé entre crochets :

```

C_COLLECTION(myColl)
myColl:=New collection("A";"B";1;2;Current time)
myColl[3] //accède au 4ème élément de la collection

```

Classes

Le langage 4D prend en charge les classes d'objets. Ajoutez un fichier `myClass.4dm` dans le dossier Project/Sources/Classes d'un projet pour créer une classe nommée "myClass".

Pour instancier un objet de la classe dans une méthode, appelez la classe utilisateur à partir du `class store` (`cs`) et utilisez la fonction membre `new()`. Vous pouvez passer des paramètres.

```

// dans une méthode 4D
$o:=cs.myClass.new()

```

Dans la méthode de classe `myClass`, utilisez l'instruction `Function <methodName>` pour définir la méthode membre de classe `methodName`. Une méthode membre de classe peut recevoir et retourner des paramètres comme n'importe quelle méthode, et utiliser `This` comme instance d'objet.

```

// dans le fichier myClass.4dm
Fonction bonjour
  C_TEXT (0 $)
  $0:= "Hello" + This.who

```

Pour exécuter une méthode membre de classe, utilisez simplement l'opérateur `()` sur la méthode membre de l'instance d'objet.

```

$o:=cs.myClass.new()
$o.who:="World"
$message:=$o.myClass.hello()
//$message: "Hello World"

```

Vous pouvez utiliser le mot-clé `Class constructor` pour déclarer les propriétés de l'objet.

```

//dans le fichier Rectangle.4dm
Class constructor
C_LONGINT($1;$2)
This.height:=$1
This.width:=$2
This.name:="Rectangle"

```

Une classe peut étendre une autre classe en utilisant `Class extends<ClassName>`. Les superclasses peuvent être appelées à l'aide de la commande `Super`. Par exemple :

```
//dans le fichier Square.4dm
Class extends rectangle

Class constructor
C_LONGINT($1)

// il appelle le constructor de la classe mère avec les dimensions
// fournies pour la largeur et la hauteur du Rectangle
Super($1;$1)

This.name:="Square"
```

Opérateurs

Lorsque vous programmez avec 4D, il est rare que vous ayez simplement besoin de données "brutes". Le plus souvent, il sera nécessaire de traiter ces données d'une manière ou d'une autre. Vous effectuez ces calculs avec des opérateurs. Les opérateurs, en général, prennent deux valeurs et effectuent avec elles une opération dont le résultat est une troisième valeur. Vous connaissez déjà la plupart des opérateurs. Par exemple, $1 + 2$ utilise l'opérateur d'addition (ou signe plus) pour faire la somme de deux nombres, et le résultat est 3. Le tableau ci-dessous présente quelques opérateurs courants :

Opérateur	Opération	Exemple
+	Addition	$1 + 2 = 3$
-	Soustraction	$3 - 2 = 1$
*	Multiplication	$2 * 3 = 6$
/	Division	$6 / 2 = 3$

Les opérateurs numériques ne représentent qu'un seul des différents types d'opérateurs disponibles. Comme 4D traite de multiples types de données, tels que des nombres, des dates ou des images, vous disposez d'opérateurs particuliers effectuant des opérations sur ces données.

Souvent, les mêmes symboles sont utilisés pour des opérations différentes, en fonction du type de données traitées. Par exemple, le signe (+) peut effectuer diverses opérations, comme le montre le tableau suivant :

Type de données	Opération	Exemple
Nombre	Addition	1 + 2 ajoute les nombres, le résultat est 3
Chaine	Concaténation	"Bonjour" + "à tous" concatène (met bout à bout) les chaînes, le résultat est "Bonjour à tous"
Date et Numérique	Addition de date	!1989-01-01! + 20 ajoute 20 jours à la date 1 janvier 1989, le résultat est la date 21 janvier 1989

Expressions

Pour parler simplement, les expressions retournent une valeur. En fait, lorsque vous programmez avec 4D, vous utilisez systématiquement des expressions et avez tendance à les manipuler uniquement à travers la valeur qu'elles représentent. Les expressions sont aussi appelées formules.

Les expressions peuvent être constituées de presque tous les éléments du langage : commandes, opérateurs, variables, champs, propriétés d'objets et éléments de collection. Vous utilisez des expressions pour écrire des lignes de code, qui sont à leur tour utilisées pour construire des méthodes. Des expressions sont employées à chaque fois que le langage 4D a besoin de connaître la valeur d'une donnée.

Les expressions sont rarement «autonomes». Il existe plusieurs endroits dans 4D où une expression peut être utilisée seule. Par exemple :

- Editeur de formule (apply formula, query with formula, order by formula)
- La commande `EXECUTE FORMULA`
- La liste de propriétés, où une expression peut être utilisée en tant que source de données pour la plupart des widgets
- Dans la fenêtre du Débogueur où la valeur des expressions peut être évaluée
- Dans l'éditeur d'états semi-automatiques en tant que formule dans une colonne

Types d'expressions

Vous vous référez à une expression via le type de données qu'elle retourne. Il existe plusieurs types d'expressions : Il existe plusieurs types d'expressions : Le tableau suivant fournit des exemples de chaque type d'expression.

Expression	Type	Description
"Bonjour"	Chaine	Le mot Bonjour est une constante chaîne, signalée par les guillemets.
"Bonjour " + "à tous"	Chaine	Deux chaînes, "Bonjour " et "à tous", sont mises bout à bout (concaténées) à l'aide de l'opérateur de concaténation de chaînes (+). La chaîne "Bonjour à tous" est retournée.
"Mr. " + [Personnes]Nom	Chaine	Deux chaînes sont concaténées : la chaîne "Mr." et la valeur courante du champ Nom dans la table Personnes. Si le champ contient "Dupont", l'expression retourne "M. Dupont".
Uppercase("smith")	Chaine	Cette expression utilise Uppercase , une commande du langage, pour convertir la chaîne "dupont" en majuscules. Elle retourne "DUPONT".
4	Nombre	C'est une constante numérique, 4.
4 * 2	Nombre	Deux nombres, 4 et 2, sont multipliés à l'aide de l'opérateur de multiplication (*). Le résultat est le nombre 8.
MonBouton	Nombre	C'est le nom d'un bouton. Il retourne la valeur courante du bouton : 1 s'il y a eu un clic sur le bouton, 0 sinon.
!1997-01-25!	Date	C'est une constante date pour la date 25/01/97 (25 janvier 1997).
Current date+ 30	Date	C'est une expression de type Date qui utilise la commande Current date pour récupérer la date courante. Elle ajoute 30 jours à la date d'aujourd'hui et retourne la nouvelle date.
?8:05:30?	Heure	C'est une constante heure qui représente 8 heures, 5 minutes, et 30 secondes.
?2:03:04? + ?1:02:03?	Heure	Cette expression ajoute une heure à une autre et retourne l'heure 3:05:07.
Vrai	Booléen	Cette commande retourne la valeur booléenne TRUE.
10 # 20	Booléen	C'est une comparaison logique entre deux nombres. Le symbole (#) signifie "est différent de". Comme 10 "est différent de" 20, l'expression retourne TRUE.
"ABC" = "XYZ"	Booléen	C'est une comparaison logique entre deux chaînes. Elles sont différentes, donc l'expression retourne FALSE.
MonImage + 50	Image	Cette expression considère l'image placée dans MonImage, la déplace de 50 pixels vers la droite, et retourne l'image résultante.
->[Personnes]Nom	Pointeur	Cette expression retourne un pointeur vers le champ [Amis]Nom.
Table(1)	Pointeur	C'est une commande qui retourne un pointeur vers la première table.
JSON Parse (MaChaine)	Object	C'est une commande qui retourne MaChaine sous forme d'objet (si format adéquat)
JSON Parse (MonTabJSON)	Collection	C'est une commande qui retourne MonTabJSON sous forme de collection (si format adéquat)
Form.pageNumber	Propriété objet	Une propriété objet est une expression qui peut être de tout type
Col[5]	Élément de collection	Un élément de collection est une expression qui peut être de tout type
\$entitySel[0]	Entity	Un élément d'une sélection d'entité ORDA est une expression de type entité. Ce type d'expression n'est pas affectable

Expressions assignables et non-assignables

Une expression peut simplement être une constante littérale, telle que le chiffre 4 ou la chaîne "Hello", ou une variable telle que \$myButton . Elle peut également utiliser des opérateurs. Par exemple, 4 + 2 est une expression qui utilise l'opérateur d'addition pour additionner deux nombres et renvoyer le résultat 6. Dans tous les cas, ces expressions sont

non-assignables, ce qui signifie que vous ne pouvez pas leur affecter de valeur. Dans 4D, les expressions peuvent être assignables. Une expression est assignable quand elle peut être utilisée à gauche de l'opérateur d'assignation. Par exemple :

```
//La variable $myVar est assignable, vous pouvez écrire :  
$myVar:="Hello" //assigner "Hello" à myVar  
//Form.pageNumber est assignable, vous pouvez écrire :  
Form.pageNumber:=10 //assigne 10 à Form.pageNumber  
//Form.pageTotal-Form.pageNumber n'est pas assignable :  
Form.pageTotal- Form.pageNumber:=10 //erreur, non assignable
```

En général, les expressions qui utilisent un opérateur ne sont pas assignables. Par exemple, [Personne] Prénom " " + [Personne]Nom n'est pas assignable.

Pointeurs

Le langage 4D fournit une mise en oeuvre avancée des pointeurs, pour vous permettre d'écrire un code puissant et modulaire. Vous pouvez utiliser des pointeurs pour référencer des tables, des champs, des variables, des tableaux et des éléments de tableaux.

Un pointeur sur un élément est créé en ajoutant un symbole "->" avant le nom de l'élément, et peut être déréférencé en ajoutant le symbole "->" après le nom du pointeur.

```
MaVar:="Bonjour"  
MonPointeur->->MaVar  
ALERT(MonPointeur->)
```

Commentaires

Les commentaires sont des lignes d'instructions inactives. Ces lignes ne sont pas interprétées par le programme (4D n'applique aucun style spécifique à l'intérieur de la ligne de commentaire) et ne sont pas exécutées lorsque la méthode est appelée.

Voici deux manières de créer des commentaires :

- // pour créer une ligne de commentaire
- /*...*/ pour les blocs de commentaire en ligne et multi-lignes.

Les deux styles de commentaires peuvent être utilisés simultanément.

Ligne de commentaire (//)

Insérez les caractères // au début de la ligne ou après une instruction pour ajouter une ligne de commentaire. Exemple :

```
//Ceci est un commentaire  
For($vCounter;1;100) //Début de la boucle  
    //commentaire  
    //commentaire  
    //commentaire  
End for
```

Commentaires en ligne ou multi-lignes /* */

Entourez le contenu avec des caractères /* ... */ pour créer des commentaires en ligne ou des blocs de commentaires multilignes. Les blocs de commentaire en ligne et multi-lignes commencent par /* et se terminent par */ .

- Les lignes de commentaires en ligne - peuvent être insérées n'importe où dans le code. Exemple :

```
For /* ligne de commentaire */ ($vCounter;1;100)
...
End for
```

- Les blocs de commentaires multi-lignes permettent de commenter un nombre illimité de lignes. Les blocs de commentaires peuvent être imbriqués (ce qui est utile, étant donné que l'éditeur de code 4D prend en charge les blocs condensés). Exemple :

```
For ($vCounter;1;100)
/*
commentaires
/*
    autres commentaires
*/
*/
...
End for
```

Opérateurs

Un opérateur est un symbole ou un groupe de symboles que vous utilisez pour vérifier, modifier ou combiner des valeurs. Vous connaissez déjà la plupart des opérateurs. Par exemple, `1 + 2` utilise l'opérateur d'addition (ou le signe "plus") pour additionner deux nombres, et a pour résultat le chiffre 3. Les opérateurs de comparaison, comme `=` ou `>`, vous permettent de comparer deux valeurs ou plus.

Le langage 4D prend en charge les opérateurs que vous connaissez peut-être déjà dans d'autres langages tels que le C ou JavaScript. Toutefois, l'opérateur d'affectation est `:=` pour éviter qu'il ne soit utilisé par erreur lorsque l'opérateur "égal à" (`=`) est prévu. [Les opérateurs de base](#) tels que les opérateurs arithmétiques (`+`, `-`, `*`, `/`, `%...`) et les opérateurs de comparaison (`=`, `>`, `>=...`) peuvent être utilisés avec des nombres, mais aussi avec des types de données au format booléen, texte, date, heure, pointeur ou image. Tout comme JavaScript, le langage 4D prend en charge le concept de valeurs [truthy et falsy](#), qui sont utilisées dans [les opérateurs de court-circuit](#).

Terminologie

Le langage 4D prend en charge les opérateurs binaires et ternaires :

- les opérateurs binaires opèrent sur deux cibles (comme `2 + 3`) et apparaissent entre leurs deux cibles.
- les opérateurs ternaires opèrent sur trois cibles. Comme le C, 4D ne possède qu'un seul opérateur ternaire, [l'opérateur conditionnel ternaire](#) (`a ? b : c`).

Les valeurs que les opérateurs affectent sont des opérandes. Dans l'expression `1 + 2`, le symbole `+` est un opérateur binaire et ses deux opérandes sont les valeurs 1 et 2.

Assignation

L'opérateur d'affectation (`a:=b`) initialise ou met à jour la valeur de `a` avec la valeur de `b` :

```
$myNumber:=3 //assigne 3 à la variable MyNumber
$myDate:=!2018/01/21! //assigne une date littérale
$myLength:=Length("Acme") //assigne le résultat de la commande (4) à $myLength
$col:=New collection // $col est initialisé avec une collection vide
```

Attention à ne PAS confondre l'opérateur d'affectation `:=` avec l'opérateur de comparaison d'égalité `=`. Un opérateur d'affectation différent (et non pas `=`) a été choisi délibérément pour éviter les problèmes et la confusion qui surviennent souvent avec `==` ou `====` dans d'autres langages de programmation. De telles erreurs sont souvent difficiles à reconnaître pour le compilateur et conduisent à un dépannage fastidieux.

Opérateurs basiques

Les résultats des opérateurs dépendent des types de données auxquels ils sont appliqués. Ils sont décrits avec les types de données, dans les sections suivantes :

- [Opérateurs logiques](#) (sur les expressions booléennes)
- [Opérateurs sur les dates](#)
- [Opérateurs sur les heures](#)
- [Opérateurs sur les nombres](#)
- [Opérateurs binaires](#) (sur les expressions d'entiers longs)
- [Opérateurs sur les images](#)
- [Opérateurs sur les pointeurs](#)
- [Opérateurs sur les chaînes](#)

Opérateurs d'affectation composés

4D fournit des opérateurs d'affectation composés qui combinent l'affectation avec une autre opération. L'opérateur d'affectation d'addition (`+=`) en est un exemple :

```
$a:=1  
$a+=2 // $a=3
```

Les opérateurs d'affectation composés suivants sont pris en charge :

Opérateur	Syntaxe	Assigné	Exemple
Addition	Text += Text	Text	<code>\$t+=" World" // \$t:=\$t+" World"</code>
	Number += Number	Nombre	<code>\$n+=5 // \$n:=\$n+5</code>
	Date += Number	Date	<code>\$d+=5 // \$d:=\$d+5</code>
	Time += Time	Heure	<code>\$t1+=\$t2 // \$t1:=\$t1+\$t2</code>
	Time += Number	Nombre	<code>\$t1+=5 // \$t1:=\$t1+5</code>
	Picture += Picture	Image	<code>\$p1+=\$p2 // \$p1:=\$p1+\$p2 (add \$p2 to the right of \$p1)</code>
	Picture += Number	Image	<code>\$p1+=5 // \$p1:=\$p1+5 (move \$p1 horizontally 5 pixels to the right)</code>
Soustraction	Number -= Number	Nombre	<code>\$n-=5 // \$n:=\$n-5</code>
	Date -= Number	Date	<code>\$d-=5 // \$d:=\$d-5</code>
	Time -= Time	Heure	<code>\$t1-=\$t2 // \$t1:=\$t1-\$t2</code>
	Time -= Number	Nombre	<code>\$t1-=5 // \$t1:=\$t1-5</code>
	Picture -= Number	Image	<code>\$p1-=5 // \$p1:=\$p1-5 (déplacer horizontalement \$p1 de 5 pixels vers la gauche)</code>
Division	Number /= Number	Nombre	<code>\$n/=5 // \$n:=\$n/5</code>
	Time /= Time	Heure	<code>\$t1/=\$t2 // \$t1:=\$t1/\$t2</code>
	Time /= Number	Nombre	<code>\$t1/=5 // \$t1:=\$t1/5</code>
	Picture /= Picture	Image	<code>\$p1/=\$p2 // \$p1:=\$p1/\$p2 (ajouter \$p2 vers le bas de \$p1)</code>
	Picture /= Number	Image	<code>\$p1/=5 // \$p1:=\$p1/5 (déplacer verticalement \$p1 de 5 pixels)</code>
Multiplication	Text *= Number	Text	<code>\$t*="abc" // \$t:=\$t*"abc"</code>
	Number *= Number	Nombre	<code>\$n*=5 // \$n:=\$n*5</code>
	Time *= Time	Heure	<code>\$t1*=\$t2 // \$t1:=\$t1*\$t2</code>
	Time *= Number	Nombre	<code>\$t1*=5 // \$t1:=\$t1*5</code>
	Picture *= Number	Image	<code>\$p1*=5 // \$p1:=\$p1*5 (redimensionner \$p1 de 5)</code>

Ces opérateurs s'appliquent à toutes les [expressions assignables](#) (à l'exception des images en tant que propriétés d'objet ou éléments de collection).

L'opération "source operator value" n'est pas strictement équivalente à "source := source operator valeur" car l'expression désignant la source (variable, champ, propriété d'objet, élément de collection) n'est évaluée qu'une seule fois. Par exemple, dans une expression telle que `getPointer()->+=1`, la méthode `getPointer` n'est appelée qu'une seule fois.

L'indexation des caractères dans le texte et l'indexation des octets dans le blob ne prennent pas en charge ces opérateurs.

Exemples

```
// Addition
$x:=2
$x+=5 // $x=7

$t:="Hello"
$t+=" World" // $t="Hello World"

$d:=!2000-11-10!
$d+=10 // $d=!2000-11-20!

// Soustraction
$x1:=10
$x1-=5 // $x1=5

$d1:=!2000-11-10!
$d1-=10 // $d1=!2000-10-31!

// Division
$x3:=10
$x3/=2 // $x3=5

// Multiplication
$x2:=10
$x2*=5 // $x2=10

$t2:="Hello"
$t2*=2 // $t2="HelloHello"
```

Short-circuit operators

The `&&` and `||` operators are short circuit operators. A short circuit operator is one that doesn't necessarily evaluate all of its operands.

The difference with the single [& and | boolean operators](#) is that the short-circuit operators `&&` and `||` don't return a boolean value. They evaluate expressions as [truthy or falsy](#), then return one of the expressions.

Short-circuit AND operator (`&&`)

The rule is as follows:

Given `Expr1 && Expr2` :

The short-circuit AND operator evaluates operands from left to right, returning immediately with the value of the first falsy operand it encounters; if all values are [truthy](#), the value of the last operand is returned.

The following table summarizes the different cases for the `&&` operator:

Expr1	Expr2	Value returned
truthy	truthy	Expr2
truthy	falsy	Expr2
falsy	truthy	Expr1
falsy	falsy	Expr1

Exemple 1

```
var $v : Variant

$v:= "Hello" && "World" //"World"
$v:=False && 0 // False
$v:=0 && False // False
$v:=5 && !00-00-00! // 00/00/00
$v := 5 && 10 && "hello" //"hello"
```

Exemple 2

Say you have an online store, and some products have a tax rate applied, and others don't.

To calculate the tax, you multiply the price by the tax rate, which may not have been specified.

So you can write this:

```
var $tax : Variant

$tax:=$item.taxRate && ($item.price*$item.taxRate)
```

\$tax will be NULL if taxRate is NULL (or undefined), otherwise it will store the result of the calculation.

Exemple 3

Short-circuit operators are useful in tests such as:

```
If(($myObject#Null) && ($myObject.value>10))
  //code
End if
```

If \$myObject is Null, the second argument is not executed, thus no error is thrown.

Short-circuit OR operator (||)

The || operator returns the value of one of the specified operands. The expression is evaluated left to right and tested for possible "short-circuit" evaluation using the following rule:

Given Expr1 || Expr2 :

If Expr1 is [truthy](#), Expr2 is not evaluated and the calculation returns Expr1.

If Expr1 is [falsy](#), the calculation returns Expr2.

The following table summarizes the different cases and the value returned for the || operator:

Expr1	Expr2	Value returned
truthy	truthy	Expr1
truthy	falsy	Expr1
falsy	truthy	Expr2
falsy	falsy	Expr2

Exemple 1

Say you have a table called Employee. Some employees have entered a phone number, and others haven't. This means that `$emp.phone` could be NULL, and you cannot assign NULL to a Text variable. But you can write the following:

```
var $phone : Text
$phone:=$emp.phone || "n/a"
```

In which case `$phone` will store either a phone number or the "n/a" string.

Exemple 2

Given a table called Person with a *name* field, as well as a *maiden name* field for married women.

The following example checks if there is a maiden name and stores it in a variable, otherwise it simply stores the person's name:

```
var $name: Text
$name:=$person.maidenName || $person.name
```

Priorité

The `&&` and `||` operators have the same precedence as the logical operators `&` and `|`, and are evaluated left to right.

This means that `a || b && c` is evaluated as `(a || b) && c`.

Opérateur ternaire

L'opérateur conditionnel ternaire vous permet d'écrire des expressions conditionnelles sur une seule ligne. Par exemple, il peut remplacer une séquence complète d'instructions [If...Else](#).

Il prend trois opérandes dans l'ordre suivant :

- une condition suivie d'un point d'interrogation (?)
- une expression à exécuter si la condition est `truthy`, suivie de deux points (:)
- une expression à exécuter si la condition est `falsy`

Syntaxe

La syntaxe est la suivante :

```
condition ? exprIfTruthy : exprIfFalsy
```

Since the [token syntax](#) uses colons, we recommend inserting a space after the colon `:` or enclosing tokens using parentheses to avoid any conflicts.

Exemples

A simple example

```
var $age : Integer
var $beverage : Text

$age:=26
$beverage:=($age>=21) ? "Beer" : "Juice"

ALERT($beverage) // "Beer"
```

Handling data from a table

This example stores a person's full name in a variable, and handles the case when no first name or last name has been specified:

```
var $fullname : Text

// If one of the names is missing, store the one that exists, otherwise store an empty string
$fullname:=($person.firstname && $person.lastname) ? ($person.firstname+" "+$person.lastname) : ($person
```

Truthy et falsy

En plus d'un type, chaque valeur possède également une valeur booléenne inhérente, généralement connue sous le nom de **truthy** ou **falsy**.

truthy and falsy values are only evaluated by [short-circuit](#) and [ternary](#) operators.

Les valeurs suivantes sont falsy:

- false
- Null
- indéfini
- Null object
- Null collection
- Null pointer
- Null picture
- Null date !00-00-00!
- "" - Chaînes vides
- [] - Collections vides
- {} - Objets vides

Toutes les autres valeurs sont considérées comme truthy, y compris :

- 0 - zéro numérique (Entier ou autre)

In 4D, truthy and falsy evaluation reflects the usability of a value, which means that a truthy value exists and can be processed by the code without generating errors or unexpected results. The rationale behind this is to provide a convenient way to handle *undefined* and *null* values in objects and collections, so that a reduced number of [If...Else](#) statements are necessary to avoid runtime errors.

For example, when you use a [short-circuit OR operator](#):

```
$value:=$object.value || $defaultValue
```

... you get the default value whenever `$object` does not contain the `value` property OR when it is `null`. So this operator checks the existence or usability of the value instead of a specific value. Note that because the numerical value 0 exists and is usable, it is not treated specially, thus it is truthy.

Regarding values representing collections, objects, or strings, "empty" values are considered falsy. It is handy when you want to assign a default value whenever an empty one is encountered.

```
$phone:=$emp.phone || "n/a"
```

Types de données

Dans 4D, les données sont gérées selon leur type à deux endroits : dans les champs de la base et dans le langage 4D.

Bien qu'ils soient généralement équivalents, certains types de données de la base ne sont pas disponibles dans le langage et sont automatiquement convertis. A l'inverse, certains types de données sont gérés uniquement par le langage. Le tableau suivant liste tous les types de données disponibles, leur prise en charge et leur déclaration :

Types de données	Pris en charge par la base(1)	Pris en charge par le langage	déclaration var	déclaration C_ ou ARRAY
Alphanumérique	Oui	Converti en texte	-	-
Text	Oui	Oui	Text	C_TEXT , ARRAY TEXT
Date	Oui	Oui	Date	C_DATE , ARRAY DATE
Heure	Oui	Oui	Heure	C_TIME , ARRAY TIME
Booléen	Oui	Oui	Booléen	C_BOOLEAN , ARRAY BOOLEAN
Integer	Oui	Converti en entier long	Integer	ARRAY INTEGER
Longint	Oui	Oui	Integer	C_LONGINT , ARRAY LONGINT
Entier long 64 bits	Oui (SQL)	Converti en réel	-	-
Réel	Oui	Oui	Réel	C_REAL , ARRAY REAL
Indéfini	-	Oui	-	-
Null	-	Oui	-	-
Pointeur	-	Oui	Pointeur	C_POINTER , ARRAY POINTER
Image	Oui	Oui	Image	C_PICTURE , ARRAY PICTURE
BLOB	Oui	Oui	Blob , 4D.Blob	C_BLOB , ARRAY BLOB
Object	Oui	Oui	Object	C_OBJECT , ARRAY OBJECT
Collection	-	Oui	Collection	C_COLLECTION
Variant(2)	-	Oui	Variant	C_VARIANT

(1) A noter que ORDA gère les champs de la base via des objets (entités). Par conséquent, seuls les types de données disponibles pour ces objets sont pris en charge. Pour plus d'informations, veuillez vous reporter à la description du type [Objet](#).

(2) Le variant n'est pas un type de *données* un type de *variable* qui peut contenir une valeur de n'importe quel autre type.

Valeurs par défaut

Au moment de leur typage via une directive de compilation, les variables reçoivent une valeur par défaut, qu'elles conserveront au cours de la session tant qu'elles n'auront pas été affectées.

La valeur par défaut dépend du type de variable :

Type	Valeur par défaut
Booléen	Faux
Date	00-00-00
Longint	0
Heure	00:00:00
Image	picture size=0
Réel	0
Pointeur	Nil=true
Text	""
Blob	Blob size=0
Object	null
Collection	null
Variant	indéfini

Convertir les types de données

Le langage de 4D comporte des fonctions et des opérateurs vous permettant de convertir des types de données en d'autres types, dans la mesure où de telles conversions ont un sens. 4D assure la vérification des types de données. Ainsi, vous ne pouvez pas écrire : "abc"+0.5+!25/12/96!-?00:30:45?, car cette opération génère une erreur de syntaxe.

Le tableau ci-dessous liste les types de données pouvant être convertis, le type dans lequel ils peuvent être convertis, ainsi que les fonctions 4D à utiliser :

Types à convertir	en Chaîne	en Numérique	en Date	en Heure	en Booléen
Chaîne (1)		Num	Date	Heure	Bool
Numérique (2)	Chaine				Bool
Date	Chaine				Bool
Heure	Chaine				Bool
Booléen		Num			

(1) Les chaînes formatées en JSON peuvent être converties en données scalaires, objets ou collections à l'aide de la commande `JSON Parse`.

(2) Les valeurs de type Heure peuvent être utilisées en tant que numériques.

Note : Ce tableau ne traite pas les conversions de données plus complexes obtenues à l'aide d'une combinaison d'opérateurs et d'autres commandes.

BLOB

Un champ, une variable ou une expression de type BLOB (Binary Large OBject) est une série contiguë d'octets qui peut être traitée comme un seul objet ou dont les octets peuvent être adressés individuellement.

Lorsque vous travaillez avec un blob, il est stocké entièrement en mémoire. Si vous travaillez avec une variable, le blob n'existe qu'en mémoire. Si vous travaillez avec un champ de type blob, il est chargé en mémoire à partir du disque, comme le reste de l'enregistrement auquel il appartient.

A l'instar d'autres types de champs pouvant contenir une grande quantité de données (comme les champs de type Image), les champs de type blob ne sont pas dupliqués en mémoire lorsque vous modifiez un enregistrement. Par conséquent, les résultats renvoyés par `Ancien` et `Modifie` ne sont pas significatifs lorsque ces fonctions sont appliquées à des champs de type blob.

Types de Blob

Dans le langage 4D, il existe deux façons de manipuler un blob :

- comme une valeur scalaire : un blob peut être stocké dans une variable ou un champ Blob et peut être modifié.
- comme un objet (`4D.Blob`) : un `4D.Blob` est un objet blob. Vous pouvez encapsuler un blob ou une partie de celui-ci dans un `4D.Blob` sans modifier le blob d'origine. Cette méthode est appelée "**boxing**". Pour plus d'informations sur l'instanciation d'un `4D.Blob`, consultez la rubrique [Blob Class](#).

Chaque type de blob a ses avantages. Utilisez le tableau suivant pour déterminer celui qui convient à vos besoins :

	Blob	4D.Blob
Modifiable	Oui	Non
Partageable en objets et collections	Non	Oui
Passé par référence*	Non	Oui
Performances lors de l'accès aux octets	+	-
Taille maximale	2Go	Mémoire

*Contrairement aux commandes 4D conçues pour prendre un blob scalaire en paramètre, le fait de passer un blob scalaire à une méthode le duplique en mémoire. Lorsque vous travaillez avec des méthodes, l'utilisation d'objets blob (`4D.Blob`) est plus efficace, car ils sont passés par référence.

Par défaut, la taille maximale des blobs scalaires est fixée à 2 Go, mais cette limite peut être inférieure en fonction de votre OS et de l'espace disponible.

Vous ne pouvez pas utiliser d'opérateurs sur les blobs.

Vérifier si une variable contient un blob scalaire ou un `4D.Blob`

Utilisez la commande `Value type` pour déterminer si une valeur est de type Blob ou Object. Pour vérifier qu'un objet est un objet blob (`4D.Blob`), utilisez `OB instance of` :

```
var $myBlob: Blob
var $myBlobObject: 4D.Blob
$myBlobObject:=4D.Blob.new()

$type:= Value type($myblobObject) // 38 (object)
$is4DBlob:= OB Instance of($myblobObject; 4D.Blob) //True
```

Passer des blobs en tant que paramètres

Les blobs scalaires et les objets blob peuvent être passés comme paramètres aux commandes 4D ou aux routines de plug-in qui attendent des paramètres blob.

Passer des blobs et des objets blob aux commandes 4D

Vous pouvez passer un blob scalaire ou un `4D.Blob` à toute commande 4D qui prend un blob comme paramètre :

```
var $myBlob: 4D.Blob
CONVERT FROM TEXT("Hello, World!"; "UTF-8"; $myBlob)
$myText:= BLOB to text( $myBlob ; UTF8 text without length )
```

Certaines commandes 4D modifient le blob d'origine et ne prennent donc pas en charge le type `4D.Blob` :

- [DELETE FROM BLOB](#)
- [INSERT IN BLOB](#)
- [INTEGER TO BLOB](#)
- [LONGINT TO BLOB](#)
- [REAL TO BLOB](#)
- [SET BLOB SIZE](#)
- [TEXT TO BLOB](#)
- [VARIABLE TO BLOB](#)
- [LIST TO BLOB](#)
- [SOAP DECLARATION](#)
- [WEB SERVICE SET PARAMETER](#)

Passer des blobs et des objets blob aux méthodes

Vous pouvez passer des blobs et des objets blob (`4D.Blob`) aux méthodes. A noter que, contrairement aux objets blob, qui sont transmis par référence, les blobs scalaires sont dupliqués en mémoire lorsqu'ils sont passés aux méthodes.

Passer un blob scalaire par référence en utilisant un pointeur

Pour passer un blob scalaire à vos propres méthodes sans le dupliquer en mémoire, définissez un pointeur vers la variable qui le stocke et passez le pointeur comme paramètre.

Voici quelques exemples :

```
// Déclarer une variable de type Blob
var $myBlobVar: Blob
// Passer le blob (en tant que paramètre) en une commande 4D
SET BLOB SIZE($myBlobVar;1024*1024)
```

```
// Passer le blob (en tant que paramètre) en routine externe
$errCode:=Do Something With This blob($myBlobVar)
```

```
//Passer le blob (en tant que paramètre) en une méthode qui retourne un blob
var $retrieveBlob: Blob
retrieveBlob:=Fill_Blob($myBlobVar)
```

```
// Passer un pointeur au blob (en tant que paramètre) à votre propre méthode,
COMPUTE BLOB(>$myBlobVar)
```

Note pour les développeurs de plug ins 4D : Un paramètre de type BLOB se déclare "&O" (la lettre "O" et non le chiffre "0").

Assigner une variable Blob à une autre

Vous pouvez affecter une variable Blob à une autre :

Exemple :

```
// Déclarer deux variables de type Blob
var $vBlobA; $vBlobB : Blob
// Fixer la taille du premier blob à 10Ko
SET BLOB SIZE($vBlobA;10*1024)
// Assigner le premier BLOB au second
$vBlobB:=$vBlobA
```

Conversion automatique du type blob

4D convertit automatiquement les blobs scalaires en objets blob, et vice versa, lorsqu'ils sont assignés l'un à l'autre. Par exemple :

```
// Créer une variable de type Blob et une variable objet
var $myBlob: Blob
var $myObject : Object

// Assigner ce blob à une propriété de $myObject nommée "blob"
$myObject:=New object("blob"; $myBlob)

// le blob stocké dans $myBlob est automatiquement converti en un 4D.Blob
$type:= OB Instance of($myObject.blob; 4D.Blob) //True

// Conversion d'un 4D.Blob en Blob
$myBlob:= $myObject.blob
$type:= Value type($myBlob) // Blob
```

Lors de la conversion d'un `4D.Blob` en un blob scalaire, si la taille du `4D.Blob` dépasse la taille maximale des blobs scalaires, le blob scalaire résultant est vide. Par exemple, lorsque la taille maximale des blobs scalaires est de 2GB, si vous convertissez un `4D.Blob` de 2,5Go en blob scalaire, vous obtenez un blob vide.

Modification d'un blob scalaire

Contrairement aux objets blob, les blobs scalaires peuvent être modifiés. Par exemple :

```
var $myBlob : Blob
SET BLOB SIZE ($myBlob ; 16*1024)
```

Accéder individuellement aux octets d'un blob

Accéder aux octets d'un blob scalaire

Vous pouvez accéder aux octets individuels d'un blob scalaire en utilisant des accolades `{}`. Dans un blob, les octets sont numérotés de 0 à N-1, N étant la taille du BLOB:

```
// Déclarer une variable de type BLOB
var $vBlob : Blob
// Fixer la taille du BLOB à 256 octets
SET BLOB SIZE($vBlob;256)
// Le code suivant initialise les octets du BLOB à zéro
For(vByte;0;BLOB size($vBlob)-1)
    $vBlob{vByte}:=0
End for
```

Etant donné que vous pouvez adresser tous les octets d'un blob individuellement, vous pouvez stocker tout ce que vous souhaitez dans une variable ou un champ Blob.

Accéder aux octets d'un **4D.Blob**

Utilisez les crochets **[]** pour accéder directement à un octet spécifique dans un **4D.Blob**

```
var $myBlob: 4D.Blob
CONVERT FROM TEXT("Hello, World!"; "UTF-8"; $myBlob)
$myText:= BLOB to text ( $myBlob ; UTF8 text without length )
$byte:=$myBlob[5]
```

Etant donné qu'un **4D.Blob** ne peut pas être modifié, vous pouvez lire les octets d'un **4D.Blob** à l'aide de cette syntaxe, mais pas les modifier.

Booléen

Un champ, une variable ou une expression de type booléen peut être soit VRAI soit FAUX.

Fonctions booléennes

Les fonctions booléennes de 4D traitent des valeurs telles que `Vrai`, `Faux` et `Non` dans le thème Booléens consacré. Pour plus d'informations, veuillez vous reporter à la description de ces commandes.

Exemple

L'exemple suivant retourne Vrai dans la variable monBooléen si l'utilisateur a cliqué sur le bouton monBouton et Faux s'il n'a pas cliqué dessus. . Lorsqu'un bouton reçoit un clic, la variable du bouton prend la valeur 1.

```
If(monBouton=1) // Si le bouton a reçu un clic  
    monBooléen:=True// monBooléen prend la valeur True  
Else // Si le bouton n'a pas reçu de clic,  
    monBooléen:=False //monBooléen prend la valeur False  
End if
```

L'exemple ci-dessus peut être simplifié et écrit en une seule ligne .

```
monBooléen:=(monBouton=1)
```

Opérateurs logiques

4D supporte deux opérateurs logiques : l'opérateur d'intersection (AND) et l'opérateur de réunion inclusive (OR). Le AND logique retourne TRUE si les deux expressions sont VRAIES. Le OR logique retourne TRUE si au moins une des expressions est VRAIE. Le tableau suivant décrit les opérateurs logiques :

Opération	Syntaxe	Retourne	Expression	Valeur
AND	Booléen & Booléen	Booléen	("A" = "A") & (15 # 3)	Vrai
			("A" = "B") & (15 # 3)	Faux
			("A" = "B") & (15 = 3)	Faux
OR	Booléen & Booléen	Booléen	("A" = "A") (15 # 3)	Vrai
			("A" = "B") (15 # 3)	Vrai
			("A" = "B") (15 = 3)	Faux

Voici la "table de vérité" pour l'opérateur logique "AND" :

Expr1	Expr2	Expr1 & Expr2
Vrai	Vrai	Vrai
Vrai	Faux	Faux
Faux	Vrai	Faux
Faux	Faux	Faux

Voici la "table de vérité" pour l'opérateur logique "OR" :

Expr1	Expr2	Expr1 Expr2
Vrai	Vrai	Vrai
Vrai	Faux	Vrai
Faux	Vrai	Vrai
Faux	Faux	Faux

Astuce : Si vous devez calculer une réunion exclusive (le "Ou" exclusif) entre Expr1 et Expr2, écrivez :

```
(Expr1|Expr2) & Not(Expr1 & Expr2)
```

Dans les contextes booléens, le langage 4D prend également en charge les [opérateurs de court-circuit](#) (`&&` et `||`) et le concept de [truthy et falsy](#) .

Collection

Les collections sont des listes ordonnées de valeurs de types similaires ou différents (texte, nombre, date, objet, booléen, collection ou null).

Pour manipuler les variables de type Collection, vous devez utiliser la notation objet (voir [Les bases de la syntaxe](#)).

Pour des informations complémentaires sur les collections 4D, passez le numéro (l'indice) de l'élément entre crochets :

```
collectionRef[expression]
```

Vous pouvez passer toute expression 4D valide qui retourne un nombre entier positif dans *expression*. Voici quelques exemples :

```
myCollection[5] //accès au 6e élément de la collection  
myCollection[$var]
```

Attention : N'oubliez pas que la numérotation des éléments de collection débute à 0.

Vous pouvez assigner une valeur à un élément de collection ou lire une valeur d'élément de collection :

```
myCol[10]:="Mon nouvel élément"  
$myVar:=myCol[0]
```

Si vous assignez un numéro d'élément plus grand que celui du dernier élément existant dans la collection, la collection est automatiquement redimensionnée et les nouveaux éléments intermédiaires prennent la valeur null :

```
var myCol : Collection  
myCol:=New collection("A";"B")  
myCol[5]:="Z"  
//myCol[2]=null  
//myCol[3]=null  
//myCol[4]=null
```

Initialisation

Les collections doivent être initialisées à l'aide, par exemple, de la commande `Creer collection`, sinon une erreur de syntaxe sera générée à la suite d'une lecture ou d'une modification d'un ou plusieurs éléments de la collection.

Exemple :

```
var $colVar : Collection //création d'une variable 4D de type collection  
$colVar:=New collection ///initialisation de la collection et assignation à la variable 4D
```

Collection standard ou collection partagée

Vous pouvez créer deux types de collections :

- collections standard (non partagées), à l'aide de la commande `New collection`. Ces collections peuvent être modifiées sans contrôle d'accès spécifique mais ne peuvent pas être partagées entre les process.
- collections partagées, à l'aide de la commande `New shared collection`. Le contenu de ces collections peut être partagé entre les process, y compris des process (thread) préemptifs. L'accès à ces collections doit être contrôlé via

des structures `Use...End use`.

Pour plus d'informations, veuillez vous reporter à la page [Objets partagés et collections partagées](#).

Fonctions de collection

Les références de collections 4D bénéficient de fonctions de classe spécifiques (souvent appelées *fonctions méthodes*). Les fonctions de collection sont répertoriées dans la section [Class API Reference](#).

Par exemple :

```
$newCol:=$col.copy() //copie de $col vers $newCol  
$col.push(10;100) //ajout de 10 et 100 à la collection
```

Certaines fonctions retournent la collection d'origine après modification, de manière à ce que vous puissiez enchaîner les appels dans une même séquence :

```
$col:=New collection(5;20)  
$col2:=$col.push(10;100).sort() //$/col2=[5,10,20,100]
```

paramètre cheminPropriété

Plusieurs fonctions admettent un paramètre nommé *cheminPropriété*. Ce paramètre peut contenir :

- soit un nom de propriété d'objet, par exemple "nomComplet"
- soit un chemin de propriété d'objet, c'est-à-dire une séquence hiérarchique de sous-propriétés reliées par des points, par exemple "employé.enfant.prénom".

Attention : Lorsque des fonctions ou un paramètre *cheminPropriété* sont attendus, l'utilisation de noms de propriétés contenant ".", "[]", ou des espaces n'est pas prise en charge car cela empêcherait 4D d'analyser correctement le chemin :

```
$vmin:=$col.min("My.special.property") //indéfini  
$vmin:=$col.min(["My.special.property"]) //erreur
```

Date

Les variables, champs ou expressions de type Date peuvent être compris entre 1/1/100 et 31/12/32767.

Bien que le mode de représentation des dates par C_DATE permette de manipuler des dates allant jusqu'à l'année 32767, certaines opérations passant par le système imposent une limite plus basse.

Note : Dans ce manuel de référence du langage 4D, les paramètres de type Date dans les descriptions des commandes sont appelés Date, sauf spécification explicite.

Constantes littérales de type date

Une constante littérale de type date est comprise entre deux points d'exclamation (!…!). Une date doit être structurée avec le format ISO (!YYYY-MM-DD!). Voici quelques exemples de constantes dates :

```
!1976-01-01!
!2004-09-29!
!2015-12-31!
```

Une date nulle s'écrit !00-00-00!.

Astuce : L'éditeur de méthodes dispose d'un raccourci pour entrer une date nulle. Pour cela, tapez un point d'exclamation (!) et appuyez sur la touche Entrée.

Notes :

- Pour des raisons de compatibilité, 4D accepte que l'année soit saisie sur deux chiffres. Dans ce cas, le programme considère qu'elle appartient au XXe ou au XXIe siècle selon qu'elle est supérieure ou inférieure à 30, sauf si ce fonctionnement par défaut a été modifié à l'aide de la commande SET DEFAULT CENTURY .
- Si vous avez coché l'option "Utiliser langage français et paramètres régionaux système" (cf. Page Méthodes), vous devez utiliser le format de date défini dans votre système. Généralement dans un environnement français, une date est saisie sous la forme jour/mois/année, une barre oblique "/" séparant les valeurs.

Opérateurs sur les dates

Opération	Syntaxe	Retourne	Expression	Valeur
Différence	Date - Date	Nombre	$!2017-01-20! - !2017-01-01!$	19
Addition	Date + Numérique	Date	$!2017-01-20! + 2017-01-20!$	$!2017-01-29!$
Soustraction	Date - Numérique	Date	$!2017-01-20! - 2017-01-20!$	$!2017-01-11!$
Egalité	Date = Date	Booléen	$!2017-01-20! == !2017-01-01!$	Vrai
			$!2017-01-20! != 2017-01-20!$	Faux
Inégalité	Date # Date	Booléen	$!2017-01-20! < !2017-01-01!$	Vrai
			$!2017-01-20! > !2017-01-20!$	Faux
Supérieur à	Date > Date	Booléen	$!2017-01-20! > !2017-01-20!$	Vrai
			$!2017-01-20! < !2017-01-20!$	Faux
Inférieur à	Date < Date	Booléen	$!2017-01-20! < !2017-01-20!$	Vrai
			$!2017-01-20! >= !2017-01-20!$	Faux
Supérieur ou égal à	Date >= Date	Booléen	$!2017-01-20! >= !2017-01-20!$	Vrai
			$!2017-01-01! <= !2017-01-20!$	Faux
Inférieur ou égal à	Date <= Date	Booléen	$!2017-01-01! <= !2017-01-20!$	Vrai
			$!2017-01-20! <= !2017-01-01!$	Faux

Null et Indefinie

Null et Undefined sont des types de données qui gèrent les cas où la valeur d'une expression n'est pas connue.

Null

Null est un type de données particulier avec une seule valeur possible : null. Cette valeur est retournée par une expression qui ne contient aucune valeur.

Dans le langage de 4D et pour les attributs des champs objets, les valeurs null sont gérées via la commande `Null`. Cette commande peut être utilisée avec les expressions suivantes pour fixer ou comparer la valeur null :

- attributs d'objets
- éléments de collections
- variables de type objet, collection, pointeur, image ou variant.

Indéfini

Indéfinie n'est pas véritablement un type de données. Une variable dite "indéfinie" est une variable n'ayant pas encore été définie. Une fonction utilisateur (c'est-à-dire une méthode projet qui retourne une valeur) peut retourner une valeur indéfinie si, à l'intérieur de la méthode, le résultat de la fonction (\$0) est assigné à une expression indéfinie (une expression issue d'un calcul effectué avec au moins une variable indéfinie). Un champ ne peut pas être indéfini (la commande `Indefinie` retourne toujours Faux pour un champ). Une variable variant porte la valeur par défaut indéfini.

Exemples

Cet exemple compare les différents résultats de la commande `Indefinie` et de la commande `Null` appliquées aux propriétés d'objets, en fonction du contexte :

```
C_OBJECT($vEmp)
$vEmp:=New object
$vEmp.name:="Smith"
$vEmp.children:=Null

$undefined:=Undefined($vEmp.name) // Faux
>null:=( $vEmp.name=NULL) //Faux

$undefined:=Undefined($vEmp.children) // Faux
>null:=( $vEmp.children=NULL) //Vrai

$undefined:=Undefined($vEmp.parent) // Vrai
>null:=( $vEmp.parent=NULL) //Vrai
```

Numérique (Réel, Entier, Entier long)

Numérique est un terme générique utilisé pour :

- Les champs, variables ou expression de type Réel. Les nombres de type Réel sont compris dans l'intervalle $\pm 1.7e\pm 308$ (13 chiffres significatifs).
- Les champs, variables ou expression de type Entier long. Les nombres de type Entier long (4 octets) sont compris dans l'intervalle $-2^{31}..(2^{31}-1)$.
- Les champs, variables ou expression de type Entier. Les nombres de type Entier (2 octets) sont compris dans l'intervalle $-32\ 768..32\ 767$.

Note : Lorsqu'elles sont utilisées dans le langage 4D, les valeurs des champs de type Entier sont automatiquement converties en Entier long.

Vous pouvez assigner tout nombre d'un type numérique à un nombre d'un autre type numérique, 4D effectue automatiquement la conversion, en tronquant ou en arrondissant les valeurs si nécessaire. Notez cependant que lorsqu'une valeur est située en-dehors de l'intervalle du type de destination, 4D ne pourra la convertir. Vous pouvez mélanger tous les types de numériques au sein d'une même expression.

Note : Dans ce manuel de référence du langage 4D, quel que soit le type précis des données, les paramètres de type Réel, Entier et Entier long dans les descriptions des commandes sont appelés numériques, sauf spécification explicite.

Constantes littérales numériques

Une constante littérale numérique s'écrit comme un nombre réel. Voici quelques exemples de constantes numériques :

**27
123.76
0.0076**

Le séparateur décimal est par défaut le point (.), quelle que soit la langue du système. Si vous avez coché l'option "Utiliser langage français et paramètres régionaux système" dans la Page Méthodes des Préférences, vous devez utiliser le séparateur défini dans votre système.

Les nombres négatifs s'écrivent précédés du signe moins (-). Par exemple :

**-27
-123.76
-0.0076**

Opérateurs sur les nombres

Opération	Syntaxe	Retourne	Expression	Valeur
Addition	Nombre + Nombre	Nombre	$2 + 3$	5
Soustraction	Nombre - Nombre	Nombre	$3 - 2$	1
Multiplication	Number * Number	Nombre	$5 * 2$	10
Division	Number / Number	Nombre	$5 / 2$	2.5
Division entière	Nombre ¥ Nombre	Nombre	$5 ¥ 2$	2
Modulo	Nombre % Nombre	Nombre	$5 \% 2$	1
Exponentiation	Nombre ^ Nombre	Nombre	$2 ^ 3$	8
Egalité	Nombre = Nombre	Booléen	$10 = 10$	Vrai
			$10 = 11$	Faux
Inégalité	Nombre # Nombre	Booléen	$10 \# 11$	Vrai
			$10 \# 10$	Faux
Supérieur à	Nombre > Nombre	Booléen	$11 > 10$	Vrai
			$10 > 11$	Faux
Inférieur à	Nombre < Nombre	Booléen	$10 < 11$	Vrai
			$11 < 10$	Faux
Supérieur ou égal à	Nombre >= Nombre	Booléen	$11 >= 10$	Vrai
			$10 >= 11$	Faux
Inférieur ou égal à	Nombre <= Nombre	Booléen	$10 <= 11$	Vrai
			$11 <= 10$	Faux

L'opérateur modulo % divise le premier nombre par le second et retourne le reste de la division entière. Voici quelques exemples :

- $10 \% 2$ retourne 0 car la division de 10 par 2 ne donne pas de reste.
- $10 \% 3$ retourne 1 car le reste est 1.
- $10,5 \% 2$ retourne 0 car le reste n'est pas un nombre entier.

ATTENTION :

- L'opérateur modulo % retourne des valeurs significatives avec des nombres appartenant à la catégorie des entiers longs (de -2^{31} à $+2^{31}$ moins 1). Pour calculer le modulo de nombres qui ne sont pas dans cet intervalle, utilisez la fonction Modulo .
- L'opérateur division entière ¥ retourne des valeurs significatives avec des nombres entiers uniquement.

Priorité

L'ordre dans lequel une expression est évaluée s'appelle la priorité. 4D applique strictement une règle de priorité de gauche à droite. L'ordre algébrique n'est pas appliqué. Par exemple :

3+4*5

retourne 35 car l'expression est évaluée comme $3 + 4$, qui donne 7, multiplié par 5, ce qui donne 35.

Les parenthèses doivent être utilisées pour forcer l'ordre de calcul en fonction de vos besoins. Par exemple :

3+(4*5)

retourne 23 car l'expression `(4 * 5)` est évaluée en premier lieu. Le résultat `(20)` est alors ajouté à `3`, ce qui donne le résultat final `23`.

Des parenthèses peuvent être incluses dans d'autres parenthèses. Assurez-vous qu'il y ait une parenthèse fermante pour chaque parenthèse ouverte. Une parenthèse manquante ou placée à un mauvais endroit peut soit donner un résultat erroné, soit renvoyer une expression invalide. De plus, si vous avez l'intention de compiler vos applications, vous devez vous assurer d'une bonne utilisation des parenthèses. Le compilateur interprétera toute parenthèse manquante ou superflue comme une erreur de syntaxe.

Opérateurs sur les bits

Les opérateurs sur les bits s'appliquent à des expressions ou valeurs de type Entier long.

Si vous passez une valeur de type Entier ou Réel à un opérateur sur les bits, 4D la convertit en Entier long avant de calculer le résultat de l'expression.

Lorsque vous employez des opérateurs sur les bits, vous devez considérer une valeur de type Entier long comme un tableau de 32 bits. Les bits sont numérotés de `0` à `31`, de droite à gauche.

Comme un bit peut valoir `0` (zéro) ou `1`, vous pouvez également considérer une valeur de type Entier long comme une expression dans laquelle vous pouvez stocker 32 valeurs de type Booléen. Lorsque le bit vaut `1`, la valeur est Vrai et lorsque le bit vaut `0`, la valeur est Faux.

Une expression utilisant un opérateur sur les bits retourne une valeur de type Entier long, à l'exception de l'opérateur Tester bit avec lequel l'expression retournée est du type Booléen. Le tableau suivant fournit la liste des opérateurs sur les bits et leur syntaxe :

Opération	Opérateur	Syntaxe	Retourne
ET	<code>&</code>	<code>long & E. long</code>	Entier long
OU (inclusif)	<code> </code>	<code>long E. long</code>	Entier long
OU (exclusif)	<code>^ </code>	<code>long ^ E. long</code>	Entier long
Décaler bits à gauche	<code><<</code>	<code>E. Long << E. Long</code>	Entier long (voir note n°1)
Décaler bits à droite	<code>>></code>	<code>E. Long >> E. Long</code>	Entier long (voir note n°1)
Mettre bit à 1	<code>?+</code>	<code>long ?+ E. E. long</code>	Entier long (voir note n°2)
Mettre bit à 0	<code>?-</code>	<code>long ??</code>	Entier long (voir note n°2)
Tester bit	<code>??</code>	<code>long & E. E. long</code>	Booléen (voir note n°2)

Notes

1. Dans les opérations utilisant `Décaler bits à gauche` et `Décaler bits à droite`, le second opérande indique le nombre de décalages de bits du premier opérande à effectuer dans la valeur retournée. Par conséquent, ce second opérande doit être compris entre `0` et `31`. Notez qu'un décalage de `0` retourne une valeur inchangée et qu'un décalage de plus de `31` bits retourne `0x00000000` car tous les bits sont perdus. Si vous passez une autre valeur en tant que second opérande, le résultat sera non significatif.
2. Dans les opérations utilisant `Mettre bit à 1`, `Mettre bit à 0` et `Tester bit`, le second opérande indique le numéro du bit sur lequel agir. Par conséquent, ce second opérande doit être compris entre `0` et `31`, sinon le résultat de l'expression sera non significatif.

Le tableau suivant dresse la liste des opérateurs sur les bits et de leurs effets :

Opération	Description
ET	<p>Chaque bit retourné est le résultat de l'opération ET logique appliquée aux deux bits opérandes.</p> <p>Voici la table du ET logique :</p> <ul style="list-style-type: none"> • 1 & 1 --> 1 • 0 & 1 --> 0 • 1 & 0 --> 0 • 0 & 0 --> 0 <p>En d'autres termes, le bit résultant est 1 si les deux bits d'opérande sont 1; sinon, le bit résultant est 0.</p>
OU (inclusif)	<p>Chaque bit retourné est le résultat de l'opération OU logique appliquée aux deux bits opérandes.</p> <p>Voici la table du OU inclusif logique :</p> <ul style="list-style-type: none"> • 1 1 --> 1 • 0 1 --> 1 • 1 0 --> 1 • 0 0 --> 0 <p>En d'autres termes, le bit résultant est 1 si au moins l'un des deux bits d'opérande est 1; sinon, le bit résultant est 0.</p>
OU (exclusif)	<p>Chaque bit retourné est le résultat de l'opération OU logique appliquée aux deux bits opérandes.</p> <p>Voici la table du OU exclusif logique :</p> <ul style="list-style-type: none"> • 1 ^ 1 --> 0 • 0 ^ 1 --> 1 • 1 ^ 0 --> 1 • 0 ^ 0 --> 0 <p>En d'autres termes, le bit résultant est 1 si seul l'un des deux bits d'opérande est 1; sinon, le bit résultant est 0.</p>
Décaler bits à gauche	<p>La valeur résultante est définie sur la première valeur d'opérande, puis les bits résultats sont décalés vers la gauche du nombre de positions indiqué par le deuxième opérande. Les bits auparavant situés à gauche sont perdus et les nouveaux bits situés à droite ont la valeur 0.</p> <p>Note: en ne tenant compte que des valeurs positives, le décalage vers la gauche de N bits équivaut à multiplier par 2^N.</p>
Décaler bits à droite	<p>La valeur résultante est définie sur la première valeur d'opérande, puis les bits résultats sont décalés vers la droite du nombre de positions indiqué par le deuxième opérande. Les bits auparavant situés à droite sont perdus et les nouveaux bits situés à gauche ont la valeur 0.</p> <p>Note: en ne tenant compte que des valeurs positives, le décalage vers la droite de N bits équivaut à diviser par 2^N.</p>
Mettre bit à 1	La valeur retournée est la valeur du premier opérande dans lequel le bit dont le numéro est spécifié par le second opérande est positionné à 0. Les autres bits demeurent inchangés.
Mettre bit à 0	La valeur retournée est la valeur du premier opérande dans lequel le bit dont le numéro est spécifié par le second opérande est positionné à 0. Les autres bits demeurent inchangés.
Tester bit	Retourne Vrai si, dans le premier opérande, le bit dont le numéro est indiqué par le second opérande vaut 1. Retourne Faux si, dans le premier opérande, le bit dont le numéro est indiqué par le second opérande vaut 0.

Exemples

Opération	Exemple	Résultat
ET	0x0000FFFF & 0xFF00FF00	0x0000FF00
OU (inclusif)	0x0000FFFF 0xFF00FF00	0xFF00FFFF
OU (exclusif)	0x0000FFFF ^ 0xFF00FF00	0xFF0000FF
Décaler bits à gauche	0x0000FFFF << 8	0x00FFFF00
Décaler bits à droite	0x0000FFFF >> 8	0x000000FF
Mettre bit à 1	0x00000000 ?+ 16	0x00010000
Mettre bit à 0	0x00010000 ?- 16	0x00000000
Tester bit	0x00010000 ?? 16	Vrai

Object

Les variables, champs ou expressions de type objet peuvent contenir des données de divers types. La structure des objets "natifs" 4D est basée sur le principe classique des paires "propriété/valeur" (aussi appelées "attribut/valeur"). La syntaxe de ces objets s'inspire du JSON :

- Un nom de propriété est toujours un texte, par exemple "Nom". Il doit suivre des [règles spécifiques](#).
- Une valeur de propriété peut être du type suivant :
 - Numérique (réel, entier long, etc.)
 - Texte
 - null
 - boolean
 - Pointeur (stocké tel quel, évalué à l'aide de la commande `JSON Stringify` ou lors d'une copie),
 - Date (type date ou chaîne au format date ISO)
 - Objet(1) (les objets peuvent être imbriqués sur plusieurs niveaux)
 - Image(2)
 - collection

(1) Les objets ORDA tels que les [entités](#) ou les [sélections d'entités](#) ne peuvent pas être stockés dans les champs objet; ils sont néanmoins entièrement pris en charge dans les variables objet en mémoire.

(2) Lorsqu'elles sont exposées sous forme de texte dans le débogueur ou exportées en JSON, les propriétés d'objet de type image indiquent "[objet Image]".

Attention : N'oubliez pas que les noms d'attributs tiennent compte des majuscules/minuscules.

Vous gérez les variables, les champs ou les expressions de type Objet à l'aide de la [notation objet](#) ou des commandes classiques du thème Objets (langage). A noter que des commandes spécifiques du thème Requêtes, telles que `QUERY BY ATTRIBUTE`, `QUERY SELECTION BY ATTRIBUTE` ou `ORDER BY ATTRIBUTE` peuvent être utilisées pour traiter des champs objets.

Chaque valeur de propriété accessible par la notation objet est considérée comme une expression. Vous pouvez utiliser ces valeurs partout où des expressions 4D sont attendues :

- Dans le code 4D, soit écrites dans les méthodes (éditeur de méthodes) soit externalisées (formules, fichiers d'étiquettes traités par la commande `PROCESS 4D TAGS` ou le Serveur Web, fichiers d'export, documents 4D Write Pro, etc.),
- Dans les zones d'expressions du débogueur et l'explorateur d'exécution,
- Dans la liste de propriétés de l'éditeur de formulaires pour les objets formulaires : champ Variable ou Expression et plusieurs expressions de list box et colonnes (source de données, couleur de fond, style ou couleur de police).

Initialisation

Les objets doivent être initialisés à l'aide, par exemple, de la commande `New object`, sinon une erreur de syntaxe sera générée à la suite d'une lecture ou d'une modification de leurs propriétés.

Exemple :

```
C_OBJECT($obVar) //création d'une variable 4D de type objet. $obVar:=Creer objet//initialisation de l'ob
```

Objet standard ou partagé

Vous pouvez créer deux types d'objets :

- standard (non partagés), à l'aide de la commande `Creer objet`. Ces objets peuvent être modifiés sans contrôle

d'accès spécifique mais ne peuvent pas être partagés entre les process.

- partagés, à l'aide de la commande `New shared object`. Le contenu de ces objets peut être partagé entre les process, y compris des process (thread) préemptifs. L'accès à ces objets doit être contrôlé via des structures `Use...End use`. Pour plus d'informations, veuillez vous reporter à la page [Objets partagés et collections partagées](#).

Principes de syntaxe

La notation objet est utilisée pour accéder aux valeurs de propriétés d'objets via des séquences de symboles et de propriétés référencées (tokens).

Propriétés des objets

Avec la notation objet, il est possible d'accéder aux propriétés d'objets (aussi appelées attributs d'objets) de deux façons :

- à l'aide du symbole "point" : `> objet.NomPropriété`

Exemple :

```
employee.name:="Smith"
```

- à l'aide d'une chaîne entre crochets : `> objet["NomPropriété"]`

Voici quelques exemples :

```
$vName:=employee["name"]
//ou :
$property:="name"
$vName:=employee[$property]
```

Comme la valeur d'une propriété d'objet peut elle-même être un objet ou une collection, la notation objet requiert une séquence de symboles pour accéder aux sous-propriétés, par exemple :

```
$vAge:=employee.children[2].age
```

La notation objet est utilisable avec tout élément de langage qui contient ou retourne un objet, c'est-à-dire :

- avec les objets eux-mêmes (stockés dans des variables, champs, propriétés d'objets, tableaux d'objets ou éléments de collections). Voici quelques exemples :

```
$age:=$myObjVar.employee.age //variable
$addr:=[Emp]data_obj.address //champ
$city:=$addr.city //propriété d'un objet
$pop:=$a0bjCountries{2}.population //tableau d'objets
$val:=$myCollection[3].subvalue //élément de collection
```

- avec les commandes 4D qui retournent des objets. Exemple :

```
$measures:=Lire mesures base.DB.tables
```

- avec les méthodes projet qui retournent des objets. Exemple :

```
// MyMethod1
C_OBJECT($0)
$0:=New object("a";10;"b";20)

//myMethod2
$result:=MyMethod1.a //10
```

- Collections Exemple :

```
myColl.length //taille de la collection
```

Pointeurs

Note : Les objets étant toujours passés par référence, l'utilisation de pointeurs n'est généralement pas nécessaire. En passant un objet, 4D utilise automatiquement, en interne, un mécanisme similaire à un pointeur pour minimiser la mémoire nécessaire, pour vous permettre de modifier le paramètre et de retourner les modifications. Par conséquent, vous n'aurez pas besoin d'utiliser des pointeurs. Cependant, si vous souhaitez utiliser des pointeurs, il est possible d'accéder aux valeurs de propriétés via des pointeurs.

La notation objet pour les pointeurs est semblable à la notation objet standard, à la seule différence que le symbole "point" doit être omis.

- Accès direct :

```
pointeurObjet->nomPropriété
```

- Accès par le nom :

```
pointeurObjet-> nomPropriété"]
```

Exemple :

```
C_OBJECT(v0bj)
C_POINTER(vPptr)
v0bj:=New object
v0bj.a:=10
vPptr:=->v0bj
x:=vPptr->a //x=10
```

Valeur Null

Lorsque la notation objet est utilisée, la valeur null est prise en charge via la commande Null. Cette commande peut être utilisée pour affecter ou comparer la valeur null aux propriétés d'objets ou aux éléments de collections, par exemple :

```
myObject.address.zip:=Null
If(myColl[2]=Null)
```

Pour plus d'informations, veuillez vous reporter à la description de la commande `Null`.

Valeur Indéfinie

L'évaluation d'une propriété d'objet peut parfois produire une valeur indéfinie (undefined). En règle générale, lorsque le code tente de lire ou d'affecter des expressions indéfinies, 4D génère des erreurs, hormis dans les cas décrits ci-

dessous :

- La lecture d'une propriété d'un objet ou d'une valeur indéfini(e) retourne Indéfini ; l'affectation d'une valeur indéfinie à des variables (hors tableaux) a le même effet qu'appeler `CLEAR VARIABLE` avec elles :

```
C_OBJET($o)
C_ENTIER Long($val)
$val:=10 // $val=10
$val:=$o.a // $o.a est indéfini (pas d'erreur), et affecter cette valeur efface la variable
// $val=0
```

- La lecture de la propriété length d'une collection indéfinie renvoie 0 :

```
C_COLLECTION($c) // variable créée mais pas de collection définie
$csize:=$c.length // $size = 0
```

- Une valeur indéfinie passée en paramètre à une méthode projet est automatiquement convertie en 0 ou en "" en fonction de la déclaration du type du paramètre.

```
C_OBJECT($o)
mymethod($o.a) // passage d'un paramètre indéfini

// Dans la méthode mymethod
C_TEXT($1) // Paramètre de type texte
// $1 contient ""
```

- Une expression de condition est automatiquement convertie à Faux lorsque son évaluation donne Indéfinie avec les mots-clés Si et Au cas où :

```
C_OBJECT($o)
If($o.a) // faux
End if
Case of
    :($o.a) // faux
End case
```

- L'affectation d'une valeur indéfinie à une propriété d'objet existante réinitialise ou efface sa valeur, selon son type :
- Objet, collection, pointeur : Null
- Image : image vide
- Booléen : False
- Chaîne : ""
- Numérique : 0
- Date : !00-00-00! si la base utilise le type date pour les objets, sinon ""
- Heure : 0 (nombre de ms)
- Indéfini, Null : pas de changement

```
C_OBJECT($o)
$o:=New object("a";2)
$o.a:=$o.b // $o.a=0
```

- L'affectation d'une valeur indéfinie à une propriété d'objet inexistante ne fait rien.

Lorsque des expressions d'un type donné sont attendues dans votre code 4D, vous pouvez vous assurer qu'elles auront le type souhaité même en cas de valeur Indéfinie en les encadrant avec la commande de transtypage 4D appropriée : `String`, `Num`, `Time`, `Date`, `Bool`. Ces commandes retournent une valeur vide du type spécifié lorsque l'expression est évaluée à Indéfinie. Par exemple :

```
$myString:=Lowercase(String($o.a.b)) //pour être sûr d'obtenir une valeur texte même si indéfinie  
//afin d'éviter des erreurs dans le code
```

Exemples

L'utilisation de la notation objet simplifie grandement le code 4D de manipulation des objets. A noter toutefois que la notation utilisant les commandes "OB" reste entièrement prise en charge.

- Ecriture et lecture de propriétés d'objets (cet exemple compare la notation objet et la syntaxe avec commandes) :

```
// Utilisation de la notation objet  
C_OBJECT($myObj) //déclaration d'une variable objet 4D  
$myObj:=New object //création d'un objet et affectation à la variable  
$myObj.age:=56  
$age:=$myObj.age //56  
  
// Utilisation de la syntaxe par commande  
C_OBJECT($myObj2) //déclaration d'une variable objet 4D  
OB SET($myObj2;"age";42) //création d'un objet et création de la propriété age  
$age:=OB Get($myObj2;"age") //42  
  
// Bien entendu, les deux notations peuvent être utilisées simultanément  
C_OBJECT($myObj3)  
OB SET($myObj3;"age";10)  
$age:=$myObj3.age //10
```

- Création de propriétés et affectation de valeurs, y compris d'autres objets :

```
C_OBJECT($Emp)  
$Emp:=New object  
$Emp.city:="London" //crée la propriété city avec la valeur "London"  
$Emp.city:="Paris" //modifie la propriété city  
$Emp.phone:=New object("office";"123456789";"home";"0011223344")  
//crée la propriété phone avec un autre objet comme valeur
```

- Lire une valeur dans un sous-objet est très simple avec la notation objet :

```
$vCity:=$Emp.city //"Paris"  
$vPhone:=$Emp.phone.home //"0011223344"
```

- Vous pouvez accéder aux propriétés d'objets via des chaînes grâce à l'opérateur []

```
$Emp["city"]:="Berlin" //modification de la propriété city  
//cette syntaxe est utile pour créer des propriétés à l'aide de variables  
C_TEXT($addr)  
$addr:="address"  
For($i;1;4)  
    $Emp[$addr+Chaine($i)]:=""  
End for  
// crée 4 propriétés vides "address1...address4" dans l'objet $Emp
```

Image

Un champ, une variable ou expression de type image peut constituer une image Windows ou Macintosh. En règle générale, n'importe quelle image peut être mise sur le conteneur de données ou lue à partir du disque, à l'aide des commandes 4D telles que `READ PICTURE FILE`.

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

- sous Windows, 4D utilise WIC (Windows Imaging Component).
- sous macOS, 4D utilise ImageIO.

WIC et ImageIO permettent l'utilisation de métadonnées dans les images. Deux commandes, `SET PICTURE METADATA` et `GET PICTURE METADATA`, vous permettent d'en bénéficier dans vos développements.

Identifiants de codecs d'images

4D prend en charge de façon native un large ensemble de [formats d'images](#), tels que .jpeg, .png, ou .svg.

Les formats d'images reconnus par 4D sont retournés par la commande `PICTURE CODEC LIST` sous forme d'identifiants de codecs d'images. Ces identifiants peuvent être :

- une extension (par exemple ".gif")
- Un type Mime (par exemple "image/jpg")

La forme utilisée pour chaque format dépend du mode de déclaration du codec au niveau du système d'exploitation. Notez que les listes de codecs disponibles pour la lecture et pour l'écriture peuvent différer, étant donné que les codecs d'encodage peuvent nécessiter des licences spécifiques.

La plupart des [commandes 4D de gestion d'images](#) peuvent recevoir un Codec ID en paramètre. Il est donc impératif d'utiliser l'identifiant système retourné par la commande `PICTURE CODEC LIST`. Les formats d'images reconnus par 4D sont retournés par la commande `PICTURE CODEC LIST`.

Opérateurs sur les images

Opération	Syntaxe	Retourne	Action	
Concaténation horizontale	Image1 + Image2	Image	Place Image2 à la droite d'Image1	
Concaténation verticale	Image1 / Image2	Image	Place Image2 au-dessous d'Image1	
Superposition exclusive	Image1 & Image2	Image	Superpose Image2 à Image1 (Image2 est au premier plan). Donne le même résultat que COMBINE PICTURES(pict3;pict1;Superposition;pict2)	
Superposition inclusive	Image1	Image	Image	Superpose Image2 et retourne le masque résultant si les deux sont de même taille même résultat que \$equal:=Equal pictures(Pict1;Pic2)
Déplacement horizontal	Image + Nombre	Image	Déplace l'image horizontalement d'un nombre de pixels égal à Nombre	
Déplacement vertical	Image / Nombre	Image	Déplace l'image verticalement d'un nombre de pixels égal à Nombre	
Redimensionnement	Image * Nombre	Image	Redimensionne l'image au pourcentage Nombre	
Extension horizontale	Image *+ Nombre	Image	Redimensionne l'image horizontalement au pourcentage Nombre	
Extension verticale	Image *	Image	Image	Redimensionne l'image verticalement au pc Nombre

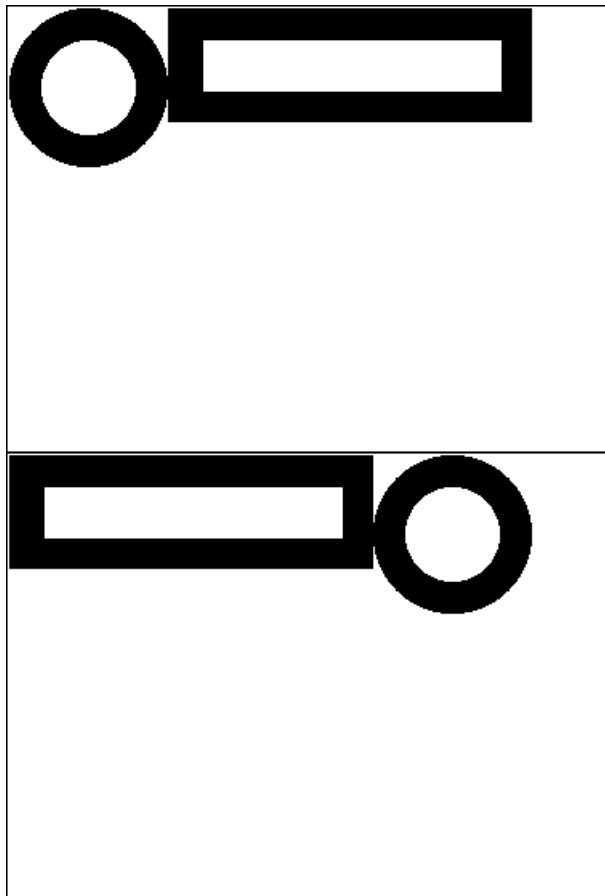
Notes :

- A noter que pour que l'opérateur | puisse être utilisé, Image1 et Image2 doivent être strictement de la même dimension. Si les deux images sont de taille différente, l'opération Image1 | Image2 produit une image vide.
- La commande **COMBINE PICTURES** permet d'effectuer des superpositions en conservant les caractéristiques de chaque image source dans l'image résultante.
- Des opérations supplémentaires peuvent être réalisées sur des images à l'aide de la commande **TRANSFORM PICTURE**.
- Il n'existe pas d'opérateurs de comparaison pour les images; en revanche 4D propose d'utiliser la commande **Images égales** pour comparer deux images.

Exemples

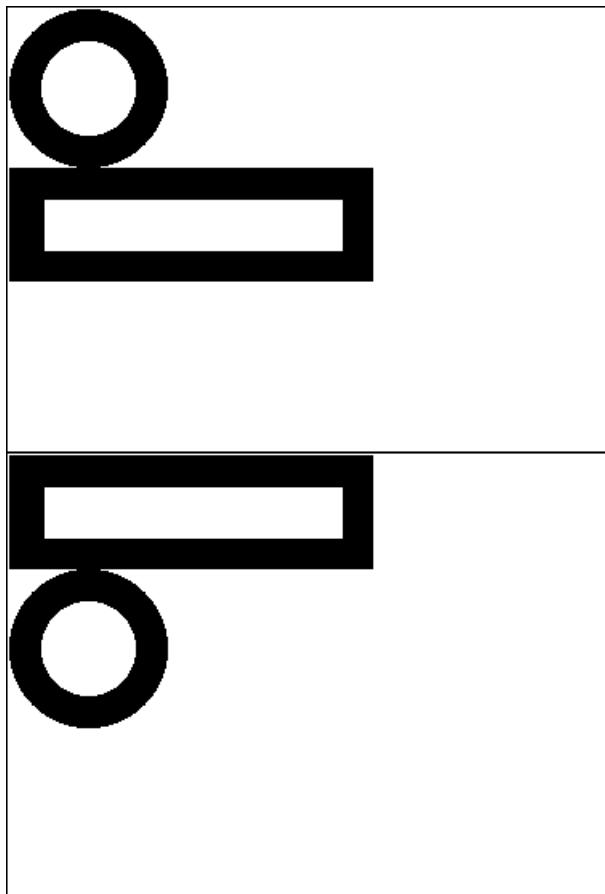
Concaténation horizontale

```
cercle+rectangle // Place le rectangle à droite du cercle
rectangle+cercle // Place le cercle à droite du rectangle
```



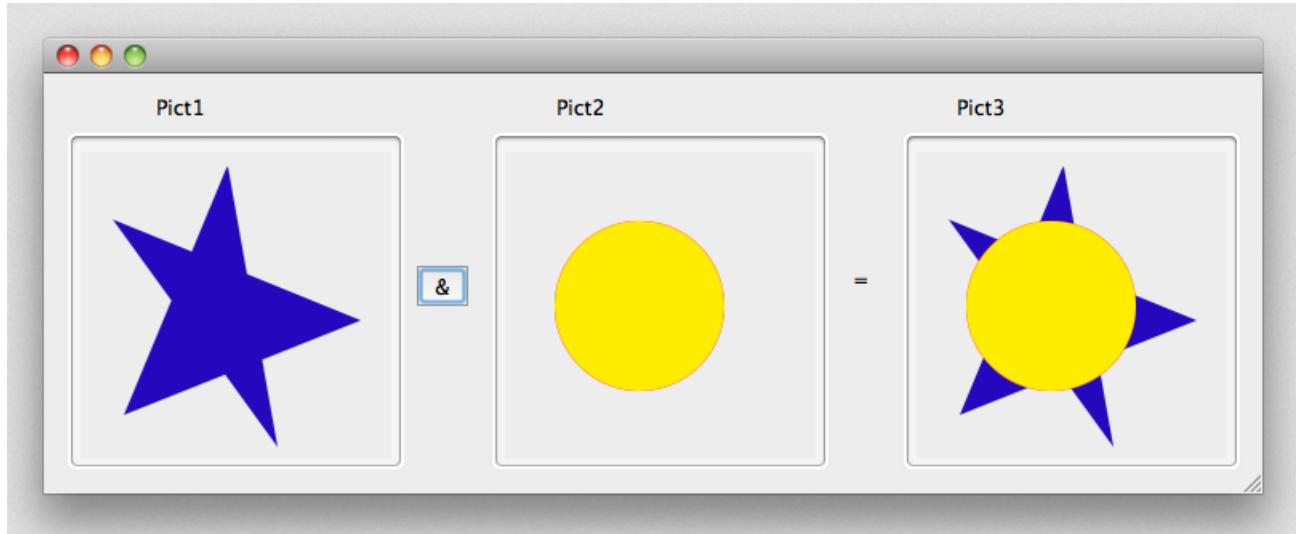
Concaténation verticale

```
circle/rectangle //Place the rectangle under the circle  
rectangle/circle //Place the circle under the rectangle
```



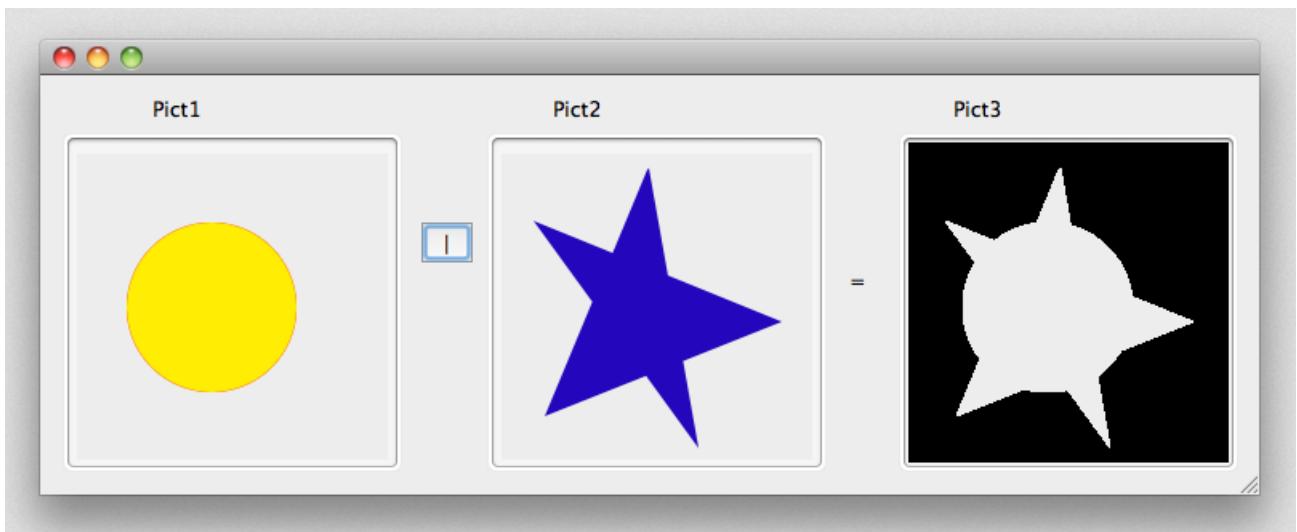
Superposition exclusive

```
Pict3:=Pict1 & Pict2 // Superposer Pict2 à Pict1
```



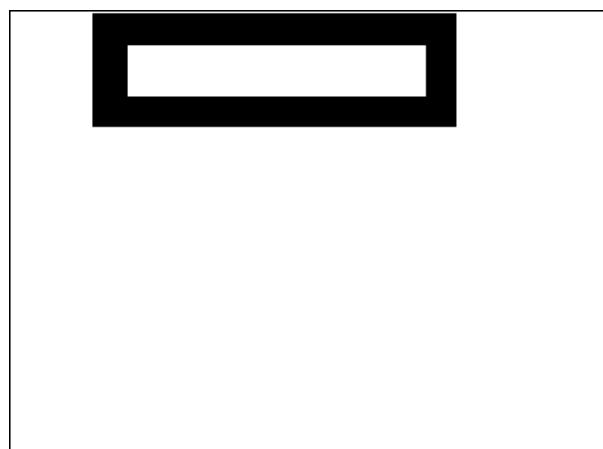
Superposition inclusive

```
Pict3:=Pict1|Pict2 // Récupérer le masque résultant de la superposition de deux images de même taille
```



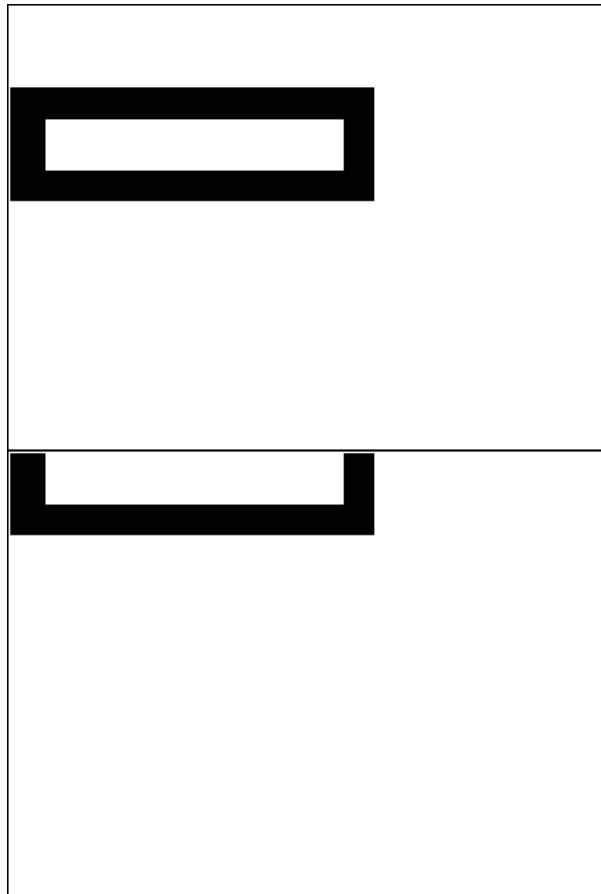
Déplacement horizontal

```
rectangle+50 // Déplace le rectangle 50 pixels vers la droite  
rectangle-50 // Déplace le rectangle 50 pixels vers la gauche
```



Déplacement vertical

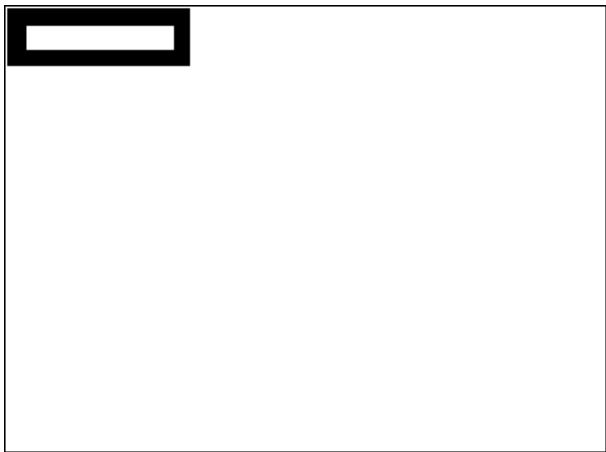
```
rectangle/50 // Déplace le rectangle 50 pixels vers le bas  
rectangle/-20 // Déplace le rectangle 20 pixels vers le haut
```



Redimensionnement

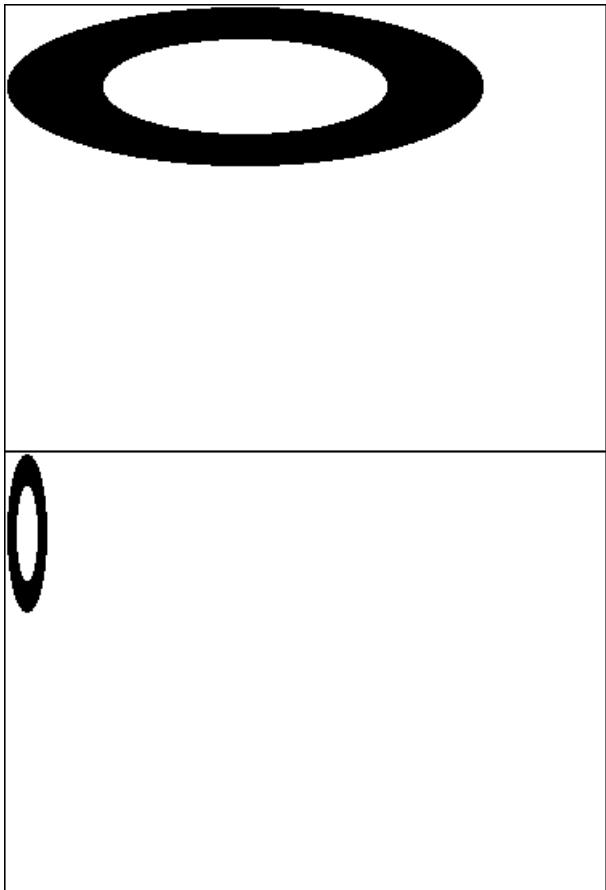
```
rectangle*1.5 // Augmente la taille du rectangle de 50%  
rectangle*0.5 // Réduit la taille du rectangle de 50%
```





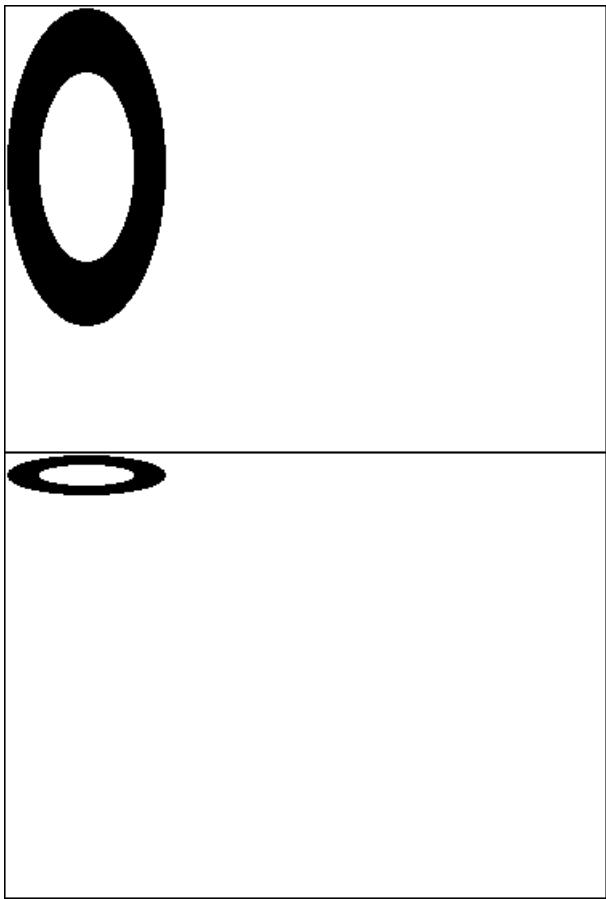
Extension horizontale

```
cercle*3 // Multiplie par 3 la largeur du cercle  
cercle*0,25 // La largeur du cercle est réduite à un quart de sa taille originale
```



Extension verticale

```
cercle*2 // Double la hauteur du cercle  
cercle*0,25 // La hauteur du cercle est réduite à un quart de sa taille originale
```



Pointeur

Les variables ou expressions de type Pointeur sont des références à d'autres variables (y compris des tableaux et des éléments de tableaux), à des tables, des champs ou des objets. Il n'existe pas de champ de type Pointeur.

Les pointeurs sont des outils de programmation avancée. Lorsque vous utilisez le langage de 4D, vous vous référez aux différents objets par l'intermédiaire de leur nom — en particulier les tables, champs, variables et tableaux. Pour appeler l'un d'entre eux, vous écrivez simplement son nom. Cependant, il est parfois utile de pouvoir appeler ou référencer ces éléments sans nécessairement connaître leur nom. C'est ce que permettent les pointeurs.

Le concept de pointeur n'est pas tellement éloigné de la vie courante. Vous vous référez souvent à des choses sans connaître leur identité exacte. Par exemple, vous pourriez dire à un ami : "Allons faire un tour avec ta voiture" au lieu de "Allons faire un tour avec la voiture qui a la plaque d'immatriculation 123ABD". Dans ce cas, vous faites référence à la voiture avec la plaque d'immatriculation 123ABD en utilisant l'expression "ta voiture". L'expression "la voiture qui a la plaque d'immatriculation 123ABD" est le nom d'un objet, et l'expression "ta voiture" revient à utiliser un pointeur pour référencer l'objet.

La capacité de se référer à quelque chose sans connaître son identité exacte est très utile. Si votre ami s'achetait une nouvelle voiture, l'expression "ta voiture" serait toujours exacte — ce serait toujours une voiture et vous pourriez toujours aller quelque part avec. Les pointeurs fonctionnent de la même manière. Par exemple, un pointeur peut pointer à un moment donné vers un champ numérique appelé Age, et plus tard vers une variable numérique appelée Ancien âge. Dans les deux cas, le pointeur référence des données numériques pouvant être utilisée dans des calculs.

Vous pouvez utiliser des pointeurs pour référencer des tables, des champs, des variables, des tableaux et des éléments de tableaux. Le tableau suivant vous fournit un exemple de chaque type :

Type	Référencement	Référencement	Affectation
Table	vpTable:=->[Table]	DEFAULT TABLE(vpTable->)	n/a
Champ	vpField:=->[Table]Field	ALERT(vpField->)	vpField->:="John"
Variable	vpVar:=->Variable	ALERT(vpVar->)	vpVar->:="John"
Tableau	vpArr:=->Array	SORT ARRAY(vpArr->;>)	COPY ARRAY (Arr;vpArr->)
Elém. tabl.	vpElem:=->Array{1}	ALERT (vpElem->)	vpElem->:="John"
Object	vpObj:=->myObject	ALERT (vpObj->myProp)	vpObj->myProp:="John"

Utiliser des pointeurs : un exemple

Il est plus facile d'expliquer l'utilisation des pointeurs au travers d'un exemple. Cet exemple vous montre comment accéder à une variable par l'intermédiaire d'un pointeur. Nous commençons par créer la variable :

```
$MyVar:="Hello"
```

\$MyVar est désormais une variable contenant la chaîne "Hello". Nous pouvons alors créer un pointeur vers \$MyVar :

```
C_POINTER($MyPointer)  
$MyPointer:=->$MyVar
```

Le symbole -> signifie "pointer vers". Ce symbole est formé du caractère "tiret" (-) suivi du caractère "supérieur à". Dans ce cas, il crée un pointeur qui référence ou "pointe vers" \$MyVar. Ce pointeur est assigné à \$MyPointer via l'opérateur d'assignation.

\$MyPointer est désormais une variable qui contient un pointeur vers \$MyVar. \$MyPointer ne contient pas "Hello", la valeur de \$MyVar, mais vous pouvez utiliser \$MyPointer pour obtenir cette valeur. L'expression suivante retourne la valeur de \$MyVar :

```
$MyPointer->
```

Dans ce cas, la chaîne "Hello" est retournée. Lorsque le symbole `->` est placé derrière un pointeur, la valeur de l'objet vers lequel pointe le pointeur est récupérée. On dit alors qu'on dépointe le pointeur.

Il est important de comprendre que vous pouvez utiliser un pointeur suivi du symbole `->` partout où vous auriez pu utiliser l'objet pointé lui-même. Vous pouvez placer l'expression `$MyPointer->` partout où vous pourriez utiliser la variable originale `$MyVar`. Par exemple, l'instruction suivante affiche une boîte de dialogue d'alerte comportant le mot Hello :

```
ALERT($MyPointer->)
```

Vous pouvez également utiliser `$MyPointer` pour modifier la valeur de `$MyVar`. Par exemple, l'instruction suivante stocke la chaîne "Goodbye" dans la variable `$MyVar` :

```
$MyPointer->:="Goodbye"
```

Si vous examinez les deux utilisations de l'expression `$MyPointer->` ci-dessus, vous constatez que cette expression se comporte exactement comme si vous aviez utilisé `$MyVar` à sa place. En résumé : les deux lignes suivantes effectuent la même opération — elles affichent une boîte de dialogue d'alerte contenant la valeur courante de la variable `$MyVar` :

```
ALERT($MyPointer->)
ALERT($MyVar)
```

Les deux lignes suivantes effectuent la même opération ; elles assignent la chaîne "Goodbye" à `$MyVar` :

```
$MyPointer->:="Goodbye"
$MyVar:="Goodbye"
```

Opérateurs sur les pointeurs

Avec :

```
// vPtrA et vPtrB pointent sur le même objet
vPtrA:=>unObjet
vPtrB:=>unObjet
// vPtrC pointe sur un autre objet
vPtrC:=>autreObjet
```

Opération	Syntaxe	Retourne	Expression	Valeur
Egalité	Pointeur = Pointeur	Booléen	vPtrA = vPtrB	Vrai
			vPtrA = vPtrB	Faux
Inégalité	Pointeur # Pointeur	Booléen	vPtrA # vPtrC	Vrai
			vPtrA # vPtrB	Faux

Principales utilisations

Utiliser des pointeurs vers des tables

Partout où le langage requiert un nom de table, vous pouvez utiliser un pointeur dépointé vers une table. Pour créer un pointeur vers une table, écrivez une instruction du type :

```
$TablePtr:==>[touteTable]
```

Vous pouvez également récupérer un pointeur vers une table à l'aide de la commande `Table`. Par exemple :

```
$TablePtr:=Table(20)
```

Vous pouvez utiliser le pointeur dépointé dans vos commandes, comme ceci :

```
DEFAULT TABLE($TablePtr->)
```

Utiliser des pointeurs vers des champs

Partout où le langage requiert un nom de champ, vous pouvez utiliser un pointeur dépointé vers un champ. Pour créer un pointeur vers un champ, écrivez une ligne d'instruction du type :

```
$ChampPtr:==>[uneTable]CeChamp
```

Vous pouvez également récupérer un pointeur vers un champ à l'aide de la fonction `Champ`. Par exemple :

```
$FieldPtr:=Field(1;2)
```

Vous pouvez utiliser le pointeur dépointé dans vos commandes, comme ceci :

```
OBJECT SET FONT($FieldPtr->;"Arial")
```

Utiliser des pointeurs vers des variables

Lorsque vous utilisez des pointeurs vers des variables locales ou des variables process, vous devez veiller à ce que la variable pointée soit bien définie au moment de l'utilisation du pointeur. Rappelons que les variables locales sont supprimées à la fin de l'exécution de la méthode qui les a créées et les variables process à la fin du process dans lequel elles ont été créées. L'appel d'un pointeur vers une variable qui n'existe plus provoque une erreur de syntaxe en mode interprété (variable indéfinie) mais peut générer une erreur plus conséquente en mode compilé.

Les pointeurs vers des variables locales permettent dans de nombreux cas d'économiser des variables process. Les pointeurs vers des variables locales peuvent être utilisés uniquement à l'intérieur d'un même process. Dans le débogueur, lorsque vous affichez un pointeur vers une variable locale déclarée dans une autre méthode, le nom de la méthode d'origine est indiquée entre parenthèses, derrière le pointeur. Par exemple, si vous écrivez dans Méthode1 :

```
$MyVar:="Hello world"  
Method2(~>$MyVar)
```

Dans Method2, le débogueur affichera \$1 de la façon suivante :

\$1	->\$MyVar (Method1)
-----	---------------------

La valeur de \$1 sera :

\$MyVar (Method1)	"Hello world"
-------------------	---------------

Utiliser des pointeurs vers des éléments de tableau

Vous pouvez créer un pointeur vers un élément de tableau. Par exemple, les lignes d'instruction suivantes créent un tableau et assignent à une variable appelée \$ElémPtr un pointeur vers le premier élément :

```
ARRAY REAL($unTableau;10) // Créer un tableau  
$ElémPtr:==>$unTableau{1} // Créer un pointeur vers l'élément de tableau
```

Vous pouvez alors utiliser le pointeur dépointé pour assigner une valeur à l'élément, comme ceci :

```
$ElémPtr->:=8
```

Utiliser des pointeurs vers des tableaux

Vous pouvez créer un pointeur vers un tableau. Par exemple, les lignes d'instruction suivantes créent un tableau et assignent à la variable nommée \$TabPtr un pointeur vers le tableau :

```
ARRAY REAL($unTableau;10) // Créer un tableau  
$TabPtr:==>$unTableau // Créer un pointeur vers le tableau
```

Il est important de comprendre que ce pointeur pointe vers le tableau, et non vers un élément du tableau. Par exemple, vous pourriez utiliser le pointeur dépointé de la manière suivante :

```
SORT ARRAY($TabPtr->:>) // Tri du tableau
```

Si vous devez vous référer au quatrième élément du tableau à l'aide du pointeur, vous pouvez écrire :

```
$TabPtr->{4}:=84
```

Passer des pointeurs aux méthodes

Vous pouvez passer un pointeur en tant que paramètre d'une méthode. A l'intérieur de la méthode, vous pouvez modifier l'objet référencé par le pointeur. Par exemple, la méthode suivante, `takeTwo`, reçoit deux paramètres qui sont des pointeurs. Elle passe l'objet référencé par le premier paramètre en caractères majuscules, et l'objet référencé par le second paramètre en caractères minuscules.

```
// Méthode projet takeTwo  
// $1 – Pointeur vers un champ ou une variable de type Chaîne. Passe la chaîne en majuscules.  
// $2 – Pointeur vers un champ ou une variable de type Chaîne. Passe la chaîne en minuscules.  
$1->:=Uppercase($1->)  
$2->:=Lowercase($2->)
```

L'instruction suivante emploie la méthode `takeTwo` pour passer un champ en caractères majuscules et une variable en caractères minuscules :

```
takeTwo(>[MaTable]MonChamp;>:$MaVar)
```

Si le champ, [MaTable]MonChamp, contenait la chaîne "dupont", celle-ci deviendrait "DUPONT". Si la variable \$MaVar contenait la chaîne "BONJOUR", celle-ci deviendrait "bonjour".

Dans la méthode `takeTwo` (et, en fait, à chaque fois que vous utilisez des pointeurs), il est important que les types de données des objets référencés soient corrects. Dans l'exemple précédent, les pointeurs doivent pointer vers des objets

contenant une chaîne ou un texte.

Pointeurs vers des pointeurs

Si vous aimez compliquer les choses à l'extrême (bien que cela ne soit pas nécessaire dans 4D), vous pouvez utiliser des pointeurs pour référencer d'autres pointeurs. Examinons l'exemple suivant :

```
$MyVar:="Hello"  
$PointerOne:-->$MyVar  
$PointerTwo:-->$PointerOne  
($PointerTwo->)->:="Goodbye"  
ALERT(($PointerTwo->)->)
```

Cet exemple affiche une boîte de dialogue d'alerte contenant "Goodbye".

Voici la description de chaque ligne de l'exemple :

- \$MyVar:="Hello" --> Cette ligne place simplement la chaîne "Hello" dans la variable \$MyVar.
- \$PointerOne:-->\$MyVar --> \$PointerOne contient désormais un pointeur vers \$MyVar.
- \$PointerTwo:-->\$PointerOne --> \$PointerTwo (une nouvelle variable) contient un pointeur vers \$PointerOne, qui, lui, pointe vers \$MyVar.
- (\$PointerTwo->)->:="Goodbye" --> \$PointerTwo-> référence le contenu de \$PointerOne, qui lui-même référence \$MyVar. Par conséquent, (\$PointerTwo->)-> référence le contenu de \$MyVar. Donc, dans ce cas, la valeur "Goodbye" est assignée à \$MyVar.
- ALERT ((\$PointerTwo->)->) --> C'est ici la même chose que précédemment : \$PointerTwo-> référence le contenu de \$PointerOne, qui lui-même référence \$MyVar. Par conséquent, (\$PointerTwo->)-> référence le contenu de \$MyVar. Donc, dans ce cas, la boîte de dialogue d'alerte affiche le contenu de \$MyVar.

La ligne suivante place la valeur "Hello" dans \$MyVar :

```
($PointerTwo->)->:="Hello"
```

La ligne suivante récupère "Hello" à partir de \$MyVar et la place dans \$NewVar :

```
$NewVar:=($PointerTwo->)->
```

Important : Vous devez utiliser des parenthèses lors des déréférencements multiples.

Chaîne

Chaîne est un terme générique utilisé pour :

- Les variables ou champs de type Texte : un champ, une variable ou une expression de type Texte peut contenir de 0 à 2 Go de texte.
- Les variables ou champs de type alphanumérique : un champ alphanumérique peut contenir de 0 à 255 caractères (la limite est fixée lors de la définition du champ).

Constantes littérales de type chaîne

Une constante littérale de type chaîne est incluse entre des guillemets droits ("…"). En voici quelques exemples :

```
"Ajouter Enregistrements"  
"Aucun enregistrement trouvé."  
"Facture"
```

Une chaîne vide est spécifiée par la succession de deux guillemets ("").

Séquences d'échappement

Les séquences d'échappement suivantes peuvent être utilisées dans les chaînes :

Séquence d'échappement	Caractère remplacé
¥n	LF (Retour ligne)
¥t	HT (Tabulation)
¥r	CR (Retour chariot)
¥¥	¥ (Barre oblique inversée)
¥"	" (Guillemets)

Note: Le caractère ¥ est utilisé comme séparateur dans les chemins d'accès sous Windows. Vous devez donc saisir un double ¥ lorsque vous souhaitez insérer une barre oblique inversée devant un caractère utilisé dans une des séquences d'échappement reconnues par 4D (ex : "C:¥MesDocuments¥Nouveaux.txt").

Opérateurs sur les chaînes

Opération	Syntaxe	Retourne	Expression	Valeur
Concaténation	Chaîne + Chaîne	Chaîne	"abc" + "def"	"abcdef"
Répétition	Chaîne * Nombre	Chaîne	"ab" * 3	"ababab"
Égalité	Chaîne = Chaîne	Booléen	"abc" = "abc"	Vrai
			"abc" = "abd"	Faux
Inégalité	Chaîne # Chaîne	Booléen	"abc" # "abd"	Vrai
			"abc" # "abc"	Faux
Supérieur à	Chaîne > Chaîne	Booléen	"abd" > "abc"	Vrai
			"abc" > "abc"	Faux
Inférieur à	Chaîne < Chaîne	Booléen	"abc" < "abd"	Vrai
			"abc" < "abc"	Faux
Supérieur ou égal à	Chaîne >= Chaîne	Booléen	"abd" >= "abc"	Vrai
			"abc" >= "abd"	Faux
Inférieur ou égal à	Chaîne <= Chaîne	Booléen	"abc" <= "abd"	Vrai
			"abd" <= "abc"	Faux
Contient mot-clé	Chaîne % Chaîne	Booléen	"Alpha Bravo" % "Bravo"	Vrai
			"Alpha Bravo" % "ravo"	Faux
	Image % Chaîne	Booléen	Expr_image % "Mer"	True (*)

(*) Si le mot-clé "Mer" a été associé à l'image stockée dans l'expression image (champ ou variable).

Comparaisons de chaînes

- Les chaînes sont toujours comparées caractère par caractère (hormis en cas de recherche par [mot-clé](#), cf. ci-dessous).
- Lors d'une comparaison de chaînes, 4D ne tient pas compte de la casse des caractères ; par exemple, "a"="A" retourne **VRAI**. Pour savoir si des caractères sont en majuscules ou en minuscules, vous devez comparer leurs codes de caractères. Par exemple, l'expression suivante retourne **FAUX** :

```
Code de caractere("A")=Code de caractere("a") // 65 n'est pas égal à 97
```

- Lors d'une comparaison de chaînes, les caractères diacritiques sont comparés à l'aide de la table de comparaison des caractères de votre machine. Par exemple, les expressions suivantes retournent **VRAI** :

```
"ñ"="ñ"  
"ñ"="Ñ"  
"À"="â"  
// etc
```

Note : Les comparaisons de chaîne tiennent compte des spécificités du langage défini pour le fichier de données 4D (qui n'est pas toujours identique au langage défini pour le système).

Le joker (@)

Le langage 4D prend en charge @ en tant que joker. Ce caractère peut être utilisé dans toute comparaison de chaînes. Ainsi, par exemple, l'expression suivante est évaluée à **TRUE** :

```
"abcdefgij"="abc@"
```

Le joker doit être utilisé dans le second opérande (la chaîne qui se trouve à droite de l'opérateur). L'expression suivante est évaluée à FAUX car le joker est alors considéré en tant que caractère :

```
"abc@"="abcdefgij"
```

Le joker signifie "un ou plusieurs caractères sinon rien". Les expressions suivantes sont évaluées à VRAI :

```
"abcdefgij"="abcdefgij@"
"abcdefgij"="@abcdefgij"
"abcdefgij"="abcd@efghij"
"abcdefgij"="@abcdefgij@"
"abcdefgij"="@abcde@fghij@"
```

En revanche, dans tous les cas, lorsque deux jokers consécutifs sont placés dans une comparaison de chaînes, celle-ci sera évaluée à FAUX . L'expression suivante est à FAUX :

```
"abcdefgij"="abc@@fg"
```

Lorsque l'opérateur de comparaison est ou contient un symbole < ou >, seule la comparaison avec un seul joker situé en fin d'opérande est prise en charge :

```
"abcd"<="abc@"
"abcd"<="abc@ef" //Comparaison non valide
```

Si vous souhaitez effectuer des comparaisons ou des recherches utilisant @ en tant que caractère (et non en tant que joker), vous devez utiliser l'instruction Code de caractere(Arobase) . Imaginons par exemple que vous souhaitez savoir si une chaîne se termine par le caractère @. L'expression suivante (si \$vaValeur n'est pas vide) retourne toujours VRAI :

```
($vaValeur[[Longueur($vaValeur)]]=="@")
```

L'expression suivante sera correctement évaluée :

```
(Code de caractere($vaValeur[[Longueur($vaValeur)]]))#64)
```

Note : Une option 4D du mode Développement vous permet de paramétrier le mode d'interprétation du caractère @ lorsque celui-ci est inclus dans une chaîne de caractères.

Mots-clés

A la différence des autres comparaisons de chaîne, les recherches par mots-clés recherchent des "mots" dans des "textes" : les mots sont évalués individuellement et dans leur globalité. L'opérateur % retournera toujours Faux si la recherche porte sur plusieurs mots ou une partie de mot (par exemple une syllabe). Les "mots" sont des chaînes de caractères encadrées par des " séparateurs ", qui sont les espaces, les caractères de ponctuation et les tirets. Une apostrophe, comme dans "aujourd'hui", est généralement considérée comme partie du mot, mais sera ignorée dans certains cas (cf. règles ci-dessous). Les nombres peuvent être recherchés car ils sont évalués dans leur ensemble (incluant les symboles décimaux). Les autres symboles (monnaie, température, etc.) seront ignorés.

```

"Alpha Bravo Charlie%" "Bravo" // Retourne Vrai
"Alpha Bravo Charlie%" "vo" // Retourne Faux
"Alpha Bravo Charlie%" "Alpha Bravo" // Retourne Faux
"Alpha,Bravo,Charlie%" "Alpha" // Retourne Vrai
"Software and Computers%" "comput@" // Retourne Vrai

```

Notes : - 4D utilise la librairie ICU pour la comparaison des chaînes (à l'aide des opérateurs <>=#) et la détection des mots-clés. Pour plus d'informations sur les règles mises en oeuvre, reportez-vous à l'adresse http://www.unicode.org/reports/tr29/#Word_Boundaries. En version japonaise, 4D utilise par défaut la librairie Mecab en lieu et place de ICU pour la détection des mots-clés.

Symboles d'indice de chaîne

Les symboles d'indice de chaîne sont les suivants : [[...]]

Ces symboles sont utilisés pour désigner un caractère particulier dans une chaîne. Cette syntaxe vous permet de référencer un caractère dans un champ ou une variable de type Alpha ou Texte.

Lorsque les symboles d'indice de chaîne sont placés à gauche de l'opérateur d'affectation (:=), un caractère est affecté à la position référencée dans la chaîne. Par exemple, en postulant que la chaîne vsNom n'est pas une chaîne vide, le code suivant passe le premier caractère de la chaîne vsNom en majuscule :

```

If(vsNom#"")

    vsNom[[1]] := Uppercase(vsNom[[1]])

End if

```

Lorsque les symboles d'indice de chaîne apparaissent dans une expression, ils retournent le caractère auquel ils font référence sous la forme d'une chaîne d'un caractère. Par exemple :

```

//L'exemple suivant teste si le dernier caractère de vtText est le caractère "@"
If(vtText#"")

    If(Character code(Substring(vtText;Length(vtText);1))=At sign)
    //...
    End if
End if

//En utilisant la syntaxe des caractères d'indice de chaîne, vous écririez plus simplement :
If(vtText#"")

    If(Character code(vtText[[Length(vtText)]])=At sign)
    // ...
    End if
End if

```

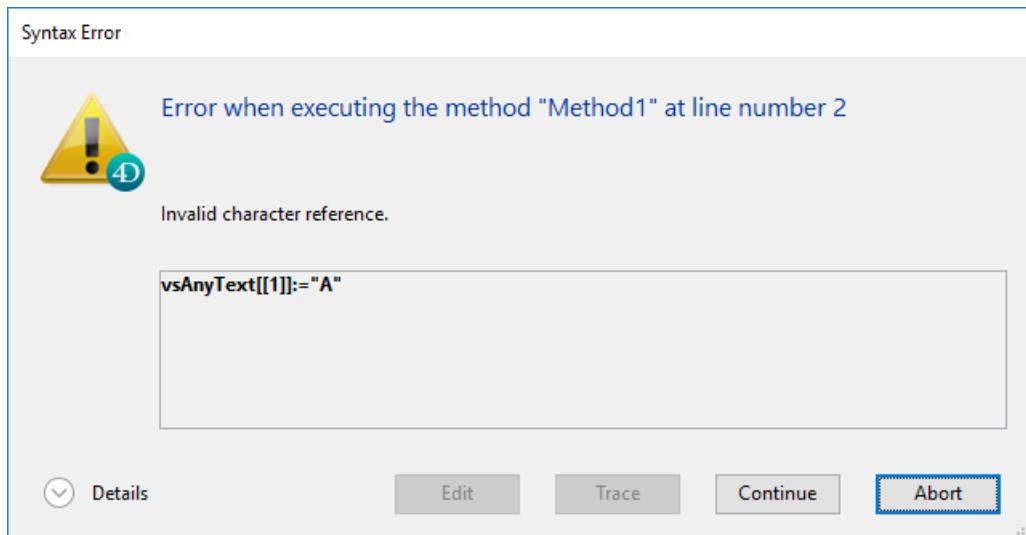
Note avancée sur la référence à des caractères invalides

Lorsque vous utilisez les symboles d'indice de chaîne, il est de votre responsabilité de vous référer à des caractères existant dans la chaîne, de la même manière que pour les éléments d'un tableau. Si, par exemple, vous référez le 20e caractère d'une chaîne, cette chaîne doit contenir au moins 20 caractères.

- Ne pas respecter cette condition en mode interprété n'est pas signalé comme une erreur par 4D.
- Ne pas respecter cette condition en mode compilé (sans options) peut entraîner une "corruption" de la mémoire, si, par exemple, vous écrivez un caractère au-delà de la fin d'une chaîne ou d'un texte.
- Ne pas respecter cette condition en mode compilé est signalé lorsque le contrôle d'exécution est activé. Si, par exemple, vous exécutez le code suivant :

```
//Ne pas faire ça !
vsAnyText:=""
vsAnyText[[1]]:="A"
```

L'alerte suivante s'affichera en mode compilé :



Exemple

La méthode projet suivante ajoute une lettre capitale à tous les mots du texte passé en paramètre et retourne le texte modifié :

```
// Méthode projet de passage en capitale
// PasserEnCap ( Texte ) -> Texte
// PasserEnCap ( Texte source ) -> Texte avec des lettres capitales

$0:=$1
$vlLen:=Length($0)
If($vlLen>0)
    $0[[1]]:=Uppercase($0[[1]])
    For($vlChar;1;$vlLen-1)
        If(Position($0[[vlChar]]; " !&()-{};:<>?/,.=+*")>0)
            $0[[vlChar+1]]:=Uppercase($0[[vlChar+1]])
        End if
    End for
End if
```

Une fois cette méthode placée dans la base, la ligne :

```
ALERT(Capitalize_text("Bonjour, mon nom est Jean Bon et je me présente aux présidentielles !"))
```

affiche l'alerte suivante :

Alert



Hello, My Name Is Jane Doe And I'm Running For President!

OK

Heure

Les variables, champs ou expressions de type Heure peuvent être compris entre 00:00:00 et 596 000:00:00.

Les heures sont stockées dans un format de 24 heures.

Une valeur de type Heure peut être utilisée en tant que numérique. Le nombre correspondant est le nombre de secondes que cette valeur représente à partir de minuit (00:00:00).

Note : Dans le manuel de *référence du langage 4D*, les paramètres de type Heure dans les descriptions des commandes sont appelés Heure, sauf spécification explicite.

Constantes littérales de type heure

Une constante heure est incluse entre deux points d'interrogation (?...?).

Avec une version française de 4D, une heure est structurée sous la forme heure:minute:seconde, deux points (:) séparant les valeurs. Les heures sont stockées dans un format de 24 heures.

Voici quelques exemples de constantes littérales de type heure :

```
?00:00:00? // minuit  
?09:30:00? // 9:30 du matin  
?13:01:59? // 13 heures, 1 minute et 59 secondes
```

Une heure nulle s'écrit ?00:00:00?

Astuce : L'éditeur de méthodes dispose d'un raccourci pour saisir une heure nulle. Pour cela, tapez un point d'interrogation (?) et appuyez sur la touche Entrée.

Opérateurs sur les heures

Opération	Syntaxe	Retourne	Expression	Valeur
Addition	Heure + Heure	Heure	?02:03:04? + ?01:02:03?	?03:05:07?
Soustraction	Heure - Heure	Heure	?02:03:04? ?02:03:04?	?01:01:01?
Addition	Heure + Nombre	Nombre	?02:03:04? ?02:03:04?	7449
Soustraction	Heure - Nombre	Nombre	?02:03:04? ?02:03:04?	7319
Multiplication	Heure * Nombre	Nombre	?02:03:04? ?02:03:04?	14768
Division	Heure / Nombre	Nombre	?02:03:04? ?02:03:04?	3692
Division entière	Heure ¥ Nombre	Nombre	?02:03:04? ?02:03:04?	3692
Modulo	Heure % Heure	Heure	?20:10:00? % ?04:20:00?	?02:50:00?
Modulo	Heure % Nombre	Nombre	?02:03:04? ?02:03:04?	0
Egalité	Heure = Heure	Booléen	?01:02:03? >=?01:02:03?	Vrai
			?01:02:03? ?01:02:03?	Faux
Inégalité	Heure # Heure	Booléen	?01:02:03? ?01:02:03?	Vrai
			?01:02:03? ?01:02:03?	Faux
Supérieur à	Heure > Heure	Booléen	?01:02:03? <=?01:02:03?	Vrai
			?01:02:03? <=?01:02:03?	Faux
Inférieur à	Heure < Heure	Booléen	?01:02:03? ?01:02:03?	Vrai
			?01:02:03? ?01:02:03?	Faux
Supérieur ou égal à	Heure >= Heure	Booléen	?01:02:03? >=?01:02:03?	Vrai
			?01:02:03? >=?01:02:04?	Faux
Inférieur ou égal à	Heure <= Heure	Booléen	?01:02:03? <=?01:02:03?	Vrai
			?01:02:03? <=?01:02:03?	Faux

Exemple 1

Vous pouvez combiner des expressions de type heure et de type numérique à l'aide des fonctions `Time` et `Time string`.

Vous pouvez combiner des expressions Time et Number à l'aide des fonctions `Time` ou `Current time`:

```
// La ligne suivante assigne à la variable $vlSecondes le nombre de secondes qui, dans une heure à p
$vlSeconds:=Current time+3600
// La ligne suivante assigne à la variable $vhBientôt l'heure qu'il sera dans une heure
$vhSoon:=Time(Current time+3600)
```

La seconde ligne peut également être écrite de la façon suivante :

```
// La ligne suivante assigne à la variable $vhBientôt l'heure qu'il sera dans une heure
$vhSoon:=Current time+?01:00:00?
```

Exemple 2

L'opérateur Modulo permet notamment d'ajouter des heures en tenant compte du format sur 24 heures d'une journée :

```
$t1:=?23:00:00? // Il est 23 heures  
// Nous souhaitons ajouter 2 heures et demi  
$t2:=$t1 +?02:30:00? // avec une addition simple, $t2 vaut ?25:30:00?  
$t2:=($t1 +?02:30:00?)%?24:00:00? // $t2 vaut ?01:30:00? , il est bien 1h30 du matin le lendemain
```

Variant

Variant est un type de variable qui permet d'encapsuler des données de type valide et standard dans une variable. En règle générale, ce type de variable peut être utilisé pour écrire du code générique qui retourne ou reçoit des valeurs dont le type n'est pas connu. C'est le cas par exemple du code traitant des attributs d'objet.

Une variable de type variant peut contenir une valeur des types de données suivants :

- BLOB
- boolean
- collection
- date
- entier long
- object
- picture
- pointer
- réel
- Texte
- time
- null
- indéfini

Les tableaux ne peuvent pas être stockés dans des variables de type variant.

En mode interprété et compilé, le même contenu peut être affecté à une même variable variant. Contrairement aux types de variable standard, le type de contenu des variable de type variant est différent du type de variable variant lui-même. Par exemple :

```
C_VARIANT($variant)

$variant:="hello world"
$type:=Type($variant) // 12 (Is variant)
$typeVal:=Value type($variant) // 2 (Is text)

$variant:=42
$type:=Type($variant) // 12 (Is variant)
$typeVal:=Value type($variant) // 1 (Is real)
```

Vous pouvez utiliser des variables variant chaque fois qu'elles sont attendues. Vous devez simplement vous assurer que le type de données du contenu de la variable est du type attendu. Lorsque vous accédez à des variables de type variant, seule leur valeur courante est prise en compte. Par exemple :

```
C_VARIANT($v)
$v:="hello world"
$v2:=$v //assign variable to another variable

$t:=Type($v) // 12 (Is variant)
$t2:=Type($v2) // 2 (Is text)
```

Variant peut être utilisé pour déclarer des paramètres de méthode (\$0, \$1, etc.) pouvant être de différents types. Dans ce cas, vous pouvez générer votre code en testant le type de valeur du paramètre, par exemple :

```
C_VARIANT($1)
Case of
: (Value type($1)=Is longint)
...
: (Value type($1)=Is text)
...
//déclaration(s)
End case
```

Lorsque des variables variant ne sont pas nécessaires (c'est-à-dire lorsque le type de données est connu), il est recommandé d'utiliser des variables typées standard. Les variables typées standard fournissent de meilleures performances, un code plus clair et permettent au compilateur d'éviter les bugs liés à des types de données passés qui sont inattendus.

Variables

Fondamentalement, dans 4D, les données peuvent être stockées de deux manières. Les champs stockent les données sur disque, de manière permanente ; les variables stockent les données en mémoire, de manière temporaire.

Lorsque vous définissez votre base, vous indiquez à 4D les noms et les types de champs que vous voulez utiliser. C'est pratiquement la même chose pour les variables — vous leur donnez un nom et un type (cf. [Type de données](#)).

Une fois créée, vous pouvez utiliser une variable partout dans votre application. Par exemple, vous pouvez stocker une variable de type texte dans un champ du même type :

```
[MaTable]MonChamp:=MonTexte
```

Les variables sont des objets du langage; vous pouvez créer et utiliser des variables qui n'apparaîtront jamais à l'écran. Dans vos formulaires, vous pouvez afficher des variables à l'écran (à l'exception des pointeurs et des BLOB), les utiliser pour saisir des données, et les imprimer dans des états. Dans ces cas, elles se comportent exactement comme des champs, et les mêmes contrôles intégrés sont disponibles lorsque vous les créez . Les variables peuvent également servir à contrôler des boutons, des list box, des zones de défilement, des boutons image, etc., ou à afficher les résultats de calculs ne devant pas être sauvegardés.

Déclaration des variables

Vous créez des variables en les déclarant. Le langage 4D propose deux manières de déclarer des variables :

- à l'aide du mot-clé `var` (recommandé particulièrement si votre code utilise des objets et des classes),
- à l'aide de l'une des commandes du langage 4D du thème "Compilateur" ou "Tableaux" (langage classique uniquement).

Note : Bien que cela ne soit généralement pas recommandé, vous pouvez créer des variables basiques simplement en les utilisant; il n'est pas obligatoire de les déclarer formellement. Par exemple, si vous souhaitez déclarer une variable qui contient la date du jour plus 30 jours, il vous suffit d'écrire ce qui suit :

```
MyDate:=Current date+30 //MyDate est créé
// 4D identifie le type date
// et affecte la date courante + 30 jours
```

Utilisation du mot-clé `var`

Il est recommandé de déclarer les variables à l'aide du mot-clé `var`, car cette syntaxe permet de lier les variables objet aux classes. L'utilisation de cette syntaxe améliore les suggestions de l'éditeur de code et les fonctionnalités type-ahead.

Pour déclarer une variable de n'importe quel type avec le mot-clé `var`, utilisez la syntaxe suivante :

```
var <varName>{; <varName2>;...}{ : <varType>}
```

Par exemple :

```
var $myText : Text //une variable de type texte
var myDate1; myDate2 : Date //plusieurs variables de type texte
var $myFile : 4D.File //une variable objet de classe de fichier
var $myVar //une variable variant
```

`varName` est le nom de la variable, il doit être conforme aux [règles 4D](#) concernant les identifiants.

Cette syntaxe prend uniquement en charge les déclarations de [variables locales](#) et [variables process](#), excluant ainsi les

variables interprocess et les tableaux.

`varType` peut être :

- un [type basique](#), auquel cas la variable contient une valeur du type déclaré,
- une [référence de classe](#) (classe 4D ou classe utilisateur), auquel cas la variable contient une référence à un objet de la classe définie.

Si `varType` est omis, une variable de type variant est créée.

Le tableau suivant répertorie toutes les valeurs `varType` prises en charge :

varType	Contenu
Text	Valeur texte
Date	Valeur date
Heure	Valeur Heure
Booléen	Valeur booléen
Integer	Valeur entier long
Réel	Valeur réel
Pointeur	Valeur pointeur
Image	Valeur image
Blob	Valeur Blob évolutive
Collection	Valeur collection
Variant	Valeur variant
Object	Objet avec classe par défaut (4D.object)
4D.<className>	Objet du nom de la classe 4D
cs.<className>	Objet du nom de la classe utilisateur

Exemples

- Pour déclarer les variables locales et les variables process basiques :

```
var $myText; myText; $vt : Text
var myVar //variant

var $o : Object
//équivalent à :
var $o : 4D.Object
//également équivalent à C_OBJECT($o)
```

- Pour déclarer les variables objet de la classe 4D :

```
var $myFolder : 4D.Folder
var $myFile : 4D.File
```

- Pour déclarer les variables objet de la classe utilisateur :

```
var $myClass : cs.MyClass
var $dataclass : cs.Employee
var $entity : cs.EmployeeEntity
```

Utilisation de la directive C_

Note de compatibilité : cette fonctionnalité n'est pas recommandée pour déclarer des variables dans des méthodes. Il est recommandé d'utiliser le mot-clé **var**.

Les directives du thème "Compilateur" vous permettent de déclarer des variables de types basiques.

Par exemple, si vous souhaitez définir une variable de type texte, il suffira d'écrire :

```
C_TEXT(monTexte)
```

Voici quelques déclarations de variables simples :

```
C_BLOB(vxMyBlob) // La variable process vxMyBlob est déclarée comme variable de type BL0B C_DATE($vdCur  
C_C_LONGINT(vg1;vg2;vg3) // Les 3 variables process vg1, vg2 et vg3 sont déclarées comme variables de t  
C_OBJECT($vObj) // La variable locale $vObj est déclarée comme variable de type Objet  
C_COLLECTION($vCol) // La variable locale $vCol est déclarée comme variable de type Collection
```

Note : Les tableaux sont un type particulier de variables (un tableau est une série ordonnée de variables du même type). Les tableaux sont déclarés avec des commandes spécifiques, telles que `ARRAY LONGINT(alAnArray;10)`. Pour plus d'informations, veuillez consulter le thème [Tableaux](#).

Assigner des valeurs

Vous pouvez donner des valeurs aux variables ou aux tableaux et/ou récupérer leur valeur. Donner une valeur à une variable s'appelle assigner une valeur (ou affecter une valeur) et s'effectue à l'aide de l'opérateur d'assignation (`:=`). L'opérateur d'assignation est également utilisé pour assigner des valeurs aux champs.

L'opérateur d'assignation est un premier moyen pour créer une variable et lui donner une valeur. Vous placez le nom de la variable que vous voulez créer à gauche de l'opérateur. Par exemple :

```
MonNombre:=3
```

crée la variable *MonNombre* et lui donne la valeur numérique 3. Si *MonNombre* existait déjà, elle prend simplement la valeur 3.

Il n'est généralement pas recommandé de créer des variables sans [déclarer leur type](#).

Bien entendu, les variables ne seraient pas très utiles si vous ne pouviez pas récupérer les valeurs qu'elles contiennent. De nouveau, vous utilisez l'opérateur d'assignation. Si vous devez placer la valeur de *MonNombre* dans un champ nommé *[Produits]Taille*, il vous suffit de placer *MonNombre* à droite de l'opérateur d'assignation :

```
[Produits]Taille:=MonNombre
```

Dans ce cas, *[Produits]Taille* vaudrait 3. Cet exemple est plutôt simple, mais il illustre le moyen élémentaire dont vous disposez pour transférer des données d'un objet vers un autre en utilisant le langage.

Vous assignez des valeurs aux éléments du tableau à l'aide d'accolades (`{...}`) :

```
atNoms{1}:="Richard"
```

Variables locales, process et interprocess

Vous pouvez créer trois types de variables : des variables locales, des variables process et des variables interprocess. La différence entre ces trois types de variables est leur portée, ou les objets pour lesquels elles sont disponibles.

Variables locales

Une variable locale, comme son nom l'indique, est locale à une méthode — c'est-à-dire accessible uniquement à l'intérieur de la méthode dans laquelle elle a été créée et inaccessible à l'extérieur de cette méthode. Le fait d'être local à une méthode est formellement qualifié de «portée locale». Les variables locales sont utilisées pour restreindre une variable afin qu'elle ne fonctionne que dans la méthode.

- Eviter des conflits de noms avec les autres variables
- Utiliser temporairement des valeurs,
- Réduire le nombre de variables process

Le nom d'une variable locale commence toujours par le signe dollar (\$) et peut contenir jusqu'à 31 autres caractères. Si vous saisissez un nom plus long, 4D le tronque pour le ramener à 31 caractères.

Lorsque vous développez un projet d'application comportant de nombreuses méthodes et variables, il arrive souvent que vous n'ayez besoin d'utiliser une variable que dans une méthode. Vous pouvez alors créer et utiliser une variable locale, sans devoir vous soucier de l'existence d'une autre variable du même nom ailleurs dans la base.

Souvent, dans une application, des informations ponctuelles sont demandées à l'utilisateur. La commande `Request` permet d'obtenir cette information. Elle affiche une boîte de dialogue comportant un message demandant à l'utilisateur de répondre et, lorsque la réponse est validée, la retourne. Généralement, il n'est pas nécessaire de conserver cette information très longtemps dans vos méthodes. C'est l'endroit parfait pour utiliser une variable locale. Voici un exemple :

```
$vsID:=Request("Saisissez votre numéro d'identification :")
If(OK=1)
    QUERY( [Personnes];[Personnes]ID=$vsID)
End if
```

Cette méthode demande simplement à l'utilisateur de saisir un numéro d'identification. La réponse est placée dans une variable locale, \$vsID, puis la méthode la recherche parmi les champs [Personnes]ID. Une fois la méthode terminée, la variable locale \$vsID est effacée de la mémoire. Ce fonctionnement est bien adapté puisque la variable n'est utile qu'une seule fois et dans cette méthode uniquement.

Note : les paramètres \$1, \$2 etc. passés aux méthodes sont des variables locales. Pour plus d'informations, veuillez consulter le thème [Paramètres](#).

Variables process

Une variable process est "visible" uniquement dans le process où elle a été créée. Elle est utilisable par toutes les méthodes du process et toutes les méthodes appelées depuis le process.

Le nom d'une variable process ne comporte aucun préfixe. Ce nom peut contenir jusqu'à 31 caractères.

En mode interprété, les variables sont gérées dynamiquement; elles sont créées et effacées de la mémoire «à la volée». En mode compilé, tous les process que vous créez (process utilisateur) partagent la même définition de variables process, mais chaque process a une instance différente pour chaque variable. Par exemple, la variable maVar est une certaine variable dans le process P_1 et une autre variable dans le process P_2.

Un process peut lire et écrire des variables process dans un autre process à l'aide des commandes `LIRE VARIABLE PROCESS` et `ECRIRE VARIABLE PROCESS`. Nous vous recommandons de n'utiliser ces commandes que dans le cadre des besoins décrits ci-dessous (qui sont les raisons pour lesquelles ces commandes ont été créées dans 4D) :

- Communication interprocess à des endroits particuliers de votre code
- Gestion du glisser-déposer interprocess
- En client/serveur, communication entre les process sur les postes clients et les procédures stockées exécutées sur le serveur

Pour plus d'informations, reportez-vous à la section Process et à la description de ces commandes.

Variables interprocess

Les variables interprocess sont visibles dans tout le projet et sont disponibles pour tous les process. Les variables interprocess sont principalement utilisées pour le partage d'informations entre les process.

L'utilisation de variables interprocess n'est pas recommandée car elles ne sont pas disponibles depuis le process préemptif et peuvent rendre le code moins maintenable.

Le nom d'une variable interprocess débute toujours par le symbole (<>) — formé du symbole "inférieur à" suivi du symbole "supérieur à" — et peut comporter jusqu'à 31 caractères supplémentaires.

En mode client/serveur, chaque poste (client et serveur) partage la même définition des variables interprocess, mais chacun utilise une instance différente d'une variable.

Tableaux

Un tableau est une série ordonnée de variables de même type. Chaque variable est appelée un élément du tableau. La taille du tableau doit être définie au moment de sa création ; vous pouvez ensuite la modifier aussi souvent que nécessaire en ajoutant, insérant, ou supprimant des éléments, ou en appelant de nouveau la commande que vous avez utilisée pour créer le tableau. Les éléments sont numérotés de 1 à N, où N est la taille du tableau. Un tableau a toujours un [élément zéro](#). Les tableaux sont des variables 4D. Comme toute variable, un tableau a une portée et suit les règles du langage 4D, bien qu'il existe quelques différences spécifiques.

Généralement, il est recommandé d'utiliser des collections plutôt que des tableaux. Les collections sont plus souples et fournissent un large éventail de méthodes spécifiques. Pour plus d'informations, veuillez consulter la section [Collection](#).

Créer des tableaux

Vous créez un tableau au moyen de l'une des commandes de déclaration du thème "Tableaux". Chaque commande de déclaration de tableau peut créer ou redimensionner des tableaux à une ou à deux dimensions. Pour plus d'informations sur les tableaux à deux dimensions, reportez-vous à la section [Tableaux à deux dimensions](#).

Cette ligne de code crée (déclare) un tableau d'entiers de 10 éléments :

```
ARRAY INTEGER(aiAnArray;10)
```

Ensuite, cette ligne de code redimensionne le même tableau à 20 éléments :

```
ARRAY INTEGER(aiAnArray;20)
```

Enfin, cette ligne de code redimensionne le même tableau à 0 élément :

```
ARRAY INTEGER(aiAnArray;0)
```

Affecter des valeurs dans un tableau

Vous référez les éléments d'un tableau en utilisant des accolades ({}). Un nombre entre accolades donne accès à l'adresse d'un élément particulier. L'exemple ci-dessous place cinq noms dans le tableau nommé atNoms et les affiche ensuite dans une fenêtre d'alerte :

```
ARRAY TEXT(atNames;5)
atNames{1}:="Richard"
atNames{2}:="Sarah"
atNames{3}:="Sam"
atNames{4}:="Jane"
atNames{5}:="John"
For($vlElem;1;5)
    ALERT("The element #"+String($vlElem)+" is equal to: "+atNames{$vlElem})
End for
```

Notez la syntaxe atNames{\$vlElem}. Au lieu de spécifier un nombre littéral comme atNames{3}, vous pouvez utiliser une variable numérique indiquant à quel élément d'un tableau vous accédez. Si vous utilisez les itérations permises par les structures répétitives (`For...End for`, `Repeat...Until` or `While...End while`), vous pouvez accéder à tout ou partie des éléments d'un tableau avec très peu de code.

Important : Veillez à ne pas confondre l'opérateur d'affectation (:=) avec l'opérateur de comparaison égal (=). L'affectation et la comparaison sont deux opérations totalement différentes.

Affecter un tableau à un autre

Contrairement à ce que vous pouvez faire avec des variables de type Texte ou Chaîne, vous ne pouvez pas affecter un tableau à un autre tableau. Pour copier (affecter) un tableau à un autre, utilisez la fonction `COPY ARRAY`.

Utiliser l'élément zéro d'un tableau

Un tableau a toujours un élément zéro. Même si l'élément zéro n'est pas affiché lorsqu'un tableau est utilisé pour remplir un objet de formulaire, vous pouvez l'utiliser sans réserve(*) dans le langage.

Voici un autre exemple : vous souhaitez initialiser un objet de formulaire avec une valeur texte mais sans définir de valeur par défaut. Vous pouvez utiliser l'élément zéro du tableau :

```
// // méthode pour une combo box ou une liste déroulante
// liée au tableau de variables atName
Case of
  : Form event code=On Load)
  // Initialise le tableau (comme indiqué ci-dessus)
  // Mais utilise l'élément zéro
  ARRAY TEXT(atName;5)
  atName{0}:=Veuillez sélectionner un élément"
  atName{1}:="Text1"
  atName{2}:="Text2"
  atName{3}:="Text3"
  atName{4}:="Text4"
  atName{5}:="Text5"
  // Positionne le tableau sur l'élément 0
  atName: = 0
End case
```

(*) However, there is one exception: in an array type List Box, the zero element is used internally to store the previous value of an element being edited, so it is not possible to use it in this particular context.

Tableaux à deux dimensions

Chaque commande de déclaration de tableau permet de créer ou de redimensionner des tableaux à une ou à deux dimensions. Exemple :

```
ARRAY TEXT(atTopics;100;50) // Créer un tableau texte composé de 100 lignes de 50 colonnes
```

Les tableaux à deux dimensions sont essentiellement des objets de langage ; vous ne pouvez ni les afficher ni les imprimer.

Dans l'exemple précédent :

- `atTopics` est un tableau à deux dimensions
- `atTopics{8}{5}` est le 5e élément (5e colonne...) de la 8e ligne
- `atTopics{20}` est la 20e ligne et est elle-même un tableau à une dimension
- `Size of array(atTopics)` retourne 100, qui est le nombre de lignes
- `Size of array(atTopics{17})` retourne 50, qui est le nombre de colonnes de la 17e ligne

Dans l'exemple suivant, un pointeur vers chaque champ de chaque table de la base est stocké dans un tableau à deux dimensions :

```

C_LONGINT($vlLastTable;$vlLastField)
C_LONGINT($vlFieldNumber)
// Créer autant de lignes (vides et sans colonnes) qu'il y a de tables
$vlLastTable:=Get last table number
ARRAY POINTER(<>apFields;$vlLastTable;0) //Tableau 2D avec N lignes et zéro colonnes
// Pour chaque table
For($vlTable;1;$vlLastTable)
    If(Is table number valid($vlTable))
        $vlLastField:=Get last field number($vlTable)
    // Donner la valeur des éléments
        $vlColumnNumber:=0
    For($vlField;1;$vlLastField)
        If(Is field number valid($vlTable;$vlField))
            $vlColumnNumber:=$vlColumnNumber+1
    // Insérer une colonne dans la ligne de la table en cours
        INSERT IN ARRAY(<>apFields{$vlTable};$vlColumnNumber;1)
    // Affecter la "cellule" avec le pointeur
        <>apFields{$vlTable}{$vlColumnNumber}:=Field($vlTable;$vlField)
    End if
End for
End if
End for

```

Dans la mesure où le tableau à deux dimensions a été initialisé, vous pouvez obtenir ainsi les pointeurs vers les champs d'une table de votre choix :

```

// Obtenir les pointeurs vers les champs pour la table affichée à l'écran:
COPY ARRAY(<>apFields{Table(Current form table)};$apTheFieldsIamWorkingOn)
// Initialiser les champs booléens et date
For($vlElem;1;Size of array($apTheFieldsIamWorkingOn))
    Case of
        :(Type($apTheFieldsIamWorkingOn{$vlElem}->)=Is date)
            $apTheFieldsIamWorkingOn{$vlElem}->:=Current date
        :(Type($apTheFieldsIamWorkingOn{$vlElem}->)=Is Boolean)
            $apTheFieldsIamWorkingOn{$vlElem}->:=True
    End case
End for

```

Note : Comme le montre cet exemple, les lignes des tableaux à deux dimensions peuvent être ou non de la même taille.

Tableaux et mémoire

A la différence des données que vous stockez sur disque lorsque vous utilisez des tables ou des enregistrements, un tableau réside toujours en mémoire dans son intégralité.

Par exemple, si tous les codes postaux américains étaient saisis dans une table [Codes Postaux], celle-ci contiendrait environ 100 000 enregistrements. De plus, cette table comporterait plusieurs champs : le code postal lui-même ainsi que la ville, le comté et l'état correspondants. Si vous ne sélectionnez que les codes postaux de Californie, 4D crée la sélection d'enregistrements correspondante à l'intérieur de la table [Codes Postaux], et ensuite ne charge les enregistrements que lorsque vous en avez besoin (par exemple, pour les afficher ou les imprimer). En d'autres termes, vous travaillez avec une série ordonnée de valeurs (du même type pour chaque champ) partiellement chargée du disque en mémoire.

Procéder de la même manière avec les tableaux serait laborieux, pour les raisons suivantes :

- Pour maintenir les quatre types d'information (code postal, ville, comté, état), vous auriez besoin de quatre grands tableaux en mémoire.
- Comme un tableau réside en mémoire dans son intégralité, vous seriez obligé de garder tous les codes postaux en mémoire pendant toute la session de travail, même si les données n'étaient pas utilisées en permanence.

- Toujours parce qu'un tableau réside en mémoire dans son intégralité, les quatre tableaux devraient être chargés ou sauvegardés sur le disque à chaque fois que vous démarreriez ou quitteriez l'application, quand bien même les données ne seraient d'aucune utilité pour la session de travail.

Conclusion : Les tableaux ont pour rôle de manipuler une certaine quantité de données pendant une période brève. En contrepartie, comme ils résident en mémoire, ils sont d'une utilisation rapide et facile.

Cependant, dans certaines circonstances, vous pouvez avoir besoin de tableaux contenant des centaines ou des milliers d'éléments. Voici les formules à appliquer pour calculer la quantité de mémoire utilisée pour chaque type de tableau :

Type de Tableau	Calcul de la quantité de mémoire en octets
Blob	(1+nombre d'éléments) * 12 + somme de la taille de chaque blob
Booléen	(31+nombre d'éléments)/8
Date	(1+nombre d'éléments) * 6
Integer	(1+nombre d'éléments) * 2
Entier long	(1+nombre d'éléments) * 4
Object	(1+nombre d'éléments) * 8 + somme de la taille de chaque objet
Image	(1+nombre d'éléments) * 8 + somme de la taille de chaque image
Pointeur	(1+nombre d'éléments) * 8 + somme de la taille de chaque pointeur
Réel	(1+nombre d'éléments) * 8
Text	(1+nombre d'éléments) * 20 + (somme de la taille de chaque texte) * 2
Heure	(1+nombre d'éléments) * 4
Deux dimensions	(1+nombre d'éléments) * 16 + somme de la taille de chaque tableau

Notes :

- La taille d'un texte en mémoire se calcule par la formule ((Longueur + 1) * 2)
- Quelques octets supplémentaires sont requis pour le repérage de l'élément, le nombre d'éléments et le tableau lui-même.

Paramètres

Vous aurez souvent besoin de fournir des valeurs à vos méthodes et fonctions. Vous pouvez facilement effectuer cette opération grâce aux paramètres.

Aperçu

Les paramètres (ou arguments) sont des données dont une méthode ou une fonction de classe a besoin pour s'exécuter. Le terme *paramètres* ou *arguments* est utilisé indifféremment dans ce manuel. Des paramètres sont également passés aux commandes intégrées de 4D. Dans l'exemple ci-dessous, la chaîne "Bonjour" est un paramètre de la commande `ALERTE` :

```
ALERTE("Bonjour")
```

Les paramètres sont passés de la même manière aux méthodes ou aux fonctions de classe (class functions). Par exemple, si une fonction de classe nommée `getArea()` accepte deux paramètres, voilà à quoi pourrait ressembler un appel à la fonction de classe :

```
$area:=$o.getArea(50;100)
```

Ou si une méthode projet `FAIRE QUELQUE CHOSE` accepte trois paramètres, l'appel à cette méthode pourrait être de la forme suivante :

```
FAIRE QUELQUE CHOSE($AvecCeci;$EtCela;$CommeCeci)
```

Les paramètres d'entrée sont séparés par des points-virgules (;).

Les mêmes principes s'appliquent lorsque des méthodes sont exécutées via des commandes dédiées, comme par exemple :

```
EXECUTE METHOD IN SUBFORM("Cal2";"SetCalendarDate";*;!05/05/20!)
//passez la date du !05/05/20! comme paramètre de SetCalendarDate
// dans le contexte d'un sous-formulaire
```

Les données peuvent également être retournées à partir de méthodes et de fonctions de classe. Par exemple, la ligne d'instruction suivante utilise une commande 4D, `Length`, qui retourne la longueur d'une chaîne. La valeur renvoyée par `Longueur` est placée dans une variable appelée `MaLongueur`.

```
MaLongueur:=Length("Comment suis-je arrivé là ?")
```

Toute sous-routine peut retourner une valeur. Un seul paramètre de sortie peut être déclaré par méthode ou fonction de classe.

Les valeurs d'entrée et de sortie sont évaluées au moment de l'appel et copiées dans des variables locales au sein de la fonction de classe ou de la méthode appelée. Deux syntaxes sont possibles pour déclarer des variables de paramètres dans le code appelé :

- les **paramètres nommés** (recommandé dans la plupart des cas) ou
- les **paramètres séquentiels**.

Les syntaxes nommées et séquentielles peuvent être combinées sans restriction pour déclarer des paramètres. Par exemple :

```
Function add($x : Integer)
  var $0;$2 : Integer
  $0:=$x+$2
```

Paramètres nommés

Dans les méthodes et fonctions de classe qui sont appelées, les valeurs des paramètres sont assignées aux variables locales. Vous pouvez déclarer des paramètres en utilisant un nom de paramètre avec un type de paramètre, séparés par deux-points.

- Pour les fonctions de classe, les paramètres sont déclarés via le mot clé `Function`.
- Pour les méthodes (méthodes projet, méthodes objet, méthodes base et triggers), les paramètres sont déclarés à l'aide du mot clé `#DECLARE` saisi au début du code de la méthode.

Voici quelques exemples :

```
Function getArea($width : Integer; $height : Integer) -> $area : Integer
```

```
//myProjectMethod
#DECLARE ($i : Integer) -> $myResult : Object
```

Les règles suivantes s'appliquent :

- La ligne de déclaration doit être la première ligne de code de la méthode ou de la fonction, sinon une erreur est affichée (seuls des commentaires ou des sauts de ligne peuvent précéder la déclaration).
- Les noms des paramètres doivent commencer par un caractère `$` et être conformes aux [règles de nommage des propriétés](#).
- Les paramètres multiples (et leurs types) sont séparés par des points-virgules `(;)`.
- Les syntaxes multilignes sont prises en charge (en utilisant le caractère `"")`).

Par exemple, lorsque vous appelez une fonction `getArea()` avec deux paramètres :

```
$area:=$o.getArea(50;100)
```

Dans le code de la fonction de classe, la valeur de chaque paramètre est copiée dans le paramètre déclaré correspondant :

```
// Class: Polygon
Function getArea($width : Integer; $height : Integer)-> $area : Integer
  $area:=$width*$height
```

Si le type n'est pas défini, le paramètre sera défini comme [Variant](#).

Tous les types de méthodes 4D prennent en charge le mot-clé `#DECLARE`, y compris les méthodes base. Par exemple, dans la méthode base `On Web Authentication`, vous pouvez déclarer des paramètres nommés :

```
// méthode base On Web Authentication
#DECLARE ($url : Text; $header : Text; \
$BrowserIP : Text; $ServerIP : Text; \
$user : Text; $password : Text) \
-> $RequestAccepted : Boolean
$entitySelection:=ds.User.query("login=:1"; $user)
// vérifier le hash du mot de passe...
```

Valeur renvoyée

Vous déclarez le paramètre de retour d'une fonction en ajoutant une flèche (->) et la définition du paramètre après la liste des paramètres d'entrée. Par exemple :

```
Function add($x : Variant; $y : Integer) -> $result : Integer
```

Vous pouvez aussi déclarer le paramètre de retour en indiquant uniquement `: type`, auquel cas il pourra être géré via le [mot-clé return](#) ou via `$0` en [syntaxe séquentielle](#)). Par exemple :

```
Function add($x : Variant; $y : Integer): Integer
$0:=$x+$y
```

Type de données pris en charge

Avec les paramètres nommés, vous pouvez utiliser les mêmes types de données que ceux qui sont [pris en charge par le mot-clé var](#), y compris les objets des classes. Par exemple :

```
Function saveToFile($entity : cs.ShapesEntity; $file : 4D.File)
```

Paramètres séquentiels

Comme alternative à la syntaxe des [paramètres nommés](#), vous pouvez déclarer les paramètres en utilisant des variables numérotées séquentiellement : \$1, \$2, \$3, et ainsi de suite. La numérotation des variables locales représente l'ordre des paramètres.

Bien que cette syntaxe soit prise en charge par les fonctions de classes, il est recommandé d'utiliser la syntaxe des [paramètres nommés](#) dans ce cas.

Par exemple, lorsque vous appelez une méthode projet `D0_SOMETHING` avec trois paramètres :

```
FAIRE QUELQUE CHOSE($AvecCeci;$EtCela;$CommeCeci)
```

Dans le code de la méthode, la valeur de chaque paramètre est automatiquement copiée dans des variables \$1, \$2, \$3 :

```
//Code de la méthode FAIRE QUELQUE CHOSE
//Supposons que tous les paramètres sont de type texte
C_TEXT($1;$2;$3)
ALERT("J'ai reçu "+$1+" et "+$2+" et aussi "+$3)
//$1 contient le paramètre $AvecCeci
//$2 contient le paramètre $EtCela
//$3 contient le paramètre $CommeCeci
```

Valeur renvoyée

La valeur à renvoyer est automatiquement placée dans la variable locale `$0`.

Par exemple, la méthode suivante, appelée `Uppercase4`, renvoie une chaîne dont les quatre premiers caractères ont été passés en majuscules :

```
$0:=Uppercase(Substring($1;1;4))+Substring($1;5)
```

Voici un exemple qui utilise la méthode `Uppercase4` :

```
$NewPhrase:=Uppercase4("This is good.")
```

Dans cet exemple, la variable `$NewPhrase` prend la valeur "THIS is good."

La valeur renvoyée, `$0`, est une variable locale à la sous-routine. Elle peut être utilisée en tant que telle à l'intérieur de la sous-routine. Par exemple, vous pouvez écrire :

```
// Faire_quelque chose  
$0:=Uppercase($1)  
ALERT($0)
```

Dans cet exemple, `$0` reçoit d'abord la valeur de `$1`, puis est utilisée en tant que paramètre de la commande `ALERT`. Dans une sous-méthode, vous pouvez utiliser `$0` comme n'importe quelle autre variable locale. C'est 4D qui renvoie la valeur finale de `$0` (sa valeur au moment où la sous-routine se termine) à la méthode appelée.

Type de données pris en charge

Vous pouvez utiliser n'importe quelle [expression](#) comme paramètre séquentiel, à l'exception des :

- tables
- arrays

Les expressions tables ou arrays (tableaux) peuvent être passées uniquement [par référence en utilisant un pointeur](#).

return {expression}

► Historique

L'instruction `return` met fin à l'exécution d'une fonction ou d'une méthode et peut être utilisée pour renvoyer une expression à l'appelant.

Par exemple, la fonction suivante renvoie le carré de son argument, `$x`, où `$x` est un nombre.

```
Function square($x : Integer)  
  return $x * $x
```

En interne, `return x` exécute `$0:=x` ou (si elle est déclarée) `myReturnValue:=x`, et renvoie à l'appelant. Si `return` est utilisé sans expression, la fonction ou la méthode renvoie une valeur nulle du type de retour déclaré (le cas échéant), sinon elle est *indéfinie*.

L'instruction `return` peut être utilisée avec la syntaxe standard pour les [valeurs renvoyées](#) (la valeur renvoyée doit être du type déclaré). Cependant, notez qu'elle met immédiatement fin à l'exécution du code. Par exemple :

```

Function getValue
    $0:=10
    return 20
    // retourne 20

Function getValue -> $v : Integer
    return 10
    $v:=20 // jamais exécutée
    // retourne 10

```

Indirections sur les paramètres (\${N})

Les méthodes projets 4D acceptent un nombre variable de paramètres. Vous pouvez adresser ces paramètres avec une boucle `For...End for`, la commande `Count parameters` et la syntaxe d'indirection des paramètres. Au sein de la méthode, une adresse d'indirection est formatée `${N}` , où `N` est une expression numérique. On qualifie `${N}` de paramètre générique.

Utilisation des paramètres génériques

Par un exemple, considérons une méthode qui calcule une somme de valeurs renournée suivant un format passé comme paramètre. A chaque appel à cette méthode, le nombre de valeurs à additionner peut varier. Il faudra donc passer comme paramètre à notre méthode les valeurs, en nombre variable, et le format, exprimé sous forme d'une chaîne de caractères.

Voici la méthode nommée `MySum` :

```

#DECLARE($format : Text) -> $result : Text
$sum:=0
For($i;2;Count parameters)
    $sum:=$sum+${$i}
End for
$result:=String($sum;$format)

```

Les paramètres de la méthode doivent être passés dans le bon ordre : le format d'abord et ensuite les valeurs, dont le nombre peut varier :

```

Result:=MySum("##0.00";125,2;33,5;24) //"182.70"
Result:=MySum("000";1;2;200) //"203"

```

Notez que même si vous avez déclaré 0, 1, ou plus paramètres dans la méthode, vous pouvez toujours passer le nombre de paramètres que vous voulez. Tous les paramètres sont accessibles dans la méthode appelée via la syntaxe `${N}` , et leur type est `Variant` par défaut (vous pouvez déclarer ces paramètres à l'aide d'une [directive de compilation](#)). Il vous suffit simplement de vous assurer que ces paramètres existent, grâce à la commande `Count parameters`. Par exemple :

```

//foo method
#DECLARE($p1: Text;$p2 : Text; $p3 : Date)
For($i;1;Count parameters)
    ALERT("param "+String($i)+" = "+String(${$i}))
End for

```

Cette méthode peut être appelée :

```

foo("hello";"world";!01/01/2021!;42;?12:00:00?) //des paramètres supplémentaires sont passés

```

Pour une bonne gestion de cette indirection, il est important de respecter la convention suivante : si tous les paramètres ne sont pas adressés par indirection, ce qui est le cas le plus fréquent, il faut que les paramètres adressés par indirection soient passés en fin de liste.

Déclaration des paramètres génériques

De même que pour les autres variables locales, la déclaration du paramètre générique par directive de compilation n'est pas obligatoire. Il est néanmoins recommandé d'éviter toute ambiguïté. Les paramètres génériques non déclarés obtiennent automatiquement le type [Variant](#).

Pour déclarer des paramètres génériques, utilisez une directive du compilateur à laquelle vous passez \${N} comme paramètre, où N spécifie le premier paramètre générique.

```
C_TEXT(${4})
```

La déclaration de paramètres génériques ne peut se faire qu'avec [la syntaxe séquentielle](#).

La commande ci-dessus signifie que tous les paramètres à partir du quatrième (inclus) seront adressés par indirection. Ils seront tous de type text. Les types de \$1, \$2 et \$3 pourront être quelconques. En revanche, si vous utilisez \$2 par indirection, le type utilisé sera le type générique. Il sera donc de type texte, même si pour vous, par exemple, il était de type Réel.

Le nombre, dans la déclaration, doit être une constante et non une variable.

Déclaration des paramètres pour le mode compilé

Pour éviter tout conflit, vous devez déclarer chaque paramètre dans les méthodes ou fonctions appelées en [mode interprété](#), même si cela est facultatif.

Lorsque vous utilisez la syntaxe des [variables nommées](#), les paramètres sont automatiquement déclarés à l'aide du mot-clé #DECLARE ou du prototype [Function](#). Par exemple :

```
Function add($x : Variant; $y : Integer)-> $result : Integer
// tous les paramètres sont déclarés avec leur type
```

Lorsque vous utilisez [la syntaxe de variable séquentielle](#), vous devez vous assurer que tous les paramètres sont correctement déclarés. Dans l'exemple suivant, la méthode projet ajoutCapitale accepte un paramètre texte et retourne un résultat texte :

```
// Méthode projet ajoutCapitale
// ajoutCapitale ( Texte ) -> Texte
// ajoutCapitale( Chaîne source ) -> chaîne avec la première lettre capitale

C_TEXTE($0;$1)
$0:=Majusc(Sous chaine($1;1;1))+Minusc(Sous chaine($1;2))
```

L'utilisation de commandes telles que [New process](#) avec les méthodes process qui acceptent les paramètres nécessite également que les paramètres soient explicitement déclarés dans la méthode appelée. Par exemple :

```

C_TEXT($string)
C_LONGINT($idProc;$int)
C_OBJECT($obj)

$idProc:=New process("foo_method";0;"foo_process";$string;$int;$obj)

```

Ce code peut être exécuté en mode compilé, uniquement si "foo_method" déclare ses paramètres :

```

//foo_method
C_TEXT($1)
C_LONGINT($2)
C_OBJECT($3)
...

```

En mode compilé, vous pouvez regrouper tous les paramètres de variables locales pour les méthodes projets dans un méthode spécifique avec un nom commençant par "Compiler". Dans cette méthode, vous pouvez prédéclarer les paramètres de chaque méthode, comme par exemple :

```

// Compiler_method
C_REAL(OneMethodAmongOthers;$1)

```

Pour plus d'informations, consultez la page [Modes interprété et compilé](#).

La déclaration des paramètres est également obligatoire dans les contextes suivants (ces contextes ne prennent pas en charge les déclarations dans une méthode "Compiler") :

- Méthodes base - Par exemple, la méthode base Sur connexion Web reçoit six paramètres, allant de \$1 à \$6, de type Texte. Au début de la méthode base, vous devez écrire (même si tous les paramètres ne sont pas utilisés) :

```

// Sur connexion Web
C_TEXT($1;$2;$3;$4;$5;$6)

```

Vous pouvez également utiliser les [paramètres nommés](#) avec le mot-clé #DECLARE .

- Triggers - Le paramètre \$0 (Entier long), qui résulte d'un trigger, sera typé par le compilateur si le paramètre n'a pas été explicitement déclaré. Néanmoins, si vous souhaitez le déclarer, vous devez le faire dans le trigger lui-même.
- Objets formulaires qui acceptent l'événement formulaire Sur glisser - Le paramètre \$0 (Entier long), qui résulte de l'événement formulaire Sur glisser est typé par le compilateur si le paramètre n'a pas été explicitement déclaré. Néanmoins, si vous souhaitez le déclarer, vous devez le faire dans la méthode projet. Note : Le compilateur n'initialise pas le paramètre \$0. Ainsi, dès que vous utilisez l'événement formulaire Sur glisser , vous devez initialiser \$0. Par exemple :

```

C_LONGINT($0)
If(Form event=On Drag Over)
    $0:=0
    ...
    If($DataType=Is picture)
        $0:=-1
    End if
    ...
End if

```

Type de paramètre erroné

L'appel d'un paramètre de type incorrect est une [erreur](#) qui empêche une exécution correcte. Par exemple, si vous écrivez les méthodes suivantes :

```
// method1  
#DECLARE($value : Text)
```

```
// method2  
method1(42) //mauvais type, texte attendu
```

Ce cas est traité par 4D en fonction du contexte :

- dans les [projets compilés](#), une erreur est générée à l'étape de compilation lorsque cela est possible. Sinon, une erreur est générée lorsque la méthode est appelée.
- dans les projets interprétés :
 - si le paramètre a été déclaré en utilisant [la syntaxe nommée](#) (`#DECLARE` ou `Function`), une erreur est générée lorsque la méthode est appelée.
 - si le paramètre a été déclaré en utilisant [la syntaxe séquentielle](#) (`C_XXX`), aucune erreur n'est générée, la méthode appelée reçoit une valeur vide du type attendu.

Variables d'entrée/de sortie

Dans une sous-méthode, vous pouvez utiliser les paramètres `$1`, `$2...` comme n'importe quelle autre variable locale. Toutefois, dans le cas où vous utilisez des commandes qui modifient la valeur de la variable passée en paramètre (par exemple `Trouver dans champ`), les paramètres `$1`, `$2`, etc. ne peuvent pas être utilisés directement. Vous devez d'abord les recopier dans des variables locales standard (par exemple `$mavar:=$1`).

Utilisation des propriétés d'objet comme paramètres nommés

L'utilisation d'objets en tant que paramètres vous permet de gérer des paramètres nommés. Ce style de programmation est simple, souple et facile à lire.

Par exemple, si vous utilisez la méthode `CreatePerson` :

```
//CreatePerson  
var $person : Object  
$person:=New object("Name";"Smith";"Age";40)  
ChangeAge($person)  
ALERT(String($person.Age))
```

Dans la méthode `ChangeAge`, vous pouvez écrire :

```
//ChangeAge  
var $1; $para : Object  
$para:=$1  
$para.Age:=$para.Age+10  
ALERT($para.Name+" is "+String($para.Age)+" years old.")
```

C'est un moyen puissant de définir des [paramètres optionnels](#) (voir ci-dessous également). Pour gérer les paramètres manquants, vous pouvez :

- vérifier si tous les paramètres attendus sont fournis en les comparant à la valeur `Null`, ou
- prédefinir les valeurs des paramètres, ou
- les utiliser sous forme de valeurs vides.

Dans la méthode `ChangeAge` ci-dessus, les propriétés `Age` et `Nom` sont obligatoires et pourraient générer des erreurs si elles sont manquantes. Pour éviter cela, vous pouvez simplement écrire :

```
//ChangeAge
var $1; $para : Object
$para:=$1
$para.Age:=Num($para.Age)+10
ALERT(String($para.Name)+" is "+String($para.Age)+" years old.")
```

Les deux paramètres sont alors optionnels. S'ils ne sont pas renseignés, le résultat sera "a 10 ans", mais aucune erreur ne sera générée.

Enfin, les paramètres nommés permettent de maintenir et de reproduire des applications en toutes simplicité et sécurité. Imaginez que vous réalisez, par la suite, qu'ajouter 10 ans n'est pas toujours approprié. Vous aurez besoin d'un autre paramètre pour définir le nombre d'années à ajouter. Vous pouvez écrire :

```
$person:=New object("Name","Smith","Age",40,"toAdd",10)
ChangeAge($person)

//ChangeAge
var $1;$para : Object
$para:=$1
If ($para.toAdd=NULL)
  $para.toAdd:=10
End if
$para.Age:=Num($para.Age)+$para.toAdd
ALERT(String($para.Name)+" is "+String($para.Age)+" years old.")
```

Ici, toute la puissance réside dans le fait de ne pas avoir à changer votre code existant. Cela fonctionnera toujours dans l'ancienne version, mais le cas échéant, vous pouvez utiliser une autre valeur que 10 ans.

Avec les variables nommées, n'importe quel paramètre peut être optionnel. Dans l'exemple ci-dessus, tous les paramètres sont optionnels et peuvent être donnés, dans n'importe quel ordre.

Paramètres optionnels

Dans le manuel *Langage de 4D*, les caractères `{ }` (accolades) indiquent des paramètres facultatifs. Par exemple, `ALERT(message{; okButtonTitle})` signifie que le paramètre `okButtonTitle` doit être omis lors de l'appel de la commande. Vous pouvez l'appeler comme suit :

```
ALERT("Etes-vous sûr?";"Oui, je le suis") //2 paramètres
ALERT("Temps écoulé") //1 paramètre
```

Les méthodes et les fonctions 4D acceptent également de ces paramètres optionnels. Vous pouvez déclarer un nombre quelconque de paramètres. Si vous appelez une méthode ou une fonction avec moins de paramètres que ceux déclarés, les paramètres manquants sont traités comme des valeurs par défaut dans le code appelé, [en fonction de leur type](#). Par exemple :

```
// fonction "concat" de myClass
Function concat ($param1 : Text ; $param2 : Text)->$result : Text
$result:=$param1+" "+$param2
```

```
// Méthode appelante
$class:=cs.myClass.new()
$class.concatenate("Hello") // "Hello "
$class.concatenate() // Affiche "
```

Vous pouvez également appeler une méthode ou une fonction avec plus de paramètres que ceux déclarés. Ils seront disponibles dans le code appelé grâce à la [syntaxe \\${N}](#).

A l'aide de la commande `Count parameters` contenue dans la méthode appelée, vous pouvez détecter le nombre de paramètres et effectuer des opérations différentes en fonction de ce nombre.

L'exemple suivant affiche un message et peut insérer le texte dans un document sur disque ou dans une zone 4D Write Pro :

```
// APPEND TEXT Project Method
// APPEND TEXT ( Text { ; Text { ; Object } } )
// APPEND TEXT ( Message { ; Path { ; 4DWPArea } } )

Method($message : Text; $path : Text; $wpArea : Object)

ALERT($message)
If(Count parameters>=3)
    WP SET TEXT($wpArea;$1;wk append)
Else
    If(Count parameters>=2)
        TEXT TO DOCUMENT($path;$message)
    End if
End if
```

Une fois que cette méthode projet a été ajoutée à votre application, vous pouvez écrire :

```
APPEND TEXT(vtSomeText) //Affichera uniquement le message
APPEND TEXT(vtSomeText;$path) //Affiche le message et l'annexe au document dans $path
APPEND TEXT(vtSomeText;"";$wpArea) //Affiche le message et l'écrit dans $wpArea
```

Lorsque les paramètres sont nécessaires dans vos méthodes, vous pouvez également envisager des [propriétés d'objet comme paramètres nommés](#) pour gérer plusieurs paramètres de manière flexible.

Valeurs ou références

Lorsque vous passez un paramètre, 4D évalue toujours l'expression du paramètre dans le contexte de la méthode appelée et définit la valeur résultante sur les variables locales dans la fonction de classe ou la sous-routine. Les variables/paramètres locaux ne correspondent pas aux véritables champs, variables ou expressions passés par la méthode appelée; ils contiennent uniquement les valeurs qui n'ont pas été passées. Les variables/paramètres locaux ne correspondent pas aux véritables champs, variables ou expressions passés par la méthode appelée; ils contiennent uniquement les valeurs qui n'ont pas été passées. Par exemple :

```
//Voici du code extrait de la méthode MY_METHOD
DO_SOMETHING([People]Name) //Let's say [People]Name value is "williams"
ALERT([People]Name)

//Voici du code extrait de la méthode DO_SOMETHING
$1:=Uppercase($1)
ALERT($1)
```

La boîte de dialogue d'alerte affichée par `FAIRE QUELQUE CHOSE` contiendra "WILLIAM" et celle affichée par `MA METHODE` contiendra "william". La méthode a modifié localement la valeur du paramètre \$1, mais cela n'affecte pas la valeur du champ `[Personnes]Nom` passé en paramètre par la méthode `MA METHODE`.

Si vous voulez réellement que la méthode `FAIRE QUELQUE CHOSE` modifie la valeur du champ, deux solutions s'offrent à vous :

1. Plutôt que de passer le champ à la méthode, vous lui passez un pointeur :

```
//Voici du code extrait de la méthode MY_METHOD
DO_SOMETHING(->[People]Name) //Let's say [People]Name value is "williams"
ALERT([People]Last Name)

//Voici du code extrait de la méthode DO_SOMETHING
$1->:=Uppercase($1->
ALERT($1->)
```

Ici, le paramètre n'est pas le champ lui-même, mais un pointeur vers le champ. Ainsi, à l'intérieur de la méthode `FAIRE QUELQUE CHOSE`, \$1 ne contient plus la valeur du champ mais un pointeur vers le champ. L'objet référencé par \$1 (\$1-> dans le code ci-dessus) est le champ lui-même. Par conséquent, la modification de l'objet référencé dépasse les limites de la sous-routine et le champ lui-même est affecté. Dans cet exemple, les deux boîtes de dialogue d'alerte afficheront "WILLIAM".

2. Plutôt que la méthode `FAIRE QUELQUE CHOSE` "fasse quelque chose", vous pouvez la réécrire de manière à ce qu'elle retourne une valeur.

```
//Voici du code extrait de la méthode MY_METHOD
[People]Name:=DO_SOMETHING([People]Name) //Let's say [People]Name value is "williams"
ALERT([People]Name)

//Voici du code extrait de la méthode DO_SOMETHING
$0:=Uppercase($1)
ALERT($0)
```

Cette deuxième technique de renvoi d'une valeur par une sous-routine est appelée «utilisation d'une fonction». Ceci est décrit dans le paragraphe [Valeurs renournées](#).

Cas particuliers : objets et collections

Veillez à ce que les types de données d'Objet et Collection ne puissent être gérés que via une référence (c'est-à-dire un* pointeur* interne).

Par conséquent, lorsque vous utilisez des types de données comme paramètres, `$1, $2 ...` ne contiennent pas des *valeurs*, mais des *références*. La modification de la valeur des paramètres `$1, $2 ...` dans la sous-routine sera propagée à chaque fois que l'objet ou la collection source est utilisé(e). C'est le même principe que pour [les pointeurs](#), à la différence que les paramètres `$1, $2...` n'ont pas besoin d'être déréférencés dans la sous-routine.

Par exemple, considérons que la méthode `CreatePerson`, qui crée un objet et qui l'envoie comme paramètre :

```
//CreatePerson
var $person : Object
$person:=New object("Name";"Smith";"Age";40)
ChangeAge($person)
ALERT(String($person.Age))
```

La méthode `ChangeAge` ajoute 10 à l'attribut Age de l'objet reçu

```
//ChangeAge  
#DECLARE ($person : Object)  
$person.Age:=$person.Age+10  
ALERT(String($person.Age))
```

Si vous exécutez la méthode `CreatePerson`, les deux messages d'alerte contiendront "50" car le même objet est traité par les deux méthodes.

4D Server : Lorsque des paramètres sont passés entre des méthodes qui ne sont pas exécutées sur la même machine (lors de l'utilisation de l'option Exécuter sur serveur par exemple), il n'est pas possible d'utiliser des références. Dans ce cas, ce sont des copies des paramètres objet ou collection qui sont envoyées au lieu de références.

Objets et collections partagés

Les objets partagés et les collections partagées sont des [objets](#) et des [collections](#) spécifiques dont le contenu est partagé entre les process. Comparés aux [Variables interprocess](#), les objets partagés et les collections partagées ont l'avantage d'être compatibles avec les Process 4D préemptifs : il peuvent être passés en paramètres (par référence) aux commandes telles que `New process` ou `CALL WORKER`.

Les objets partagés et les collections partagées peuvent être stockés dans des variables déclarées à l'aide des commandes standard `C_OBJECT` et `C_COLLECTION`, mais doivent être instanciées à l'aide de commandes spécifiques :

- pour créer un objet partagé, utilisez la commande `New shared object`,
- pour créer une collection partagée, utilisez la commande `New shared collection`.

Note : Des objets partagés et des collections partagées peuvent être définis comme propriétés d'objets/éléments de collections standard (non partagés).

Toute modification d'un objet/d'une collection partagé(e) doit s'effectuer à l'intérieur d'une structure `Use...End use`. La lecture d'une valeur d'objet/collection ne nécessite pas de structure `Use...End use`.

Un catalogue unique et global, retourné par la commande `Storage`, est disponible à tout moment et depuis tout process de l'application et de ses composants.

Utilisation des objets et collections partagés

Une fois instanciés à l'aide des commandes `New shared object` ou `New shared collection`, les objets partagés et les collections partagées peuvent être modifiés et lus depuis n'importe quel process.

Modification

Les modifications suivantes peuvent être effectuées sur les objets partagés et les collections partagées :

- ajout ou suppression de propriétés d'objets,
- ajout ou modification de valeurs (prises en charge par les objets/collections partagé(e)s), y compris d'autres objets et collections partagé(s) (ce qui crée un groupe partagé, cf. ci-dessous).

Toute instruction de modification d'objet ou de collection partagé(e) doit être encadrée par les mots-clés `Use...End use`, sinon une erreur est générée.

```
$s_obj:=New shared object("prop1";"alpha")
Use($s_obj)
  $s_obj.prop1:="omega"
End Use
```

Un objet/une collection partagé(e) ne peut être modifié(e) que par un seul process à la fois. `Use` permet de verrouiller l'objet/la collection partagé(e) pour les autres threads, tandis que `End use` déverrouille l'objet/la collection partagé(e) (si le compteur de verrouillage est à 0, voir ci-dessous). Toute tentative de modification d'un objet/d'une collection partagé(e) sans au moins un appel à `Use...End use` génère une erreur. Lorsqu'un process appelle `Use...End use` avec un objet/une collection partagé(e) qui est déjà "utilisé(e)" par un autre process, il est simplement mis en attente jusqu'à ce qu'il soit déverrouillé par l'appel à `End use` (aucune erreur n'est générée). En conséquence, les instructions situées à l'intérieur des structures `Use...End use` doivent toujours s'exécuter rapidement et déverrouiller les éléments dès que possible. Il est donc fortement déconseillé de modifier un objet ou une collection partagé(e) directement depuis l'interface, par exemple depuis une boîte de dialogue.

L'assignation d'objets/collections partagé(e)s à des propriétés ou éléments d'autres objets/collections partagé(e)s est autorisée et entraîne la création de groupes partagés. Un groupe partagé est automatiquement créé lorsqu'un objet ou une collection partagé(e) est assigné(e) en tant que valeur de propriété ou élément à un autre objet ou collection partagé(e). Les groupes partagés permettent d'imbriquer des objets et collections partagé(e)s mais nécessitent

d'observer des règles supplémentaires :

- L'appel de `Use` avec un(e) objet/collection partagé(e) appartenant à un groupe provoquera le verrouillage des propriétés/éléments de tous/toutes les objets/collections partagé(e)s du groupe et incrémentera son compteur de verrouillage. L'appel à `End use` décrémentera le compteur de verrouillage du groupe et lorsque le compteur est à 0, tous les objets ou collections partagés sont déverrouillés.
- Un objet ou une collection partagé(e) peut appartenir à un seul groupe partagé. Une erreur est générée si vous tentez d'assigner un objet ou une collection appartenant déjà à un groupe à un groupe différent.
- Les objets/collections groupé(e)s ne peuvent plus être dégroupé(e)s. Une fois inclus dans un groupe partagé, un objet ou une collection partagé(e) est lié(e) définitivement au groupe pendant toute la durée de la session. Même si toutes les références de l'objet/la collection sont supprimé(e)s des objets/collections parent(e)s, ils resteront liés.

Reportez-vous à l'exemple 2 pour l'illustration des règles des groupes partagés.

Note : Les groupes partagés sont gérés via une propriété interne nommée *locking identifier*. Si vous avez besoin de plus d'informations sur les mécanismes utilisés, reportez-vous au Guide du développeur de 4D.

Lecture

La lecture de propriétés ou d'éléments d'un objet ou d'une collection partagé(e) est possible sans appel de la structure `Use...End use`, même si l'objet ou la collection partagé(e) est "utilisé(e)" par un autre process.

Cependant, lorsque plusieurs valeurs sont interdépendantes et doivent être lues simultanément, il est nécessaire d'encadrer l'accès en lecture par une structure `Use...End use` pour des raisons de cohérence.

Duplication

Appeler `OB Copy` avec un objet partagé (ou avec un objet dont des propriétés sont des objets partagés) est possible, mais dans ce cas un objet standard (non partagé) est retourné.

Storage

Storage est un objet partagé unique, disponible automatiquement pour chaque application et machine. Cet objet partagé est retourné par la commande `Storage`. Il est destiné à référencer les objets ou collections partagé(e)s défini(e)s durant la session que vous souhaitez rendre accessibles à tous les process, préemptifs ou standard.

A noter que, à la différence de objets partagés standard, l'objet `Storage` ne crée pas de groupe partagé lorsque des objets/collection lui sont assigné(e)s en tant que propriétés. Cette exception permet à l'objet Storage d'être utilisé sans verrouiller les objets/collections partagé(e)s connecté(e)s.

Pour plus d'informations, reportez-vous à la description de la commande `Storage`.

Use...End use

La syntaxe de la structure `Use...End use` est la suivante :

```
Use(Shared_object_or_Shared_collection)
    instruction(s)
End use
```

La structure `Use...End use` définit une séquence d'instructions qui exécutera des tâches sur le paramètre `Shared_object_or_Shared_collection` sous la protection d'un sémaphore interne. `Shared_object_or_Shared_collection` peut être tout objet partagé ou collection partagée valide.

Les objets partagés et les collections partagées permettent d'établir des communications entre les process, en particulier les Process 4D préemptifs. Ils peuvent être passés par référence en paramètre d'un process à un autre. Pour plus de détails sur les objets partagés et les collections partagées, reportez-vous à la page Objets et collections partagés. Encadrer les modifications sur les objets partagés et les collections partagées à l'aide des mots-clés `Use...End use` est obligatoire pour empêcher les accès concurrents entre les process.

- Une fois que la ligne Use est exécutée avec succès, toutes les propriétés/éléments de *Shared_object_or_Shared_collection* sont verrouillé(e)s en écriture pour tous les autres process jusqu'à ce que la ligne End use correspondante soit exécutée.
- La séquence d'*instructions* peut alors effectuer toute modification dans les propriétés/éléments de *Shared_object_or_Shared_collection* sans risque d'accès concurrent.
- Si un autre objet ou collection partagé(e) est ajouté(e) en tant que propriété du paramètre *Shared_object_or_Shared_collection*, il ou elle devient connecté(e) et appartient au même groupe partagé (cf. Utilisation des objets et collections partagés).
- Si un autre process tente d'accéder à une propriété de *Shared_object_or_Shared_collection* ou une propriété connectée alors qu'une séquence Use...End use est en cours d'exécution sur le même *Shared_object_or_Shared_collection*, il est automatiquement placé en attente et attendra jusqu'à ce que la séquence courante soit terminée.
- La ligne End use déverrouille les propriétés de *Shared_object_or_Shared_collection* et tous les objets du même groupe.
- Plusieurs structures Use...End use peuvent être imbriquées dans le code 4D. Dans le cas d'un groupe, chaque Use incrémentera le compteur de verrouillage du groupe et chaque End use le décrémentera ; tous les éléments/propriétés ne seront libérés que lorsque le dernier appel à End use mettra le compteur de verrouillage à 0.

Note : Si une fonction membre d'une collection modifie une collection partagée, un Use interne est automatiquement mis en place pour cette collection partagée durant l'exécution de la fonction.

Exemple 1

Vous souhaitez lancer plusieurs process qui vont effectuer des tâches d'inventaire parmi différents produits et mettre à jour le même objet partagé. Le process principal instancie un objet partagé vide et ensuite lance les autres process, passant en paramètre l'objet partagé et les produits à comptabiliser :

```

ARRAY TEXT($_items;0)
... //remplir le tableau avec les éléments à compter
$nbItems:=Size of array($_items)
C_OBJECT($inventory)
$inventory:=New shared object
Use($inventory)
    $inventory.nbItems:=$nbItems
End use

//Créer un process
For($i;1;$nbItems)
    $ps:=New process("HowMany";0;"HowMany_"+$_items{$i};$_items{$i};$inventory)
    // objet $inventory envoyé par référence
End for

```

Dans la méthode "HowMany", l'inventaire est effectué et l'objet partagé \$inventory est mis à jour dès que possible :

```

C_TEXT($1)
C_TEXT($what)
C_OBJECT($2)
C_OBJECT($inventory)
$what:=$1 //pour une meilleure lisibilité
$inventory:=$2

$count:=CountMethod($what) //méthode de comptage des produits
Use($inventory) //Utiliser l'objet partagé
    $inventory[$what]:=$count //stockage des résultats pour cet article
End use

```

Exemple 2

Les exemples suivants illustrent les règles spécifiques à observer lorsque vous utilisez des groupes partagés :

```
$ob1:=New shared object
$ob2:=New shared object
Use($ob1)
    $ob1.a:=$ob2 //un premier groupe est créé
End use

$ob3:=New shared object
$ob4:=New shared object
Use($ob3)
    $ob3.a:=$ob4 //un 2e groupe est créé
End use

Use($ob1) //Utilisation d'un objet du groupe 1
    $ob1.b:=$ob4 //ERREUR
//$ob4 appartient déjà à un autre groupe
//son assignation n'est pas permise
End use

Use($ob3)
    $ob3.a:=Null //on enlève la référence de $ob4 du groupe 2
End use

Use($ob1) //Utilisation d'un objet du groupe 1
    $ob1.b:=$ob4 //ERREUR
//$ob4 appartient toujours au groupe 2
//son assignation n'est pas permise
End use
```

Classes

Aperçu

Le langage 4D prend en charge le concept de classes. Dans un langage de programmation, l'utilisation d'une classe vous permet de définir le comportement d'un objet avec des propriétés et des fonctions associées.

Chaque objet est une instance de sa classe. Une fois qu'une classe utilisateur (user class) est définie, vous pouvez instancier des objets de cette classe n'importe où dans votre code. Une classe peut s'étendre à une autre classe avec le mot-clé `extend`, puis hériter de ses [fonctions](#) et de ses propriétés ([statiques](#) and [calculés](#)).

Les modèles de classe 4D et de classe JavaScript sont similaires, et sont basés sur une chaîne de prototypes.

Par exemple, vous pouvez créer une classe `Person` avec la définition suivante :

```
//Class: Person.4dm
Class constructor($firstname : Text; $lastname : Text)
    This.firstName:=$firstname
    This.lastName:=$lastname

Function get fullName() -> $fullName : text
    $fullName:=This.firstName+" "+This.lastName

Function sayHello()->$welcome : Text
    $welcome:="Hello "+This.fullName
```

Dans une méthode, créons une "Personne" :

```
var $person : cs.Person //objet de classe Person
var $hello : Text
$person:=cs.Person.new("John";"Doe")
// $person:{firstName: "John"; lastName: "Doe"; fullName: "John Doe"}
$hello:=$person.sayHello() //Hello John Doe
```

Gestion des classes

Définition d'une classe

Une classe utilisateur dans 4D est définie par un fichier de méthode spécifique (.4dm), stocké dans le dossier `/Project/Sources/Classes/`. Le nom du fichier est le nom de la classe.

Lorsque vous nommez des classes, gardez à l'esprit les règles suivantes :

- Un [nom de classe](#) doit être conforme aux [règles de nommage des propriétés](#).
- Les noms de classe sont sensibles à la casse.
- Il n'est pas recommandé de donner le même nom à une classe et à une table de base de données, afin d'éviter tout conflit.

Par exemple, si vous souhaitez définir une classe nommée "Polygon", vous devez créer le fichier suivant :

- Dossier Project
 - Project

Supprimer une classe

Pour supprimer une classe existante, vous pouvez :

- sur votre disque, supprimer le fichier de classe .4dm du dossier "Classes",
- dans l'Explorateur 4D, sélectionner la classe et cliquer sur  ou choisir Déplacer vers la corbeille dans le menu contextuel.

Utiliser l'interface 4D

Les fichiers de classe sont automatiquement stockés à l'emplacement approprié lorsqu'ils sont créés via l'interface de 4D, soit via le menu Fichier, soit via l'Explorateur.

Menu Fichier et barre d'outils

Vous pouvez créer un nouveau fichier de classe pour le projet en sélectionnant Nouveau> Classe... dans le menu Fichier de 4D Developer ou dans la barre d'outils.

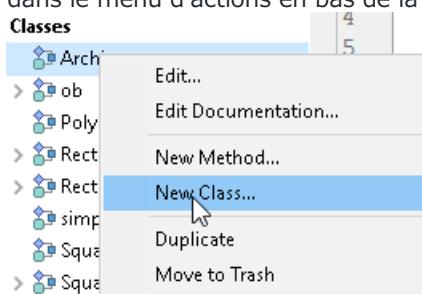
Vous pouvez également utiliser le raccourci Ctrl+Maj+Alt+k.

Explorateur

Dans la page Méthodes de l'Explorateur, les classes sont regroupées dans la catégorie Classes.

Pour créer une nouvelle classe, vous pouvez :

- sélectionnez la catégorie Classes et cliquez sur le bouton .
- sélectionnez Nouvelle classe... dans le menu d'actions en bas de la fenêtre de l'Explorateur ou dans le menu



contextuel du groupe Classes.

- sélectionnez Nouveau> Classe... dans le menu contextuel de la page d'accueil de l'Explorateur.

Prise en charge du code de classe

Dans les différentes fenêtres 4D (éditeur de code, compilateur, débogueur, explorateur d'exécution), le code de classe est essentiellement géré comme une méthode projet avec quelques spécificités :

- Dans l'éditeur de code :
 - une classe ne peut pas être exécutée
 - une fonction de classe est un bloc de code
 - Aller à définition... sur un objet membre permet de rechercher des déclarations de fonction de classe; par exemple, "\$o.f()" donnera comme résultat de recherche "Function f".
 - Chercher les références... sur la déclaration de fonction de classe recherche la fonction utilisée comme membre d'objet; par exemple, "Function f" donnera comme résultat "\$o.f()".
- Dans l'explorateur d'exécution et le Débogueur, les fonctions de classe sont affichées avec le format <ClassName> constructor ou <ClassName>. <FunctionName> .

Class stores

Les classes disponibles sont accessibles depuis leurs class stores. Deux class stores sont disponibles :

- `cs` pour le class store utilisateur
- `4D` pour le class store intégré

CS

`cs -> classStore`

Paramètres	Type		Description
<code>classStore</code>	object	<-	Class store utilisateur utilisateurs pour le projet ou le composant

La commande `cs` retourne le class store utilisateur pour le projet ou le composant courant. La commande `cs` retourne le class store utilisateur pour le projet ou le composant courant. Par défaut, seules les [classes ORDA](#) du projet sont disponibles.

Exemple

Vous souhaitez créer une nouvelle instance d'un objet de `myClass` :

```
$instance:=cs.myClass.new()
```

4D

`4D -> classStore`

Paramètres	Type		Description
<code>classStore</code>	object	<-	Class store 4D

La commande `4D` retourne le class store des classes 4D intégrées disponibles. Elle donne accès à des API spécifiques telles que [CryptoKey](#).

Exemple

Vous souhaitez créer une nouvelle clé dans la classe `CryptoKey` :

```
$key:=4D.CryptoKey.new(New object("type";"ECDSA";"curve";"prime256v1"))
```

L'objet classe

Lorsqu'une classe est [définie](#) dans le projet, elle est chargée dans l'environnement de langage 4D. Une classe est un objet de la [classe "Class"](#). Un objet classe possède les propriétés et fonctions suivantes :

- chaîne `name`
- objet `superclass` (null si aucun)
- fonction `new()`, permettant d'instancier les objets de la classe.

De plus, un objet classe peut référencer un objet `constructor` (facultatif).

Un objet de classe est un [objet partagé](#) et est donc accessible simultanément à partir de différents processus 4D.

Héritage

Si une classe hérite d'une autre classe (c'est-à-dire que le mot-clé [Class extends](#) est utilisé dans sa définition), la classe parente est sa `superclass`.

Lorsque 4D ne trouve pas de fonction ou de propriété dans une classe, il la recherche dans sa `superclass` ; s'il ne la trouve pas, 4D continue la recherche dans la superclasse de la superclasse, et ainsi de suite, jusqu'à ce qu'il n'y ait plus de superclasse (tous les objets héritent de la superclasse "Object").

Mots-clés de classe

Des mots-clés 4D spécifiques peuvent être utilisés dans les définitions de classe :

- `Function <Name>` pour définir les fonctions de classe des objets.
- `Function get <Name>` et `Function set <Name>` pour définir les propriétés calculées des objets.
- `Class constructor` pour définir les propriétés statiques des objets.
- `Class extends <ClassName>` pour définir l'héritage.

Function

Syntaxe

```
Function <name>({$parameterName : type; ...}){->$parameterName : type}  
// code
```

Les fonctions de classe sont des propriétés spécifiques de la classe. Ce sont des objets de la classe [4D.Function](#).

Dans le fichier de définition de classe, les déclarations de fonction utilisent le mot-clé `Function`, et le nom de la fonction. Le nom de la fonction doit être conforme aux [règles de nommage des propriétés](#).

Astuce : Si vous préfixez le nom d'une fonction par un trait de soulignement ("_"), elle ne sera pas proposée par les fonctionnalités d'auto-complétion dans l'éditeur de code 4D. Par exemple, si vous déclarez `Function _myPrivateFunction` dans `MyClass`, elle ne sera pas proposée dans l'éditeur de code lorsque vous tapez `"cs.MyClass. "`.

Immédiatement après le nom de la fonction, les [paramètres](#) de la fonction peuvent être déclarés avec un nom et un type de données affectés, y compris le paramètre de retour (facultatif). Par exemple :

```
Function computeArea($width : Integer; $height : Integer)->$area : Integer
```

Dans une fonction de classe, la commande `This` est utilisée comme instance d'objet. Par exemple :

```
Function setFullscreen($firstname : Text; $lastname : Text)  
    This.firstName:=$firstname  
    This.lastName:=$lastname  
  
Function getFullscreen()->$fullname : Text  
    $fullname:=This.firstName+" "+Uppercase(This.lastName)
```

Pour une fonction de classe, la commande `Current method name` retourne : `<ClassName>.<FunctionName>`, par exemple "MyClass.myFunction".

Dans le code de l'application, les fonctions de classes sont appelées comme des méthodes membres des instances d'objets et peuvent recevoir des [paramètres](#) si besoin. Les syntaxes suivantes sont prises en charge :

- utilisation de l'opérateur `()`. Par exemple, `myObject.methodName("hello")`
- utilisation d'une méthode membre de la classe "4D.Function" :
 - `apply()`
 - `call()`

Thread-safety : Si une fonction de classe n'est pas thread-safe et est appelée par une méthode ayant l'attribut "Peut être exécutée dans un process préemptif" : - le compilateur ne génère aucune erreur (ce qui est différent par rapport aux méthodes classiques), - une erreur est générée par 4D à l'exécution uniquement.

Paramètres

Les paramètres des fonctions sont déclarés via le nom du paramètre et son type, séparés par deux-points. Le nom du paramètre doit être conforme aux [règles de nommage des propriétés](#). Les paramètres multiples (et leurs types) sont séparés par des points-virgules (;).

```
Function add($x; $y : Variant; $z : Integer; $xy : Object)
```

Si le type n'est pas fourni, le paramètre sera défini comme `Variant`.

La [syntaxe 4D classique](#) pour les paramètres de méthodes peut être utilisée pour déclarer les paramètres des fonctions de classes. Les deux syntaxes peuvent être mélangées. Par exemple :

```
Function add($x : Integer)
  var $2; $value : Integer
  var $0 : Text
  $value:=$x+$2
  $0:=String($value)
```

Valeur retournée

Vous déclarez le paramètre de retour d'une fonction (optionnel) en ajoutant une flèche (`->`) et la définition du paramètre de retour après la liste des paramètres d'entrée, ou les deux points (`:`) et le type de paramètre de retour uniquement. Par exemple :

```
Function add($x : Variant; $y : Integer)->$result : Integer
  $result:=$x+$y
```

Vous pouvez également déclarer le paramètre de retour en ajoutant seulement `: type` et utiliser l'expression `return` (elle mettra également fin à l'exécution de la fonction). Par exemple :

```
Function add($x : Variant; $y : Integer): Integer
  // du code
  return $x+$y
```

Exemple 1

```
// Class: Rectangle
Class constructor($width : Integer; $height : Integer)
  This.name:="Rectangle"
  This.height:=$height
  This.width:=$width

// Définition de fonction
Function getArea()->$result : Integer
  $result:=(This.height)*(This.width)
```

```
// Dans une méthode projet

var $rect : cs.Rectangle
var $area : Real

$rect:=cs.Rectangle.new(50;100)
$area:=$rect.getArea() //5000
```

Exemple 2

Dans cet exemple, nous utilisons `l'expression return` :

```
Function getRectArea($width : Integer; $height : Integer) : Integer
    If ($width > 0 && $height > 0)
        return $width * $height
    Else
        return 0
    End if
```

Function get et Function set

Syntaxe

```
Function get <name>()->$result : type
// code
```

```
Function set <name>($parameterName : type)
// code
```

`Function get` et `Function set` sont des accesseurs définissant des propriétés calculées dans la classe. Une propriété calculée est une propriété nommée avec un type de données qui masque un calcul. Lorsqu'on accède à la valeur d'une propriété calculée, 4D substitue le code de l'accesseur correspondant :

- lorsque la propriété est lue, `Function get` est exécutée,
- lorsque la propriété est écrite, `Function set` est exécutée.

Si la propriété n'est pas accédée, le code n'est jamais exécuté.

Les propriétés calculées sont conçues pour gérer les données qui n'ont pas besoin d'être conservées en mémoire. Elles sont généralement basées sur des propriétés persistantes. Par exemple, si un objet de classe contient comme propriété persistante le *prix brut* et le *taux de TVA*, le *prix net* pourrait être traité par une propriété calculée.

Dans le fichier de définition de la classe, les déclarations de propriétés calculées utilisent les mots-clés `Function get` (le *getter*) et `Function set` (le *setter*), suivis du nom de la propriété.

Dans le fichier de définition de la classe, les déclarations de propriétés calculées utilisent les mots-clés `Function get` (le *getter*) et `Function set` (le *setter*), suivis du nom de la propriété.

Dans le fichier de définition de la classe, les déclarations de propriétés calculées utilisent les mots-clés `Function get` (le *getter*) et `Function set` (le *setter*), suivis du nom de la propriété. Le nom doit être conforme aux [règles de nommage des propriétés](#).

`Function get` retourne une valeur du type de la propriété et `Function set` prend un paramètre du type de la propriété. Les deux arguments doivent être conformes aux [paramètres de fonction](#) standard.

Lorsque les deux fonctions sont définies, la propriété calculée est en lecture-écriture. Si seule une `Function get` est définie, la propriété calculée est en lecture seule. Dans ce cas, une erreur est retournée si le code tente de modifier la

propriété. Si seule une `Function set` est définie, 4D retourne *undefined* lorsque la propriété est lue.

Le type de la propriété calculée est défini par la déclaration de type `$return` du *getter*. Il peut s'agir de n'importe quel [type de propriété valide](#).

Assigner *undefined* à une propriété d'objet efface sa valeur tout en préservant son type. Pour ce faire, la `Function get` est d'abord appelée pour récupérer le type de valeur, puis `Function set` est appelée avec une valeur vide de ce type.

Exemple 1

```
//Class: Person.4dm

Class constructor($firstname : Text; $lastname : Text)
    This.firstName:=$firstname
    This.lastName:=$lastname

Function get fullName() -> $fullName : Text
    $fullName:=This.firstName+" "+This.lastName

Function set fullName( $fullName : Text )
    $p:=Position(" "; $fullName)
    This.firstName:=Substring($fullName; 1; $p-1)
    This.lastName:=Substring($fullName; $p+1)
```

```
//dans une méthode projet
$fullName:=$person.fullName // Function get fullName() est appelée
$person.fullName:="John Smith" // Function set fullName() est appelée
```

Exemple 2

```
Function get fullAddress()->$result : Object

    $result:=New object

    $result.fullName:=This.fullName
    $result.address:=This.address
    $result.zipCode:=This.zipCode
    $result.city:=This.city
    $result.state:=This.state
    $result.country:=This.country
```

Class Constructor

Syntaxe

```
// Class: MyClass
Class Constructor({$parameterName : type; ...})
// code
// code
```

Une fonction class constructor, qui accepte des [paramètres](#), peut être utilisée pour définir une classe utilisateur.

Dans ce cas, lorsque vous appelez la fonction `new()`, le class constructor est appelé avec les paramètres optionnellement passés à la fonction `new()`.

Pour une fonction class constructor, la commande `Current method name` retourne : `<ClassName>:constructor`, par exemple "MyClass.constructor".

Exemple :

```
// Class: MyClass
// Class constructor de MyClass
Class Constructor ($name : Text)
  This.name:=$name
```

```
// Dans une méthode projet
// Vous pouvez instancier un objet
var $o : cs.MyClass
$o:=cs.MyClass.new("HelloWorld")
// $o = {"name":"HelloWorld"}
```

Class extends <ClassName>

Syntaxe

```
// Class enfant
Class extends <ParentClass>
```

Le mot-clé `Class extends` est utilisé dans une déclaration de classe pour créer une classe utilisateur "enfant" d'une autre classe utilisateur. La classe "enfant" hérite de toutes les fonctions de la classe "parente".

L'extension de classe doit respecter les règles suivantes :

- Une classe utilisateur ne peut pas étendre une classe 4D système (à l'exception de 4D.Object qui est étendue par défaut pour les classes utilisateurs)
- Une classe utilisateur ne peut pas étendre une classe utilisateur d'un autre projet ou composant.
- Une classe utilisateur ne peut pas s'étendre elle-même.
- Il n'est pas possible d'étendre des classes de manière circulaire (i.e. "a" étend "b" qui étend "a").

La violation de ces règles n'est pas détectable par l'éditeur de code ou l'interpréteur, seul le compilateur et la fonction `vérifier syntaxe` retourneront une erreur dans ce cas.

Une classe étendue peut appeler le constructeur de sa classe parente en utilisant la commande `Super`.

Exemple

Cet exemple crée une classe nommée `Square` à partir d'une classe nommée `Polygon`.

```

//Class: Square

//path: Classes/Square.4dm

Class extends Polygon

Class constructor ($side : Integer)

    // Il appelle le constructeur de la classe parente avec les longueurs
    // fournies pour la largeur et la hauteur du polygone.
    Super($side;$side)
    // Dans les classes dérivées, Super doit être appelé avant que vous puissiez
    // utiliser 'This'.
    This.name:="Square"

Function getArea()
    C_LONGINT($0)
    $0:=This.height*This.width

```

Super

Syntaxe

```
Super {{ param{;...;paramN} }} {-> Object}
```

Paramètres	Type		Description
param	mixte	->	Paramètre(s) à passer au constructeur de la classe parente
Résultat	object	<-	Parent de l'objet

Le mot-clé `Super` permet d'appeler la `superclass`, i.e.

`Super` peut être utilisé de deux différentes manières :

1. A l'intérieur du code de la fonction `constructeur`, `Super` est une commande qui permet d'appeler le constructeur de la superclass. Lorsqu'elle est utilisée dans une fonction constructeur, la commande `Super` est seule et doit être appelée avant que le mot-clé `This` soit utilisé.
- Si tous les class constructors dans l'arbre des héritages ne sont pas appelés correctement, l'erreur -10748 est générée. Il est de la responsabilité du développeur 4D de s'assurer que tous les appels sont valides.
- Si la commande `This` est appelée sur un objet dont les superclasses n'ont pas été construites, l'erreur -10743 est générée.
- Si `Super` est appelée en dehors d'un contexte d'objet, ou sur un objet dont le constructeur de la superclass a déjà été appelé, l'erreur -10746 est générée.

```

// dans la fonction constructor de myClass
var $text1; $text2 : Text
Super($text1) //appel du constructeur de la superclass avec un paramètre text
This.param:=$text2 // utilisation d'un second param

```

2. A l'intérieur d'une `fonction de classe`, `Super` désigne le prototype de la superclass et permet d'appeler une fonction de la hiérarchie de superclasses.

```

Super.doSomething(42) //appelle la fonction "doSomething"
//déclarée parmi les superclasses

```

Exemple 1

Cet exemple illustre l'utilisation de `Super` dans un class constructor. La commande est appelée pour éviter de dupliquer les parties du constructeur qui sont communes aux classes `Rectangle` et `Square`.

```
// Class: Rectangle
Class constructor($width : Integer; $height : Integer)
    This.name:="Rectangle"
    This.height:=$height
    This.width:=$width

Function sayName()
    ALERT("Hi, I am a "+This.name+".") 

// Définition de fonction
Function getArea()
    var $0 : Integer
    $0:=(This.height)*(This.width)

// Function definition
Function getArea()
    var $0 : Integer

    $0:=(This.height)*(This.width)
```

```
//Class: Square

Class extends Rectangle

Class constructor ($side : Integer)
    // Elle appelle le class constructor de la classe parente en lui passant
    // les longueurs fournies pour la largeur et hauteur du polygone
    Super($side;$side)
    // Dans les classes dérivées, Super doit être appelé avant
    // de pouvoir utiliser 'This'
    This.name:="Square"

Function getArea()
    C_LONGINT($0)
    $0:=This.height*This.width
```

Exemple 2

Cet exemple illustre l'utilisation de `Super` dans une fonction de classe. Vous avez créé la classe `Rectangle` contenant une fonction :

```
//Class: Rectangle

Function nbSides()
    var $0 : Text
    $0:="I have 4 sides"
```

Vous avez également créé la classe `Square` contenant une fonction qui appelle la fonction superclasse :

```
//Class: Square

Class extends Rectangle

Function description()
  var $0 : Text
  $0:=Super.nbSides()+" which are all equal"
```

Vous pouvez donc écrire dans une méthode projet :

```
var $square : Object
var $message : Text
$square:=cs.Square.new()
$message:=$square.description() //I have 4 sides which are all equal
```

This

Syntaxe

This -> Object

Paramètres	Type		Description
Résultat	object	<-	Objet courant

Le mot-clé **This** retourne une référence vers l'objet en cours de traitement. Dans 4D, il peut être utilisé dans [différents contextes](#).

Dans la plupart des cas, la valeur de **This** est déterminée par la manière dont une fonction est appelée. Il ne peut pas être défini par affectation lors de l'exécution, et il peut être différent à chaque fois que la fonction est appelée.

Lorsqu'une formule est appelée en tant que méthode membre d'un objet, son **This** désigne l'objet sur lequel la méthode est appelée. Par exemple :

```
$o:=New object("prop";42;"f";Formula(This.prop))
$val:=$o.f() //42
```

Lorsqu'une fonction **class constructor** est utilisée (avec la fonction **new()**), son **This** désigne le nouvel objet en cours de construction.

```
//Class: ob

Class Constructor

// Créer des propriétés en
// les assignant au This
This.a:=42
```

```
// dans une méthode 4D
$o:=cs.ob.new()
$val:=$o.a //42
```

En cas d'appel de la superclasse du constructeur depuis le constructeur en utilisant le mot-clé **Super**, n'oubliez

pas que `This` ne doit pas être appelé avant le constructeur de la superclasse, sinon une erreur est générée.
Voir [cet exemple](#).

Dans tous les cas, `This` se réfère à l'objet sur lequel la fonction a été appelée, comme s'il s'agissait d'une fonction de l'objet.

```
//Class: ob  
  
Function f()  
  $0:=This.a+This.b
```

Vous pouvez donc écrire dans une méthode projet :

```
$o:=cs.ob.new()  
$o.a:=5  
$o.b:=3  
$val:=$o.f() //8
```

Dans cet exemple, l'objet affecté à la variable `$o` n'a pas de propriété `f`, il hérite de celle de sa classe. Comme `f` est appelée comme une méthode de `$o`, son `This` se réfère à `$o`.

Commandes de classes

Plusieurs commandes du langage 4D se rapportent à la manipulation des classes.

OB Class

```
OB Class ( object ) -> Object | Null
```

`OB Class` retourne la classe de l'objet passé en paramètre.

OB Instance of

```
OB Instance of ( object ; class ) -> Boolean
```

`OB Instance of` retourne `true` si `object` appartient à la `class` ou à l'une de ses classes héritées, et `false` sinon.

Conditions et boucles

Quelle que soit la simplicité ou la complexité d'une méthode ou d'une fonction, vous utiliserez toujours un ou plusieurs types de structure de programmation. Les structures de programmation déterminent si et dans quel ordre les lignes d'instructions sont exécutées à l'intérieur d'une méthode. Il existe trois types de structures :

- Séquentielle : une structure séquentielle est une structure simple, linéaire. Une séquence est une série d'instructions que 4D exécute les unes après les autres, de la première à la dernière. Une instruction d'une ligne, fréquemment utilisée pour les méthodes objet, est le cas le plus simple de structure séquentielle. Par exemple :
`[Personnes]Nom:=Uppercase([Personnes]Nom)`
- Conditionnelle : une structure conditionnelle permet aux méthodes de tester une condition et d'exécuter des séquences d'instructions différentes en fonction du résultat. La condition est une expression booléenne, c'est-à-dire pouvant retourner VRAI ou FAUX. L'une des structures conditionnelles est la structure `If...Else...End if`, qui aiguille le déroulement du programme vers une séquence ou une autre. L'autre structure conditionnelle est la structure `Case of...Else...End case`, qui aiguille le programme vers une séquence parmi une ou plusieurs alternatives.
- Répétitive : il est très courant, lorsque vous écrivez des méthodes, de rencontrer des cas où vous devez répéter une séquence d'instructions un certain nombre de fois. Pour traiter ces besoins, le langage 4D vous propose plusieurs structures répétitives :
 - `While...End while`
 - `Repeat...Until`
 - `For...End for`
 - `For each...End for each`

Les boucles sont contrôlées de deux manières : soit elles se répètent jusqu'à ce qu'une condition soit remplie, soit elles se répètent un nombre fixé de fois. Chaque structure répétitive peut être utilisée de l'une ou l'autre manière, mais les boucles `While` et `Repeat` sont mieux adaptées à la répétition jusqu'à ce qu'une condition soit remplie, alors que les boucles `For` sont mieux adaptées à la répétition un certain nombre de fois. `For each...End for each`, destinée à effectuer des boucles dans les objets et les collections, permet de combiner les deux manières.

Note : 4D vous permet d'imbriquer des structures de programmation jusqu'à une "profondeur" de 512 niveaux.

return {expression}

► Historique

L'instruction `return` peut être appelée de n'importe où. Lorsqu'une instruction `return` est utilisée dans une fonction ou une méthode, l'exécution de la fonction ou de la méthode est arrêtée. Le code restant n'est pas exécuté et le contrôle est renvoyé à l'appelant.

L'instruction `return` peut être utilisée pour [retourner une valeur](#) à l'appelant.

Exemple

```
var $message : Text
var $i : Integer

While (True) //boucle infinie
    $i:=$i+1
    $message+=String($i)+"A\r" // jusqu'à 5
    logConsole($message)
    If ($i=5)
        return //stops the loop
    End if
    $message+=String($i)+"B\r" // jusqu'à 4
    logConsole($message)
End while
$message+=String($i)+"C\r" //jamais exécutée
logConsole($message)

// 1A
// 1B
// 2A
// 2B
// 3A
// 3B
// 4A
// 4B
// 5A
```

Structures conditionnelles

Une structure de branchement permet aux méthodes de tester une condition et d'emprunter des chemins alternatifs, en fonction du résultat.

If...Else...End if

La syntaxe de la structure conditionnelle `If...Else...End if` est la suivante :

```
If(Boolean_Expression)
    instruction(s)
Else
    instruction(s)
End if
```

A noter que l'élément `Else` est optionnel, vous pouvez écrire :

```
If(Boolean_Expression)
    instruction(s)
End if
```

La structure `If...Else...End if` permet à votre méthode de choisir dans une alternative, en fonction du résultat, TRUE ou FALSE, d'un test (une expression booléenne). Si l'expression booléenne est TRUE, les instructions qui suivent immédiatement le test sont exécutées. Si l'expression booléenne est FALSE, les instructions suivant la ligne `Else` sont exécutées. Le `Else` est optionnel ; lorsqu'il est omis, c'est la première ligne d'instructions suivant le `End if` (s'il y en a une) qui est exécutée.

A noter que l'expression booléenne est toujours évaluée en totalité. Examinons en particulier le test suivant :

```
If(MethodA & MethodB)
    ...
End if
```

L'expression n'est TRUE que si les deux méthodes sont mises à TRUE. Or, même si `MethodA` retourne FALSE, 4D évaluera quand même `MethodB`, ce qui représente une perte de temps inutile. Dans ce cas, il est préférable d'utiliser une structure du type :

```
If(MethodA)
    If(MethodB)
        ...
    End if
End if
```

Le résultat est équivalent et `MethodB` n'est évaluée que si nécessaire.

Note : L'[opérateur ternaire](#) permet d'écrire des expressions conditionnelles d'une ligne et peut remplacer une séquence complète d'instructions `If... Else`.

Exemple

```

// Demander à l'utilisateur de saisir un nom
$Find:=Request(Type a name)
If(OK=1)
    QUERY([People];[People]LastName=$Find)
Else
    ALERT("You did not enter a name.")
End if
End if
End if
End if
End if

```

Astuce : Il n'est pas obligatoire que des instructions soient exécutées dans chaque branche de l'alternative. Lorsque vous développez un algorithme, ou lorsque vous poursuivez un but précis, rien ne vous empêche d'écrire :

```

If(Expression_booléenne)
Else
    instruction(s)
End if

```

ou :

```

If(Expression_booléenne)
    instruction(s)
Else
End if

```

Au cas ou...Sinon...Fin de cas

La syntaxe de la structure conditionnelle `Case of...Else...End case` est la suivante :

```

Case of
    :(Expression_booléenne)
        instruction(s)
    :(Expression_booléenne)
        statement(s)
    .
    .
    .

    :(Expression_booléenne)
        instruction(s)
Else
    instruction(s)
End case

```

A noter que l'élément `Else` est optionnel, vous pouvez écrire :

```
Case of
:(Expression_booléenne)
    instruction(s)
:(Expression_booléenne)
    statement(s)
.
.
.

:(Expression_booléenne)
    instruction(s)
End case
```

Tout comme la structure `If...Else...End if`, la structure `Case of...Else...End case` permet également à votre méthode de choisir parmi plusieurs séquences d'instructions. A la différence de la structure `If...Else...End`, la structure `Case of...Else...End case` peut tester un nombre illimité d'expressions booléennes et exécuter la séquence d'instructions correspondant à la valeur TRUE.

Chaque expression booléenne débute par le caractère deux points (`:`). La combinaison de deux points et d'une expression booléenne est appelée un cas. Par exemple, la ligne suivante est un cas :

```
: (bValidate=1)
```

Seules les instructions suivant le premier cas TRUE (et ce, jusqu'au cas suivant) seront exécutées. Si aucun des cas n'est TRUE, aucune instruction n'est exécutée (s'il n'y a pas d'élément `Else`).

Vous pouvez placer une instruction Else après le dernier cas. Si tous les cas sont FALSE, les instructions suivant le `Else` seront exécutées.

Exemple

Cet exemple teste une variable numérique et affiche une boîte de dialogue d'alerte comportant un simple mot :

```

Case of
:(vResult=1) //Tester si le chiffre est 1
  ALERT("One.") Case of
:(vResult=1) //Tester si le chiffre est 1
  ALERT("One.") Case of
:(vResult=1) //Tester si le chiffre est 1
  ALERT("One.") //Si le chiffre est 1, afficher une alerte
:(vResult=2) //Tester si le chiffre est 2
  ALERT("Two.") //Si le chiffre est 2, afficher une alerte
:(vResult=3) //Tester si le chiffre est 3
  ALERT("Three.") //Si le chiffre est 3, afficher une alerte
Else //Si le chiffre n'est pas 1, 2 ou 3, afficher une alerte
  ALERT("It was not one, two, or three.")
//déclaration(s)
End case //Si le chiffre est 2, afficher une alerte
:(vResult=3) //Tester si le chiffre est 3
  ALERT("Three.") //Si le chiffre est 3, afficher une alerte
Else //Si le chiffre n'est pas 1, 2 ou 3, afficher une alerte
  ALERT("It was not one, two, or three.")
//déclaration(s)
End case //Si le chiffre est 2, afficher une alerte
:(vResult=3) //Tester si le chiffre est 3
  ALERT("Three.") //Si le chiffre est 3, afficher une alerte
Else //Si le chiffre n'est pas 1, 2 ou 3, afficher une alerte
  ALERT("It was not one, two, or three.")
//déclaration(s)
End case

```

A titre de comparaison, voici la version avec `If...Else...End if` de la même méthode :

```

If(vResult=1) //Tester si le chiffre est 1
    ALERT("One.") If(vResult=1) //Tester si le chiffre est 1
        ALERT("One.") If(vResult=1) //Tester si le chiffre est 1
            ALERT("One.") If(vResult=1) //Tester si le chiffre est 1
                ALERT("One.") //Si le chiffre est 1, afficher une alerte
Else
    If(vResult=2) //Tester si le chiffre est 2
        ALERT("Two.") //Si le chiffre est 2, afficher une alerte
Else
    If(vResult=3) //Tester si le chiffre est 3
        ALERT("Three.") //Si le chiffre est 3, afficher une alerte
Else //Si le chiffre n'est pas 1, 2 ou 3, afficher une alerte
    ALERT("It was not one, two, or three.")
    End if
End if
End if //Si le chiffre est 2, afficher une alerte
Else
If(vResult=3) //Tester si le chiffre est 3
    ALERT("Three.") //Si le chiffre est 3, afficher une alerte
Else //Si le chiffre n'est pas 1, 2 ou 3, afficher une alerte
    ALERT("It was not one, two, or three.")
    End if
End if
End if //Si le chiffre est 2, afficher une alerte
Else
If(vResult=3) //Tester si le chiffre est 3
    ALERT("Three.") //Si le chiffre est 3, afficher une alerte
Else //Si le chiffre n'est pas 1, 2 ou 3, afficher une alerte
    ALERT("It was not one, two, or three.")
    End if
End if
End if //Si le chiffre est 2, afficher une alerte
Else
If(vResult=3) //Tester si le chiffre est 3
    ALERT("Three.") //Si le chiffre est 3, afficher une alerte
Else //Si le chiffre n'est pas 1, 2 ou 3, afficher une alerte
    ALERT("It was not one, two, or three.")
    End if
End if
End if

```

Rappelez-vous qu'avec une structure de type **Case of...Else...End case**, seul le premier cas TRUE rencontré est exécuté. Même si d'autres cas sont TRUE, seules les instructions suivant le premier cas TRUE seront prises en compte.

Par conséquent, lorsque vous testez dans la même méthode des cas simples et des cas complexes, vous devez placer les cas complexes avant les cas simples, sinon ils ne seront jamais exécutés. Par exemple, si vous souhaitez traiter le cas simple (vResult=1) et le cas complexe (vResult=1) & (vCondition#2) et que vous structurez la méthode de la manière suivante :

```

Case of
    :(vResult=1) //Tester si le chiffre est 1
        ALERT("One.") //Si le chiffre est 1, afficher une alerte
    :(vResult=2) //Tester si le chiffre est 2
        ALERT("Two.") //Si le chiffre est 2, afficher une alerte
    :(vResult=3) //Tester si le chiffre est 3
        ALERT("Three.") //Si le chiffre est 3, afficher une alerte
    Else //Si le chiffre n'est pas 1, 2 ou 3, afficher une alerte
        ALERT("It was not one, two, or three.")
End case

```

... les instructions associées au cas complexe ne seront jamais exécutées. En effet, pour que ce cas soit TRUE, ses deux conditions booléennes doivent l'être. Or, la première condition est celle du cas simple situé précédemment. Lorsqu'elle

est TRUE, le cas simple est exécuté et 4D sort de la structure conditionnelle, sans évaluer le cas complexe. Pour que ce type de méthode fonctionne, vous devez écrire :

```
If(vResult=1) //Tester si le chiffre est 1
    ALERT("One.") //Si le chiffre est 1, afficher une alerte
Else
    If(vResult=2) //Tester si le chiffre est 2
        ALERT("Two.") //Si le chiffre est 2, afficher une alerte
    Else
        If(vResult=3) //Tester si le chiffre est 3
            ALERT("Three.") //Si le chiffre est 3, afficher une alerte
        Else //Si le chiffre n'est pas 1, 2 ou 3, afficher une alerte
            ALERT("It was not one, two, or three.")
        End if
    End if
End if
```

Astuce : Il n'est pas obligatoire que des instructions soient exécutées dans toutes les alternatives. Lorsque vous développez un algorithme, ou lorsque vous poursuivez un but précis, rien ne vous empêche d'écrire :

```
Case of
    :(Expression_booléenne)
    :(Expression_booléenne)
        instruction(s)
    ...
    :(Expression_booléenne)
        instruction(s)
    Else
        instruction(s)
End case
```

ou :

```
Case of
    :(Expression_booléenne)
    :(Expression_booléenne)
    ...
    :(Expression_booléenne)
        instruction(s)
    Else
        instruction(s)
End case
```

ou :

```
Case of
    Else
        instruction(s)
End case
```

Structures répétitives (ou "boucles")

Les structures en boucle répètent une séquence d'instructions jusqu'à ce qu'une condition soit remplie ou qu'un certain nombre de fois est atteint.

While...End while

La syntaxe de la structure répétitive (ou boucle) `While...End while` est la suivante :

```
While(Boolean_Expression)
    statement(s)
    {break}
    {continue}
End while
```

Une boucle `While...End while` exécute les instructions comprises entre `While` et `End while` aussi longtemps que l'expression booléenne est TRUE. Elle teste l'expression booléenne initiale et n'entre pas dans la boucle (et donc n'exécute aucune instruction) si l'expression est à FALSE.

Les instructions `break` et `continue` sont [déesrites ci-dessous](#).

Il est utile d'initialiser la valeur testée dans l'expression booléenne juste avant d'entrer dans la boucle `While...End while`. Initialiser la valeur signifie lui affecter un contenu approprié, généralement pour que l'expression booléenne soit TRUE et que le programme entre dans la boucle.

La valeur de l'expression booléenne doit pouvoir être modifiée par un élément situé à l'intérieur de la boucle, sinon elle s'exécutera indéfiniment. La boucle suivante est sans fin car `NeverStop` est toujours TRUE :

```
NeverStop:=True
While(NeverStop)
End while
```

Si vous vous retrouvez dans une telle situation (où une méthode s'exécute de manière incontrôlée), vous pouvez utiliser les fonctions de débogage de 4D et remonter à la source du problème. Pour plus d'informations sur ce point, reportez-vous à la section [Débogueur](#).

Exemple

```
CONFIRM("Add a new record?") //L'utilisateur souhaite-t-il ajouter un enregistrement ? //L'utilisateur
While(OK=1) // Tant que l'utilisateur accepte
    ADD RECORD([aTable]) // Ajouter un nouvel enregistrement
End while // Une boucle While se termine toujours par End while
```

Dans cet exemple, la valeur de la variable système `OK` est définie par la commande `CONFIRM` avant que le programme n'entre dans la boucle. Si l'utilisateur clique sur le bouton OK dans la boîte de dialogue de confirmation, la variable `OK` prend la valeur 1 et la boucle est exécutée. Dans le cas contraire, la variable `OK` prend la valeur 0 et la boucle est ignorée. Une fois que le programme entre dans la boucle, la commande `ADD RECORD` permet de continuer à l'exécuter car elle met la variable système `OK` à 1 lorsque l'utilisateur sauvegarde l'enregistrement. Lorsque l'utilisateur annule (ne valide pas) le dernier enregistrement, la variable système `OK` prend la valeur 0 et la boucle s'arrête.

Repeat...Until

La syntaxe de la structure répétitive (ou boucle) `Repeat...Until` est la suivante :

```
Repeat
    statement(s)
    {break}
    {continue}
Until(Boolean_Expression)
```

La boucle `Repeat...Until` est semblable à la boucle `While...End while`, à la différence qu'elle teste la valeur de l'expression booléenne après l'exécution de la boucle et non avant. Ainsi, la boucle est toujours exécutée au moins une fois, tandis que si l'expression booléenne est initialement à Faux, la boucle `While...End while` ne s'exécute pas du tout.

L'autre particularité de la boucle `Repeat...Until` est qu'elle se poursuit jusqu'à ce que l'expression booléenne soit à TRUE.

Les instructions `break` et `continue` sont [décrites ci-dessous](#).

Exemple

Comparez l'exemple suivant avec celui de la boucle `While...End while`. Vous constatez qu'il n'est pas nécessaire d'initialiser l'expression booléenne — il n'y a pas de commande `CONFIRM` pour initialiser la variable `OK`.

```
Repeat
    ADD RECORD([aTable])
Until(OK=0)
```

For...End for

La syntaxe de la structure répétitive `For...End for` est la suivante :

```
For(Counter_Variable;Start_Expression;End_Expression{;Increment_Expression})
    statement(s)
    {break}
    {continue}
End for
```

La structure `For...End for` est une boucle contrôlée par un compteur :

- La variable compteur `Counter_Variable` est une variable numérique (Réel ou Entier long) initialisée par `For...End for` à la valeur spécifiée par `Start_Expression`.
- La variable `Variable_Compteur` est incrémentée de la valeur spécifiée par le paramètre optionnel `Increment_Expression` à chaque fois que la boucle est exécutée. Si vous ne passez pas de valeur dans `Increment_Expression`, la variable compteur est incrémentée par défaut de un (1).
- Lorsque le compteur atteint la valeur définie par `End_Expression`, la boucle s'arrête.

Important : Les expressions numériques `Start_Expression`, `End_Expression` et `Increment_Expression` sont évaluées une seule fois, au début de la boucle. Si ces expressions sont des variables, leur modification depuis l'intérieur de la boucle n'affectera pas l'exécution de la boucle.

Astuce : En revanche, vous pouvez, si vous le souhaitez, modifier la valeur de la variable `Counter_Variable` depuis l'intérieur de la boucle et cela affectera l'exécution de la boucle.

- Généralement, `Start_Expression` est inférieure à `End_Expression`.
- Si les deux expressions sont égales, la boucle ne sera exécutée qu'une fois.
- Si `Start_Expression` est supérieure à `End_Expression`, la boucle ne s'exécutera pas du tout, à moins que vous ne spécifiez une `Increment_Expression` négative. Reportez-vous ci-dessous au paragraphe décrivant ce point.

Les instructions `break` et `continue` sont décrites ci-dessous.

Exemples élémentaires

- La boucle suivante s'exécute 100 fois :

```
For(vCounter;1;100)
//Faire quelque chose
End for
```

- L'exemple suivant permet de traiter tous les éléments du tableau `anArray` :

```
For($vlElem;1;Size of array(anArray))
//Faire quelque chose avec l'élément
anArray[$vlElem]:=...
End for
```

- L'exemple suivant permet d'examiner chaque caractère du texte `vtSomeText` :

```
For($vlChar;1;Length(vtSomeText))
//Faire quelque chose avec le caractère si c'est une tabulation
If(Character code(vtSomeText[[ $vlChar ]])=Tab)
//...
End if
End for
```

- L'exemple suivant permet de traiter tous les enregistrements de la sélection de la table `[aTable]` :

```
FIRST RECORD([aTable])
For($vlRecord;1;Records in selection([aTable]))
//Faire quelque chose avec chaque enregistrement
SEND RECORD([aTable])
//...
// Passer à l'enregistrement suivant
NEXT RECORD([aTable])
End for
```

La plupart des structures `For...End for` que vous écrirez dans vos projets ressembleront à celles présentées ci-dessus.

Décrémente la variable Compteur

Dans certains cas, vous pouvez souhaiter disposer d'une boucle dont la valeur de la variable compteur décroît au lieu de croître. Pour cela, *Start_Expression* doit être supérieure à *End_Expression* et *Increment_Expression* doit être négative. Les exemples suivants effectuent les mêmes tâches que les précédents, mais en sens inverse :

- La boucle suivante s'exécute 100 fois :

```
For(vCounter;100;1;-1)
//Faire quelque chose
End for
```

- L'exemple suivant permet de traiter tous les éléments du tableau `anArray` :

```

For($vlElem;Size of array(anArray);1;-1)
  //Faire quelque chose avec l'élément
  anArray[$vlElem]:=...
End for

```

7. L'exemple suivant permet d'examiner chaque caractère du texte vtSomeText :

```

For($vlChar;Length(vtSomeText);1;-1)
  //Faire quelque chose avec le caractère si c'est une tabulation
  If(Character code(vtSomeText[$vlChar])=Tab)
  //...
  End if
End for

```

8. L'exemple suivant permet de traiter tous les enregistrements de la sélection de la table [aTable] :

```

LAST RECORD([aTable])
For($vlRecord;Records in selection([aTable]);1;-1)
  //Faire quelque chose avec chaque enregistrement
  SEND RECORD([aTable])
  //...
  //Passer à l'enregistrement précédent
  PREVIOUS RECORD([aTable])
End for

```

Incrementer la variable compteur de plus de 1

Si vous le souhaitez, vous pouvez passer dans *Increment_Expression* une valeur (positive ou négative) dont la valeur absolue est supérieure à un.

9. La boucle suivante ne traite que les éléments pairs du tableau anArray :

```

For($vlElem;2;Size of array(anArray);2)
  //Faire quelque chose avec l'élément 2,4...2n
  anArray[$vlElem]:=...
End for

```

Comparaison des structures répétitives

Revenons au premier exemple `For...End for`. La boucle suivante s'exécute 100 fois :

```

For(vCounter;1;100)
  //Faire quelque chose
End for

```

Il est intéressant d'examiner la manière dont les boucles `While...End while` et `Repeat...Until` effectuent la même action. Voici la boucle `While...End while` équivalente :

```

$ i :=1 // Initialisation du compteur
While ($i<=100) // Boucle 100 fois
  // Faire quelque chose
  $ i :=$ i +1 // Il faut incrémenter le compteur
End while

```

Voici la boucle `Repeat...Until` équivalente :

```
$i :=1 // Initialisation du compteur
Repeat
    // Faire quelque chose
    $i :=$i +1 // Il faut incrémenter le compteur
Until($i=100) // Boucle 100 fois
```

Astuce : La boucle `For...End for` est généralement plus rapide que les boucles `While...End while` et `Repeat...Until` car 4D teste la condition en interne pour chaque cycle de la boucle et incrémente lui-même le compteur. Par conséquent, nous vous conseillons de préférer à chaque fois que c'est possible la structure `For...End for`.

Optimiser l'exécution de For...End for

Vous pouvez utiliser comme compteur une variable interprocess, process ou locale, et lui attribuer le type Réel, Entier ou Entier long. Pour des boucles longues, et particulièrement en mode compilé, nous vous conseillons d'employer des variables locales de type Entier long.

10. Voici un exemple :

```
C_LONGINT($vlCounter) // Utilisons une variable locale de type Entier long
For($vlCounter;1;10000)
    // Faire quelque chose
End for
```

Structures For...End for emboîtées

Vous pouvez emboîter autant de structures répétitives que vous voulez (dans les limites du raisonnable). Cela s'applique aux structures de type `For...End for`. Il y a dans ce cas une erreur courante à éviter : assurez-vous d'utiliser une variable compteur différente par structure de boucle.

Voici deux exemples :

1. L'exemple suivant permet de traiter tous les éléments d'un tableau à deux dimensions :

```
For($vlElem;1;Size of array(anArray))
    //...
    // Faire quelque chose avec la ligne
    // ...
    For($vlSubElem;1;Size of array(anArray{$vlElem}))
        //Faire quelque chose avec l'élément
        anArray{$vlElem}{$vlSubElem}:=...
    End for
End for
```

2. L'exemple suivant construit un tableau de pointeurs vers tous les champs de type Date présents dans la base :

```

ARRAY POINTER($apDateFields;0)
$vlElem:=0
For($vlTable;1;Get last table number)
  If(Is table number valid($vlTable))
    For($vlField;1;Get last field number($vlTable))
      If(Is field number valid($vlTable;$vlField))
        $vpField:=Field($vlTable;$vlField)
        If(Type($vpField->)=Is date)
          $vlElem:=$vlElem+1
          INSERT IN ARRAY($apDateFields;$vlElem)
          $apDateFields{$vlElem}:=$vpField
        End if
      End if
    End for
  End if
End for

```

For each...End for each

La syntaxe de la structure répétitive (ou boucle) `For each...End for each` est la suivante :

```

For each(Current_Item;Expression{;begin{;end}}){Until|While(Boolean_Expression)}
  statement(s)
  {break}
  {continue}
End for each

```

La structure `For each...End for each` exécute le cycle d'instructions définies pour chaque *Elément_courant* de *Expression*. Le type de *Elément_courant* dépend du type de *Expression*. La boucle `For each...End for each` peut itérer parmi trois types d'*Expression* :

- collections : boucle sur chaque élément de la collection,
- entity selections : boucle sur chaque entity,
- objets : boucle sur chaque propriété d'objet.

Le tableau suivant compare les trois types de `Pour chaque...Fin de chaque` :

	Boucle sur collections	Boucle sur entity selections	Boucle sur objets
Type Elément_courant	Variable du même type que les éléments de la collection	Entity	Variable texte
Types d'expressions	Collection (avec des éléments du même type)	Entity selection	Object
Nombre de boucles (par défaut)	Nombre d'éléments de la collection	Nombre d'entités dans la sélection	Nombre de propriétés d'objets
Prise en charge de Paramètres début / fin	Oui	Oui	Non

- Le nombre de boucles est évalué au démarrage et ne changera pas en cours de traitement. L'ajout ou la suppression d'éléments pendant la boucle est donc déconseillé car il pourra en résulter une redondance ou un manque d'itérations.
- Par défaut, les *instructions* incluses sont exécutées pour chaque valeur de *Expression*. Il est toutefois possible de sortir de la boucle en testant une condition soit au début de chaque itération (`While`) ou à la fin de chaque itération (`Until`).
- Les paramètres optionnels *début* et *fin* peuvent être utilisés avec les collections et les entity selections afin de définir des bornes pour la boucle.

- La boucle `For each...End for each` peut être utilisée sur une collection partagée ou un objet partagé. Si vous souhaitez modifier un ou plusieurs éléments des propriétés d'objets ou de la collection dans le code, vous devez utiliser les mots-clés `Use...End use`. Vous pouvez, si vous le souhaitez, appeler les mots-clés `Use...End use` :
 - avant de saisir la boucle, si les éléments doivent être modifiés ensemble pour des raisons d'intégrité, ou bien
 - dans la boucle, lorsque quelques éléments/propriétés seulement doivent être modifiés et qu'aucune gestion de l'intégrité n'est requise.

Les instructions `break` et `continue` sont [décris ci-dessous](#).

Boucle sur collections

Lorsque `For each...End for each` est utilisée avec une *Expression* de type *Collection*, le paramètre *Elément_courant* est une variable du même type que les éléments de la collection. Par défaut, le nombre de boucles est basé sur le nombre d'éléments de la collection.

La collection doit contenir uniquement des éléments du même type. Dans le cas contraire, une erreur sera retournée dès que la première valeur de type différent sera assignée à la variable *Elément_courant*.

A chaque itération de la boucle, la variable *Elément_courant* reçoit automatiquement l'élément correspondant de la collection. Vous devez tenir compte des points suivants :

- La variable *Elément_courant* doit être du même type que les éléments de la collection. Si un seul élément de la collection n'est pas du même type que la variable, une erreur est générée et la boucle s'arrête.
- Si la variable *Elément_courant* est de type objet ou collection (i.e. Si un seul élément de la collection n'est pas du même type que la variable, une erreur est générée et la boucle s'arrête).
- Si la collection contient des éléments de valeur `Null`, une erreur sera générée si le type de la variable *Elément_courant* ne prend pas en charge la valeur `Null` (comme par exemple les variables entier long).

Exemple

Vous souhaitez calculer quelques statistiques sur une collection de nombres :

```
C_COLLECTION($nums)
$nums:=New collection(10;5001;6665;33;1;42;7850)
C_LONGINT($item;$vEven;$vOdd;$vUnder;$vOver)
For each($item;$nums)
  If($item%2=0)
    $vEven:=$vEven+1
  Else
    $vOdd:=$vOdd+1
  End if
  Case of
    :($item<5000)
      $vUnder:=$vUnder+1
    :($item>6000)
      $vOver:=$vOver+1
  End case
End for each
//$vEven=3, $vOdd=4
//$vUnder=4,$vOver=2
```

Boucle sur entity selections

Lorsque `For each...End for each` est utilisée avec une *Expression* de type *Entity selection*, le paramètre *Elément_courant* contient l'entity en cours de traitement.

Le nombre de boucles est basé sur le nombre d'entities présentes dans l'*entity selection*. A chaque itération de la boucle, le paramètre *Elément_courant* reçoit automatiquement l'*entity* qui est en cours de traitement.

Note : Si l'*entity selection* contient une entity qui a été supprimée entre-temps par un autre process, elle est automatiquement ignorée durant la boucle.

N'oubliez pas que toute modification effectuée sur l'entity en cours de traitement doit être explicitement sauvegardée (si nécessaire) à l'aide de la méthode `entity.save()`.

Exemple

Vous souhaitez augmenter le salaire de tous les employés britanniques dans une entity selection :

```
C_OBJECT(emp)
For each(emp;ds.Employees.query("country='UK'"))
    emp.salary:=emp.salary*1,03
    emp.save()
End for each
```

Boucles sur des propriétés d'objets

Lorsque `For each...End for each` est utilisée avec une *Expression* de type Objet, le paramètre *Elément_courant* est une variable texte qui reçoit automatiquement le nom de la propriété en cours de traitement.

Les propriétés de l'objet sont itérées en fonction de leur ordre de création. Pendant la boucle, il est possible d'ajouter ou de supprimer des propriétés dans l'objet, sans pour autant modifier le nombre de boucles qui reste basé sur le nombre de propriétés initial de l'objet.

Exemple

Vous souhaitez passer en majuscules les propriétés contenant des noms dans l'objet suivant :

```
{
    "firstname": "gregory",
    "lastname": "badikora",
    "age": 20
}
```

Vous pouvez écrire :

```
For each(property;v0bject)
    If(Value type(v0bject[property])=Is text)
        v0bject[property]:=Uppercase(v0bject[property])
    End if
End for each
```

```
{
    "firstname": "GREGORY",
    "lastname": "BADIKORA",
    "age": 20
}
```

Paramètres début / fin

Vous pouvez définir des bornes pour l'itération à l'aide des paramètres optionnels *début* et *fin*.

Note : Les paramètres *début* et *fin* sont utilisables uniquement avec les boucles sur des collections et des entity selections (ils sont ignorés avec les boucles sur des propriétés d'objets).

- Dans le paramètre *début*, passez la position de l'élément de *Expression* auquel démarrer l'itération (*début* est inclus).
- Dans le paramètre *fin*, vous pouvez passer la position de l'élément de *Expression* auquel stopper l'itération (*fin* est exclus).

Si *fin* est omis ou si *fin* est plus grand que le nombre d'éléments de *Expression*, les éléments sont itérés depuis *début* jusqu'au dernier inclus. Si les paramètres *début* et *fin* sont des valeurs positives, ils représentent des positions d'éléments dans *Expression*. Si *begin* est une valeur négative, elle est recalculée comme `begin:=begin+Taille expression` (elle est considérée comme un décalage à partir de la fin de *Expression*). Si la valeur calculée est négative, *begin* prend la valeur 0. Note : Même si *début* est une valeur négative, l'itération est toujours effectuée dans le même ordre. Si *fin* est une valeur négative, elle est recalculée comme `fin:=fin+Taille expression`

Par exemple :

- une collection contient 10 éléments (numérotés de 0 à 9)
- *début*=-4 > *début*=-4+10=6 > l'itération démarre au 6e élément (numéro 5)
- *fin*=-2 > *fin*=-2+10=8 > l'itération stoppe avant le 8e élément (numéro 7), i.e.

Exemple

```
C_COLLECTION($col;$col2)
$col:=New collection("a";"b";"c";"d";"e")
$col2:=New collection(1;2;3)
C_TEXT($item)
For each($item;$col;0;3)
    $col2.push($item)
End for each
//$col2=[1,2,3,"a","b","c"]
For each($item;$col;-2;-1)
    $col2.push($item)
End for each
//$col2=[1,2,3,"a","b","c","d"]
```

Conditions Until et While

Vous pouvez contrôler l'exécution de `For each...End for each` en ajoutant une condition `Jusque` ou `Tant que` à la boucle. Lorsqu'une instruction `Until(condition)` est associée à la boucle, l'itération stoppe dès que la condition est évaluée à `True`, tandis que dans le cas d'une instruction `While(condition)`, l'itération stoppe dès que la condition est évaluée à `False`.

Vous pouvez passer un mot-clé ou l'autre en fonction de vos besoins :

- La condition `Until` est testée à la fin de chaque itération, donc si *Expression* n'est ni vide ni Null, la boucle sera exécutée au moins une fois.
- La condition `While` est testée au début de chaque itération, donc en fonction du résultat de la condition, la boucle peut ne pas être exécutée du tout.

Exemple

```
$colNum:=New collection(1;2;3;4;5;6;7;8;9;10)

$total:=0
For each($num;$colNum)While($total<30) //testé au début
    $total:=$total+$num
End for each
ALERT(String($total)) //total = 36 (1+2+3+4+5+6+7+8)

$total:=1000
For each($num;$colNum)Until($total>30) //testé à la fin
    $total:=$total+$num
End for each
ALERT(String($total)) //total = 1001 (1000+1)
```

break et continue

Toutes les structures de boucles ci-dessus prennent en charge les instructions `break` et `continue`. Ces instructions vous donnent plus de contrôle sur les boucles en vous permettant de sortir de la boucle et de contourner, à tout moment, l'itération en cours.

break

L'instruction `break` met fin à la boucle qui la contient. Le contrôle du programme passe à l'instruction située immédiatement après le corps de la boucle.

Si l'instruction `break` se trouve à l'intérieur d'une [boucle imbriquée](#) (boucle dans une autre boucle), l'instruction `break` mettra fin à la boucle la plus interne.

Exemple

```
For (vCounter;1;100)
    If ($tab{vCounter}=="") //si une condition devient vraie
        break //fin de la boucle For
    End if
End for
```

continue

L'instruction `continue` met fin à l'exécution des instructions de l'itération de la boucle courante, et poursuit l'exécution de la boucle à l'itération suivante.

```
var $text : Text
For ($i; 0; 9)
    If ($i=3)
        continue //traite directement l'itération suivante
    End if
    $text:=$text+String($i)
End for
// $text="012456789"
```

Gestion des erreurs

Le traitement des erreurs consiste à anticiper les erreurs pouvant survenir dans votre application et à y répondre. 4D fournit un support complet pour la détection et la signalisation des erreurs lors de l'exécution, ainsi que pour l'analyse de leurs conditions.

La gestion des erreurs répond à deux besoins principaux :

- rechercher et corriger les éventuels bugs et erreurs dans votre code pendant la phase de développement,
- détecter et récupérer des erreurs inattendues dans les applications déployées; vous pouvez notamment remplacer les boîtes de dialogue d'erreur système (disque plein, fichier manquant, etc.) par votre propre interface.

Il est fortement recommandé d'installer une méthode de gestion des erreurs sur 4D Server, pour tout le code exécuté sur le serveur. Cette méthode éviterait d'afficher des boîtes de dialogue inattendues sur le serveur et pourrait consigner les erreurs dans un fichier consacré en vue d'analyses ultérieures.

Erreur ou statut

De nombreuses fonctions de classe 4D, telles que `entity.save()` ou `transporter.send()`, retournent un objet `status`. Cet objet permet de stocker les erreurs "prévisibles" dans le contexte d'exécution, telles qu'un mot de passe invalide, une entité verrouillée, etc., qui ne stoppe pas l'exécution du programme. Cette catégorie d'erreurs peut être gérée par du code habituel.

D'autres erreurs "imprévisibles" peuvent inclure une erreur en écriture sur le disque, une panne de réseau ou toute interruption inattendue. Cette catégorie d'erreurs génère des exceptions et doit être traitée par une méthode de gestion des erreurs.

Installer une méthode de gestion des erreurs

Dans 4D, toutes les erreurs peuvent être capturées et traitées dans une méthode projet spécifique, la méthode de gestion des erreurs (ou méthode de capture d'erreurs).

Cette méthode projet est installée pour le process en cours et sera automatiquement appelée pour toute erreur survenant dans le process, en mode interprété ou compilé. Pour *installer* cette méthode projet, il vous suffit d'appeler la commande `APPELER SUR ERREUR` avec le nom de la méthode projet en paramètre. Par exemple :

```
APPELER SUR ERREUR("IO_ERRORS") //Installe la méthode de gestion des erreurs
```

Pour ne plus détecter d'erreurs et redonner le contrôle à 4D, appelez la méthode `ON ERR CALL` à l'aide d'une chaîne vide :

```
ON ERR CALL("") //redonne le contrôle à 4D
```

La commande `Method called on error` vous permet de connaître le nom de la méthode installée par `ON ERR CALL` pour le process courant. Cela est particulièrement utile dans le contexte du code générique car il vous permet de modifier temporairement puis de restaurer la méthode de capture d'erreur :

```

$methCurrent:=Method called on error
ON ERR CALL("NewMethod")
//Si le document ne peut pas être ouvert, une erreur est générée
$ref:=Open document("MyDocument")
//Rétablissement de la méthode précédente
ON ERR CALL($methCurrent)

```

Portée et composants

Vous pouvez définir une seule méthode d'erreur pour l'ensemble de l'application ou différentes méthodes par module d'application. Cependant, une seule méthode peut être installée par process.

Une méthode de gestion des erreurs installée par la commande `APPELER SUR ERREUR` s'applique uniquement à l'application en cours d'exécution. En cas d'erreur générée par un composant, la méthode `APPELER SUR ERREUR` de l'application hôte n'est pas appelée, et inversement.

Gérer les erreurs dans une méthode

Dans la méthode d'erreur personnalisée, vous pouvez accéder à plusieurs informations qui vous aideront à identifier l'erreur :

- Variables système (*) :

 - `Error` (entier long) : Code d'erreur
 - `Error method` (texte) : nom de la méthode ayant engendré l'erreur
 - `Error line` (entier long) : Numéro de ligne de la méthode ayant généré l'erreur
 - `Error formula` (texte) : formule du code 4D (texte brut) à l'origine de l'erreur.

() 4D conserve automatiquement le nombre de variables appelées variables système, qui répondent à différents besoins. Consultez le manuel Language de 4D.*

- La commande `GET LAST ERROR STACK` qui retourne les informations sur la pile d'erreur courant de l'application 4D.
- la commande `Get call chain` qui retourne une collection d'objets décrivant chaque étape de la chaîne d'appel de la méthode dans le process courant.

Exemple

Voici un système de gestion des erreurs simple :

```

//installer la méthode de gestion d'erreur
ON ERR CALL("errorMethod")
//... exécuter le code
ON ERR CALL("") //redonner le contrôle à 4D

```

```

// méthode projet errorMethod
If(Error#1006) //ceci n'est pas une interruption générée par l'utilisateur
  ALERT("L'erreur "+String(Error)+" s'est produite". Le code en question est : \""+Error formula+"\")"
End if

```

Utiliser une méthode de gestion des erreurs vide

Si vous souhaitez cacher la boîte de dialogue d'erreur standard, vous pouvez installer une méthode de gestion d'erreurs vide. La variable système `Error` peut être testée dans n'importe quelle méthode, c'est-à-dire en dehors de la méthode de gestion d'erreurs :

```
ON ERR CALL("emptyMethod") //emptyMethod existe mais elle est vide
$doc:=Open document( "myFile.txt")
If (Error=-43)
    ALERT("File not found.")
End if
ON ERR CALL.("")
End if
ON ERR CALL.("")
```

Modes interprété et compilé

Les applications 4D fonctionnent en mode interprété ou en mode compilé :

- En mode interprété, les déclarations sont lues et traduites en langage machine lorsqu'elles sont exécutées. Vous pouvez ajouter ou modifier le code là où vous le souhaitez, l'application se met à jour automatiquement.
- En mode compilé, toutes les méthodes sont lues et traduites une seule fois, lors de la compilation. Par la suite, l'application contient uniquement des instructions au niveau de l'assemblage, il n'est donc plus possible de modifier le code.

Les avantages procurés par le compilateur sont les suivants :

- Vitesse : votre application s'exécute de 3 à 1000 fois plus vite.
- Vérification du code : la cohérence interne du code de votre application est entièrement contrôlée. Les conflits de logique et de syntaxe sont détectés.
- Protection : une fois votre application compilée, vous pouvez en supprimer le code interprété. Alors, l'application compilée dispose des mêmes fonctionnalités que la base originale, à la différence près que la structure et les méthodes ne peuvent plus être visualisées ni modifiées délibérément ou par inadvertance.
- Application indépendantes "double-cliquables" : une application compilée peut également être transformée en application indépendante (sous Windows, des fichiers ".EXE") comportant sa propre icône.
- Exécution en mode préemptif : seul le code compilé peut être exécuté dans un process préemptif.

Différences entre le code interprété et le code compilé

Bien que l'application fonctionnera de la même manière en modes interprété et compilé, il est important de connaître les différences que l'on peut rencontrer pendant la saisie du code qui sera compilé. L'interpréteur de 4D est généralement plus souple que le compilateur.

Compilé	Interprété
You ne devez pas avoir une méthode qui aurait le même nom qu'une variable.	Aucune erreur n'est générée, mais la méthode est prioritaire
Toutes les variables doivent être typées, soit via une directive de compilateur (ex : C_ENTIER LONG), soit via le compilateur au moment de la compilation.	Les variables peuvent être typées à la volée (non recommandé)
You ne pouvez pas modifier le type d'une variable ou d'un tableau.	La modification du type d'une variable ou d'un tableau est possible (non recommandé)
You ne pouvez pas convertir un tableau simple en tableau à deux dimensions, et vice-versa.	Possible
Bien que le compilateur déduise le type des variables si nécessaire, il est conseillé de déclarer le type des variables à l'aide des directives de compilation lorsque le type de données est ambigu, en particulier dans un formulaire.	
La fonction <code>Indefinie</code> retournera toujours Faux. Les variables sont toujours définies.	
Si vous avez coché la propriété "Peut être exécutée dans un process préemptif" pour la méthode, le code ne doit pas appeler de commandes thread-unsafe ou d'autres méthodes thread-unsafe.	Les propriétés du process préemptif sont ignorées
La commande <code>APPELER 4D</code> est nécessaire pour appeler des boucles spécifiques	Il est toujours possible d'interrompre 4D

Utiliser les directives du compilateur avec l'interpréteur

Les directives de compilateur ne sont pas requises pour les applications non compilées. L'interpréteur type automatiquement chaque variable selon son utilisation dans la déclaration, et une variable peut être retypée librement dans le projet d'application.

Grâce à cet aspect flexible, il est possible qu'une application agisse différemment en modes interprété et compilé.

Par exemple, si vous écrivez :

```
C_ENTIER LONG(MyInt)
```

et ailleurs dans l'application, vous écrivez :

```
MyInt:=3.1416
```

Dans cet exemple, `MyInt` se voit assigner la même valeur (3) dans les modes interprété et compilé, étant donné que la directive du compilateur est interprétée *avant* la déclaration d'affectation.

L'interpréteur 4D utilise des directives de compilateur pour typer les variables. Lorsque l'interpréteur rencontre une directive de compilateur, il type la variable en fonction de la directive. Si une déclaration ultérieure tente d'affecter une valeur incorrecte (ex : affecter une valeur alphanumérique à une variable numérique), l'affectation n'aura pas lieu et générera une erreur.

L'ordre d'apparition des deux déclarations importe peu au compilateur, car il scanne d'abord le projet dans son intégralité pour les directives du compilateur. En revanche, l'interpréteur n'est pas systématique. Il interprète les déclarations dans leur ordre d'exécution. Cet ordre peut évidemment changer d'une session à l'autre, en fonction de ce que fait l'utilisateur. C'est pourquoi il est important de concevoir votre projet afin que vos directives de compilateur s'exécutent avant n'importe quelle déclaration contenant des variables déclarées.

Utiliser des pointeurs pour éviter les retypages

Il n'est pas possible de retyper une variable. Il est, en revanche, tout à fait possible de faire pointer un pointeur successivement sur des variables de type différent. Par exemple, le code suivant s'applique aussi bien dans le mode interprété que dans le mode compilé :

```
C_POINTER($p)
C_TEXT($name)
C_LONGINT($age)

$name:="Smith"
$age:=50

$p:==>$name //texte cible pour le pointeur
$p->:="Wesson" //assigne une valeur texte

$p:==>$age
// nouvelle cible de type différent pour le pointeur
$p->:=55 //assigne une valeur numérique
```

Imaginez une fonction qui retourne la longueur (nombre de caractères) de valeurs de tout type.

```
// Calc_Length (combien de caractères)
// $1 = pointeur vers un type de variable flexible, numérique, text, heure, booléen

C_POINTER($1)
C_TEXT($result)
C_LONGINT($0)
$result:=String($1->)
$0:=Length($result)
```

La méthode peut alors être appelée :

```
$var1:="my text"
$var2:=5.3
$var3:=?10:02:24?
$var4:=True

$vLength:=Calc_Length(->$var1)+Calc_Length(->$var2)+Calc_Length(->$var3)+Calc_Length(->$var4)

ALERT("Total length: "+String($vLength))
```

Composants

Un composant 4D est un ensemble de méthodes et de formulaires 4D représentant une ou plusieurs fonctionnalité(s), qu'il est possible d'installer et d'utiliser dans différents projets. Par exemple, le [composant 4D SVG](#) ajoute des commandes avancées et un moteur de rendu intégré qui peut être utilisé pour afficher des fichiers SVG.

Où se trouvent les composants ?

Plusieurs composants sont [préinstallés dans l'environnement de développement 4D](#), mais de nombreux composants 4D de la communauté 4D [sont disponibles sur GitHub](#). De plus, vous pouvez [développer vos propres composants 4D](#).

Installation des composants

Pour installer un composant, il suffit de copier les fichiers du composant dans le dossier [Components du projet](#). Vous pouvez utiliser des alias ou des raccourcis.

Un projet hôte fonctionnant en mode interprété peut utiliser des composants interprétés ou compilés. Un projet hôte fonctionnant en mode compilé ne peut pas utiliser de composants interprétés. Dans ce cas, seuls les composants compilés peuvent être utilisés.

Utilisation des composants

Les formulaires et méthodes composant peuvent être utilisés comme éléments standard de votre développement 4D.

Lorsqu'un composant installé contient des méthodes, celles-ci apparaissent dans le thème Méthodes composant de la page Méthodes de l'Explorateur.

Vous pouvez sélectionner une méthode composant et cliquer sur le bouton Documentation de l'Explorateur pour obtenir des informations à son sujet, [le cas échéant](#).

The screenshot shows the 4D Explorer interface with the title bar "myProject - Explorateur". On the left, there's a sidebar with navigation links: Démarrage, Tables, Formulaires, Méthodes (which is selected and highlighted in blue), Commandes, Constantes, Plug-ins, and Corbeille. The main pane displays the "Méthodes" section under "Méthodes composant". A tree view shows various methods, with "PushNotification" being selected. To the right of the tree view, there's a large panel for the selected method. The title of the panel is "PushNotification" with a yellow bell icon. Below the title, a short description reads: "Utility class to send a push notification to one or multiple recipients." A section titled "Usage" provides instructions: "In order to use the component to send push notification, it is required to have an authentication key file `AuthKey_XXXXXX.p8` from Apple." It also links to "Check how to generate your authentication key .p8 file". Further down, it says: "To experiment the default behaviour, this file should be placed in your application sessions folder" and provides a path: "(`MobileApps/TEAM123456.com.sample.myappname`)." At the bottom of the panel, there's a code editor window showing some sample code:

```
$pushNotification:=MobileAppServer .PushNotification.new()  
  
$notification:=New object  
$notification.title:="This is title"
```

At the very bottom of the interface, there are buttons for "+", "-", settings, "Aperçu" (Preview), and "Documentation" (which is currently active).

Plug-ins

En développant une application 4D, vous découvrirez de nombreuses fonctionnalités que vous n'aviez pas remarquées au début. Vous pouvez même étendre la version standard de 4D en ajoutant des plug-ins à votre environnement de développement 4D.

Qu'est-ce qu'un plug-in et à quoi sert-il ?

Un plug-in est un morceau de code, écrit dans n'importe quel langage tel que C ou C++, que 4D lance au démarrage. Il ajoute des fonctionnalités à 4D et augmente ainsi sa capacité. Un plug-in contient généralement un ensemble de routines fournies au développeur 4D. Il peut gérer des zones externes et exécuter des process externes.

Où se trouvent les plug-ins ?

De nombreux plug-ins ont déjà été écrits par la communauté 4D. Les plug-ins publiés [sont disponibles sur GitHub](#). De plus, vous pouvez [développer vos propres plug-ins](#).

Installer un plug-in

Installez les plug-ins dans l'environnement 4D en copiant leurs fichiers dans le dossier Plugins, au [même niveau que le dossier Project](#).

Les plug-ins et les composants sont chargés par 4D lors du lancement de l'application. Vous devez donc quitter votre application 4D avant d'effectuer vos copies de fichiers ou dossiers. Si l'utilisation d'un plug-in nécessite une licence spécifique, le plug-in est chargé mais n'est pas actif.

Utilisation des plug-ins

Les commandes de plug-ins peuvent être utilisées comme des commandes 4D classiques de votre développement 4D. Les commandes de plug-ins apparaissent dans la page Plug-ins de l'Explorateur.

Identifiants

Cette section détaille les règles d'écriture et de nommage appliquées aux divers identifiants utilisés dans le langage de 4D (variables, propriétés d'objets, tableaux, formulaires, etc.).

En cas d'utilisation de caractères non-romans dans les noms des identifiants, leur longueur maximum peut être inférieure.

Tableaux

Les noms de tableaux suivent les mêmes règles que les noms de [variables](#).

Classes

Le nom d'une classe peut contenir jusqu'à 31 caractères.

Un nom de classe doit être conforme aux [règles standard de nommage des propriétés](#) au regard de la notation à points.

Donner le même nom à une classe et à une [table de la base](#) est déconseillé afin d'éviter tout conflit.

Fonctions

Les noms de fonctions doivent être conformes aux [règles standard de nommage des propriétés](#) au regard de la notation à points.

Astuce : Si vous préfixez le nom d'une fonction par un trait de soulignement ("_"), elle ne sera pas proposée par les fonctionnalités d'auto-complétion dans l'éditeur de code 4D.

Propriétés des objets

Le nom d'une propriété d'objet (aussi appelé *attribut*) peut contenir jusqu'à 255 caractères.

Les propriétés d'objets peuvent référencer des valeurs scalaires, des éléments ORDA, des fonctions de classe, d'autres objets, etc. Quelle que soit leur nature, les noms des propriétés d'objets doivent suivre certaines règles si vous souhaitez utiliser la [notation à point](#) :

- Un nom de propriété doit commencer par une lettre, un trait de soulignement ("_") ou un dollar ("\$").
- Ensuite, le nom peut inclure des lettres, des chiffres, des traits de soulignement ("_") ou des dollars ("\$").
- Les noms de propriétés sont sensibles à la casse.

Voici quelques exemples :

```
monObjet.monAttribut:="10"  
$valeur:=$clientObj.data.address.city
```

Si vous utilisez la notation chaîne avec des crochets, les noms de propriété peuvent contenir n'importe quel caractère (ex: `myObject["1. First property"]`).

Voir également le [standard ECMA Script](#).

Paramètres

Les noms de paramètres doivent commencer par un caractère `$` et suivent les mêmes règles que les [noms de variables](#).

Voici quelques exemples :

```
Function getArea($width : Integer; $height : Integer) -> $area : Integer  
#DECLARE ($i : Integer ; $param : Date) -> $myResult : Object
```

Méthodes

Le nom d'une méthode projet peut contenir jusqu'à 31 caractères.

- Un nom de méthode projet doit commencer par une lettre, un chiffre ou un trait de soulignement
- Ensuite, le nom peut inclure n'importe quelle lettre, chiffre, un trait de soulignement ("_") ou un caractère espace.
- N'utilisez pas de noms réservés, i.e. des noms de commandes 4D (`Date`, `Time`, etc), des mots-clés (`If`, `For`, etc.), des noms de constantes (`Euro`, `Black`, `Friday`), etc.
- Les noms des méthodes projets ne sont PAS sensibles à la casse.

Voici quelques exemples :

```
If(New client)  
DELETE DUPLICATED VALUES  
APPLY TO SELECTION([Employees];INCREASE SALARIES)
```

Conseil : Nous vous recommandons d'adopter, pour nommer vos méthodes, la même convention que celle utilisée dans le langage de 4D. Ecrivez les noms de vos procédures en caractères majuscules, et vos fonctions en minuscules avec la première lettre en majuscule. Ainsi, lorsque vous rouvrirez un projet au bout de plusieurs mois, vous identifierez immédiatement si une méthode retourne ou non un résultat, en regardant son nom dans la fenêtre de l'Explorateur.

Lorsque vous appelez une méthode, vous saisissez simplement son nom. Toutefois, certaines commandes 4D intégrées telles que `ON EVENT CALL`, ainsi que les commandes des plug-ins, nécessitent que vous passiez le nom d'une méthode en tant que chaîne lorsqu'un paramètre de type méthode est attendu.

Voici quelques exemples :

```
// Cette commande attend une méthode (fonction) ou une formule  
QUERY BY FORMULA([aTable];Special query)  
// Cette commande attend une méthode (procédure) ou une formule  
APPLY TO SELECTION([Employees];INCREASE SALARIES)  
// Mais cette commande attend un nom de méthode  
ON EVENT CALL("HANDLE EVENTS")
```

Tables et champs

Vous désignez une table en écrivant son nom entre crochets : [...]. Vous désignez un champ en spécifiant d'abord la table à laquelle il appartient (le nom du champ suit immédiatement celui de la table).

Un nom de table ou de champ peut contenir jusqu'à 31 caractères.

- Un nom de table ou de champ doit commencer par une lettre, un trait de soulignement ("_") ou un dollar ("\$").
- Le nom peut ensuite contenir des caractères alphabétiques, des caractères numériques, des espaces et des tirets bas (_).
- N'utilisez pas de noms réservés, i.e. des noms de commandes 4D (`Date`, `Time`, etc), des mots-clés (`If`, `For`,

etc.), des noms de constantes (Euro , Black , Friday), etc.

- Des règles supplémentaires sont à respecter lorsque la base doit être manipulée via le SQL : seuls les caractères _0123456789abcdefghijklmnoprstuvwxyz sont acceptés, et le nom ne doit pas comporter de mot-clé SQL (commande, attribut, etc.).

Voici quelques exemples :

```
FORM SET INPUT([Clients];"Entry")
ADD RECORD([Letters])
[Orders]Total:=Sum([Line]Amount)
QUERY([Clients];[Clients]Name="Smith")
[Letters]Text:=Capitalize_text([Letters]Text)
```

Donner le même nom à une table et à une classe est déconseillé afin d'éviter tout conflit.

Variables

Le nom d'une variable peut contenir jusqu'à 31 caractères, symboles de portée non compris (\$ ou <>).

- Un nom de variable doit commencer par une lettre, un trait de soulignement ou un dollar ("\$") pour les paramètres et les variables locales, ou "<>" pour les variables interprocess.
- Un chiffre en premier caractère est autorisé mais non recommandé, et n'est pas pris en charge par la déclaration de syntaxe var .
- Ensuite, le nom peut inclure des lettres, chiffres, et traits de soulignement ("_").
- Un espace en premier caractère est autorisé mais non recommandé, et n'est pas pris en charge par la déclaration de syntaxe var .
- N'utilisez pas de noms réservés, i.e. des noms de commandes 4D (Date , Time , etc), des mots-clés (If , For , etc.), des noms de constantes (Euro , Black , Friday), etc.
- Les noms de variables ne sont PAS sensibles à la casse.

Voici quelques exemples :

```
For($vlRecord;1;100) //variable locale
$vsMyString:="Hello there" //variable locale
var $vName; $vJob : Text //variables locales
If(bValidate=1) //variable process
<>vlProcessID:=Current process() //variable interprocess
```

Autres noms

Dans le langage de 4D, plusieurs éléments ont des noms manipulés sous forme de chaînes : formulaires, objets de formulaires, sélections temporaires, process, ensembles, barres de menus, etc.

Ces noms peuvent contenir jusqu'à 255 caractères, sans compter les éventuels caractères "\$" ou "<>".

- Les noms sous forme de chaînes peuvent contenir n'importe quel caractère.
- Les noms sous forme de chaînes ne sont pas sensibles à la casse.

Voici quelques exemples :

```
DIALOG([Storage];"Note box"+String($vlStage))
OBJECT SET FONT(*;"Binfo";"Times")
USE NAMED SELECTION([Customers];"Closed")//Sélection temporaire process
USE NAMED SELECTION([Customers];"<>ByZipcode") //Sélection temporaire interprocess
    //Démarrage du process global "Add Customers"
$vlProcessID:=New process("P_ADD_CUSTOMERS";48*1024;"Add Customers")
    //Démarrage du process local "$Follow Mouse Moves"
$vlProcessID:=New process("P_MOUSE_SNIFFER";16*1024;"$Follow Mouse Moves")
CREATE SET([Customers];"Customer Orders")//Ensemble process
USE SET("<>Deleted Records") //Ensemble interprocess
If(Records in set("$Selection"+String($i))>0) //Ensemble client
```

Objets Data Model

La technologie ORDA est fondée sur une cartographie automatique d'une structure de base sous-jacente. Elle permet également d'accéder aux données via des objets sélection d'entités (entity selection) et entité (entity). Par conséquent, ORDA expose la base de données entière comme un ensemble d'objets de modèle de données.

Correspondance de la structure

Lorsque vousappelez un datastore à l'aide de la commande `ds` ou `Open datastore`, 4D référence automatiquement les tables et les champs de la structure 4D correspondante en tant que propriétés de l'objet `datastore` retourné :

- Les tables correspondent à des dataclasses.
- Les champs correspondent à des attributs de stockage.
- Les relations correspondent à des attributs de relation - les noms de relation, définis dans l'éditeur de structure, sont utilisés comme noms d'attribut de relation.

Expression	Value
ds.Company	DataClass: Company ("name": "companyProjects", "kind": "relatedEntities", "relatedDataClass": "Project", "type": "ProjectSelection") ("name": "discount", "kind": "storage", "type": "number", "fieldNumber": 3) ("name": "ID", "kind": "storage", "type": "long", "fieldNumber": 1) ("name": "name", "kind": "storage", "type": "string", "fieldNumber": 2)
ds.Project	DataClass: Project ("name": "companyID", "kind": "storage", "type": "long", "fieldNumber": 3) ("name": "ID", "kind": "storage", "type": "long", "fieldNumber": 1) ("name": "name", "kind": "storage", "type": "string", "fieldNumber": 2) ("name": "theClient", "kind": "relatedEntity", "relatedDataClass": "Company", "type": "Company")

Règles générales

Les règles suivantes s'appliquent à toutes les conversions :

- Les noms de table, de champ et de relation correspondent à des noms de propriété d'objet. Assurez-vous que ces noms sont conformes aux règles générales de dénomination des objets, comme expliqué dans la section [Conventions de dénomination des objets](#).
- Un datastore ne référence que les tables avec une seule clé primaire. Les tables suivantes ne sont pas référencées :
 - Tables sans clé primaire
 - Tables avec clés primaires composites.
- Les champs BLOB sont automatiquement disponibles comme attributs de type [objet Blob](#).

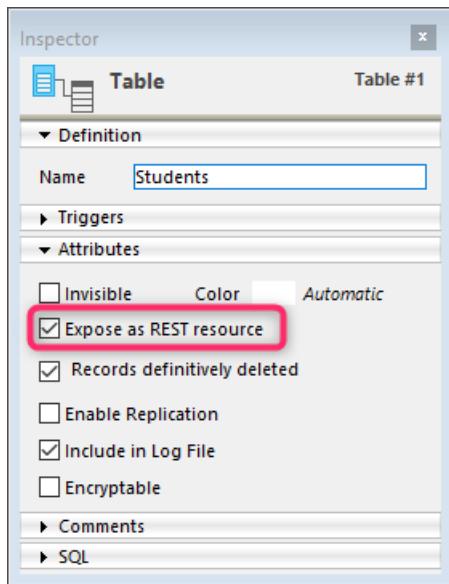
ORDA mapping does not take into account:

- the "Invisible" option for tables or fields, - the virtual structure defined through `SET TABLE TITLES` or `SET FIELD TITLES`, - the "Manual" or "Automatic" property of relations.

Règles de contrôle d'accès à distance

Lorsque vous accédez à un datastore distant via la commande `Ouvrir datastore` ou des [requêtes REST](#), seules les tables et les champs avec la propriété de ressource Expose as REST resource sont disponibles à distance.

Cette option doit être choisie au niveau de la structure 4D pour chaque table et chaque champ que vous souhaitez voir apparaître comme dataclass et attribut dans le datastore :



Mise à jour des données

Toute modification apportée à la structure de la base invalide la couche courante de données ORDA. Ces modifications incluent :

- l'ajout ou la suppression d'une table, d'un champ ou d'une relation
- le renommage d'une table, d'un champ ou d'une relation
- la modification d'une propriété principale d'un champ (type, unique, index, autoincrement, valeur null)

Lorsque la couche courante de données ORDA est invalidée, elle est automatiquement rechargée et mise à jour dans les prochains appels du datastore local `ds` vers 4D et 4D Server. A noter que les références existantes vers des objets ORDA tels que des entités ou des sélections d'entités continueront d'utiliser les données à partir desquelles elles ont été créées, et ce jusqu'à ce qu'elles soient regénérées.

Toutefois, la couche de données ORDA mise à jour n'est pas automatiquement disponible dans les contextes suivants :

- une application 4D distante connectée à 4D Server -- l'application distante doit être reconnectée au serveur.
- un datastore distant ouvert à l'aide de `Ouvrir datastore` ou des [appels REST](#) -- une nouvelle session doit être ouverte.

Définitions des objets

Datastore

Un datastore est l'objet d'interface d'une base de données. Il crée une représentation de toute la base sous forme d'objet. Un datastore est constitué d'un modèle et à de données :

- Le modèle contient et décrit toutes les dataclasses qui composent le datastore. Il est indépendant de la base de données sous-jacente.
- Les données se réfèrent à l'information qui va être utilisée et stockée dans ce modèle. Par exemple, les noms, adresses et dates de naissance des employés sont des éléments de données que vous pouvez utiliser dans un datastore.

Lorsqu'il est géré via le code, le datastore est un objet dont les propriétés sont toutes les [dataclasses](#) ayant été spécifiquement exposées.

4d vous permet de gérer les datastores suivants :

- le datastore local, fondé sur la base 4D courante, retourné par la commande `ds` (le datastore principal).
- un ou plusieurs datastores distants, exposés en tant que ressources RESET dans des bases 4D distantes, retournés par la commande `Ouvrir datastore`.

Un datastore ne référence qu'une seule base de données locale ou distante.

L'objet datastore lui-même ne peut pas être copié en tant qu'objet :

```
$mydatastore:=OB Copy(ds) //retourne null
```

Les propriétés du datastore sont toutefois énumérables :

```
ARRAY TEXT($prop;0)
OB GET PROPERTY NAMES(ds;$prop)
//$prop contient les noms de toutes les dataclasses
```

Le datastore principal (par défaut) est toujours disponible via la commande `ds`, mais la commande `Ouvrir datastore` permet de référencer n'importe quel datastore distant.

Dataclass

Une dataclasse est l'équivalent d'une table. Elle est utilisée comme modèle d'objet et référence tous les champs comme attributs, y compris les attributs relationnels (attributs construits à partir des relations entre les dataclasses). Les attributs relationnels peuvent être utilisés dans les requêtes comme tout autre attribut.

Toutes les dataclasses d'un projet 4D sont disponibles en tant que propriété du datastore `ds`. Pour les datastores distants accédés via `Ouvrir datastore` ou les [requêtes REST](#), l'option Exposer comme ressource REST doit être sélectionnée au niveau de la structure 4D pour chaque table que vous souhaitez exposer en tant que dataclass du datastore.

Par exemple, considérons cette table dans la structure suivante :

Company	
ID	2 ³²
name	A
creationDate	17/01/2017
revenues	0.5
extra	{}

La table `Company` est automatiquement disponible en tant que dataclasse dans la banque de données `ds`. Vous pouvez écrire :

```
var $compClass : cs.Company //déclare une variable objet $compClass de la classe Company
$compClass:=ds.Company //affecte la référence de dataclasse Company à $compClass
```

Un objet dataclass peut contenir :

- attributs
- des attributs relationnels

La dataclass offre une abstraction de la base de données physique et permet de gérer un modèle de données conceptuel. La dataclass est le seul moyen d'interroger le datastore. Une requête est effectuée à partir d'une seule dataclass. Les requêtes sont construites autour des attributs et des noms d'attributs relationnels des dataclasses. Les attributs relationnels sont ainsi les moyens d'impliquer plusieurs tables liées dans une requête.

L'objet dataclass lui-même ne peut pas être copié en tant qu'objet :

```
$mydataclass:=OB Copy(ds.Employee) //retourne null
```

Les propriétés de la dataclass sont toutefois énumérables :

```
ARRAY TEXT($prop;0)
OB GET PROPERTY NAMES(ds.Employee;$prop)
//$prop contient les noms de tous les attributs de dataclass
```

Attribut

Les propriétés de dataclass sont des objets attribut décrivant les champs ou relations sous-jacents. Par exemple :

```
$nameAttribute:=ds.Company.name //référence à un attribut de classe
$revenuesAttribute:=ds.Company["revenues"] //méthode alternative
```

Ce code attribue à `$nameAttribute` et `$revenuesAttribute` des références aux attributs name et revenues de la classe `Company`. Cette syntaxe ne retourne PAS les valeurs contenues dans l'attribut, mais retourne plutôt des références aux attributs eux-mêmes. Pour gérer les valeurs, vous devez passer par les [Entités](#).

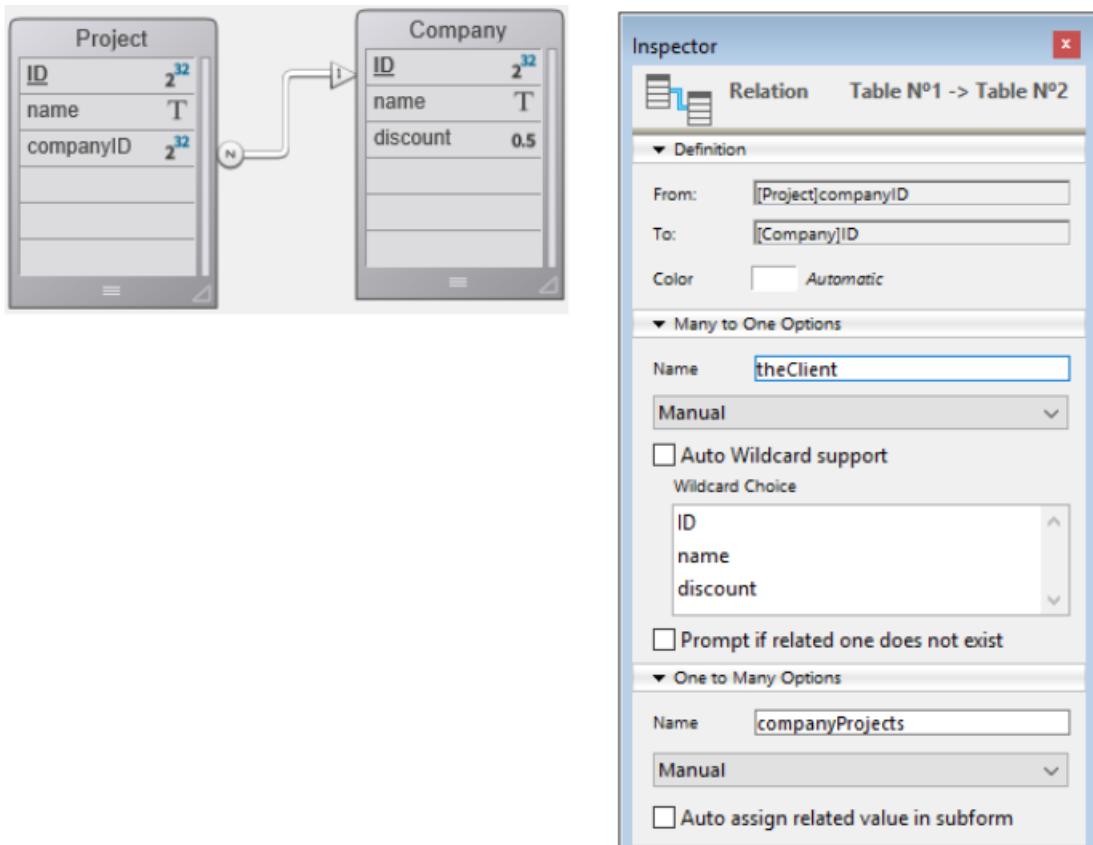
Tous les fichiers éligibles d'une table sont disponibles en tant qu'attributs de leur `dataclass` parente. Pour les datastores distants accédés via [Ouvrir datastore](#) ou les [requêtes REST](#), l'option Exposer comme ressource REST doit être sélectionnée au niveau de la structure 4D pour chaque champ que vous souhaitez exposer en tant qu'attribut de dataclass.

Attributs de stockage et relationnels

Les attributs de la Dataclass sont de plusieurs types : storage, relatedEntity et relatedEntities. Les attributs scalaires (c'est-à-dire qui ne fournissent qu'une seule valeur) prennent en charge tous les types de données standard 4D (Entier, texte, objet, etc.).

- Un attribut de stockage (storage) est équivalent à un champ dans la base de données 4D et peut être indexé. Les valeurs affectées à un attribut de stockage sont stockées en tant que partie de l'entité lors de son enregistrement. Lorsqu'on accède à un attribut de stockage, sa valeur provient directement du datastore. Les attributs de stockage sont le bloc de construction le plus élémentaire d'une entité et sont définis par un nom et un type de données.
- Un attribut relationnel (relatedEntity et relatedEntities) donne accès à d'autres entités. Les attributs relationnels peuvent aboutir soit à une entité unique (ou à aucune entité), soit à une sélection d'entité (0 à N entités). Les attributs relationnels s'appuient sur les relations "classiques" dans la structure relationnelle pour fournir un accès direct à une entité ou à des entités reliées. Tous les attributs relationnels sont directement disponibles dans ORDA si vous utilisez leurs noms.

Prenons l'exemple de la structure de base de données partielle suivante et les propriétés relationnelles :



Tous les attributs relationnels seront disponibles automatiquement :

- dans la dataclass Project : "ID", "name", et "companyID"
- dans la dataclass Company : "ID", "name", et "discount"

En outre, les attributs relationnels suivant seront également disponibles automatiquement :

- dans la dataclass Project : l'attribut `theClient`, du type "relatedEntity" ; il y a au plus une compagnie pour chaque projet (le client)
- dans la dataclass Company : l'attribut `companyProjects`, du type "relatedEntities" ; pour chaque compagnie, il existe un certain nombre de projets reliés.

Les entity selections peuvent également être "partageables" ou "non partageables", selon [la façon dont elles ont été créées](#).

Tous les attributs de la dataclass sont exposés en tant que propriétés de la dataclass :

Expression	Value
ds.Company	DataClass: Company
ds.Company.companyProjects	{"name":"companyProjects","kind":"relatedEntities","relatedDataClass":"Project","type":"ProjectSelection"} {"name":"discount","kind":"storage","type":"number","fieldNumber":3} {"name":"ID","kind":"storage","type":"long","fieldNumber":1} {"name":"name","kind":"storage","type":"string","fieldNumber":2}
ds.Company.discount	
ds.Company.ID	
ds.Company.name	
ds.Project	DataClass: Project
ds.Project.companyID	{"name":"companyID","kind":"storage","type":"long","fieldNumber":3} {"name":"ID","kind":"storage","type":"long","fieldNumber":1} {"name":"name","kind":"storage","type":"string","fieldNumber":2}
ds.Project.ID	
ds.Project.name	
ds.Project.theClient	{"name":"theClient","kind":"relatedEntity","relatedDataClass":"Company","type":"Company"}

Gardez à l'esprit que ces objets décrivent des attributs, mais ne donnent pas accès aux données. La lecture ou l'écriture des données se fait à travers des [objets entité](#).

Champs calculés

dans la [définition de la classe Entity](#). Leur valeur n'est pas stockée mais évaluée à chaque fois qu'on y accède. Ils n'appartiennent pas à la structure sous-jacente de la base, mais ils se basent sur elle et peuvent être utilisés comme

n'importe quel champ du modèle de données.

Entity

Une entité est l'équivalent d'un enregistrement. Il s'agit d'un objet qui fait référence à un enregistrement de la base de données. Elle peut être perçue comme une instance de la [dataclass](#), comme un enregistrement de la table correspondante à la dataclass. Toutefois, une entité contient également des données corrélées à la base de données liée au datastore.

Le but de l'entité est de gérer les données (créer, mettre à jour, supprimer). Lorsqu'une référence d'entité est obtenue au moyen d'une sélection d'entité, elle conserve également des informations sur la sélection d'entité qui permet une itération à travers la sélection.

L'objet entité lui-même ne peut pas être copié en tant qu'objet :

```
$myentity:=OB Copy(ds.Employee.get(1)) //retourne null
```

Les propriétés de l'entité sont toutefois énumérables :

```
ARRAY TEXT($prop;0)
OB GET PROPERTY NAMES(ds.Employee.get(1);$prop)
//$prop contient les noms de tous les attributs de l'entité
```

Entity selection

Une sélection d'entité est un objet contenant une ou plusieurs référence(s) à des entités appartenant à la même dataclasse. Elle est généralement créée à la suite d'une requête ou retournée à partir d'un attribut relationnel. Une entity selection peut contenir 0, 1 ou X entités de la dataclass - où X peut représenter le nombre total d'entités contenues dans la dataclass.

Exemple :

```
var $e : cs.EmployeeSelection //déclare une variable objet $e de type de classe EmployeeSelection
$e:=ds.Employee.all() //assigne la référence de la sélection d'entité résultante à la variable $e
```

Les entity selections peuvent être "triées" ou "non triées" ([voir ci-dessous](#)).

Les entity selections peuvent également être "partageables" ou "non partageables", selon [la façon dont elles ont été créées](#).

L'objet sélection d'entités lui-même ne peut pas être copié en tant qu'objet :

```
$myentitysel:=OB Copy(ds.Employee.all()) //retourne null
```

Les propriétés des sélections d'entités sont toutefois énumérables :

```
ARRAY TEXT($prop;0)
OB GET PROPERTY NAMES(ds.Employee.all();$prop)
//$prop contient les noms des propriétés des sélections d'entités
//("length", "00", "01")
```

Entity selections triées vs Entity selections non-triées

Pour des raisons d'optimisation, par défaut, 4D ORDA crée généralement des sélections d'entités non-ordonnées, sauf

lorsque vous utilisez la méthode `orderBy()` ou si vous utilisez les options appropriées. Dans cette documentation, sauf indication contraire, "sélection d'entités" fait généralement référence à une "sélection d'entités non-ordonnée".

Les sélections d'entités ordonnées sont créées uniquement lorsque cela est nécessaire ou lorsqu'elles sont spécifiquement demandées à l'aide d'options, c'est-à-dire dans les cas suivants :

- résultat d'un `orderBy()` sur une sélection (de n'importe quel type) ou un `orderBy()` sur une dataclass,
- résultat de la méthode `newSelection()` avec l'option `dk keep ordered`

Les sélections d'entités non-triées sont créées dans les cas suivants :

- résultat d'un `query()` standard sur une sélection (de n'importe quel type) ou un `query()` sur une dataclass,
- résultat de la méthode `newSelection()` sans option,
- résultat de l'une des méthodes de comparaison, quel que soit le type de sélection saisi : `or()`, `and()`, `minus()`.

Les sélections d'entités suivantes sont toujours triées :

- sélections d'entités retournées par 4D Server vers un client distant
- sélections d'entités fondées sur des datastores distants.

Notez que lorsqu'une sélection d'entités ordonnée devient une sélection non-ordonnée, toute référence d'entité répétée est supprimée.

Classes du modèle de données

ORDA vous permet de créer des fonctions de classe de haut niveau au-dessus du modèle de données. Cela vous permet d'écrire du code orienté métier et de le «publier» comme une API. Le datastore, les dataclasses, les sélections d'entités et les entités sont tous disponibles en tant qu'objets de classe pouvant contenir des fonctions.

Par exemple, vous pouvez créer une fonction `getNextWithHigherSalary()` dans la classe `EmployeeEntity` pour retourner les employés ayant un salaire supérieur à celui qui est sélectionné. Il serait aussi simple à appeler que :

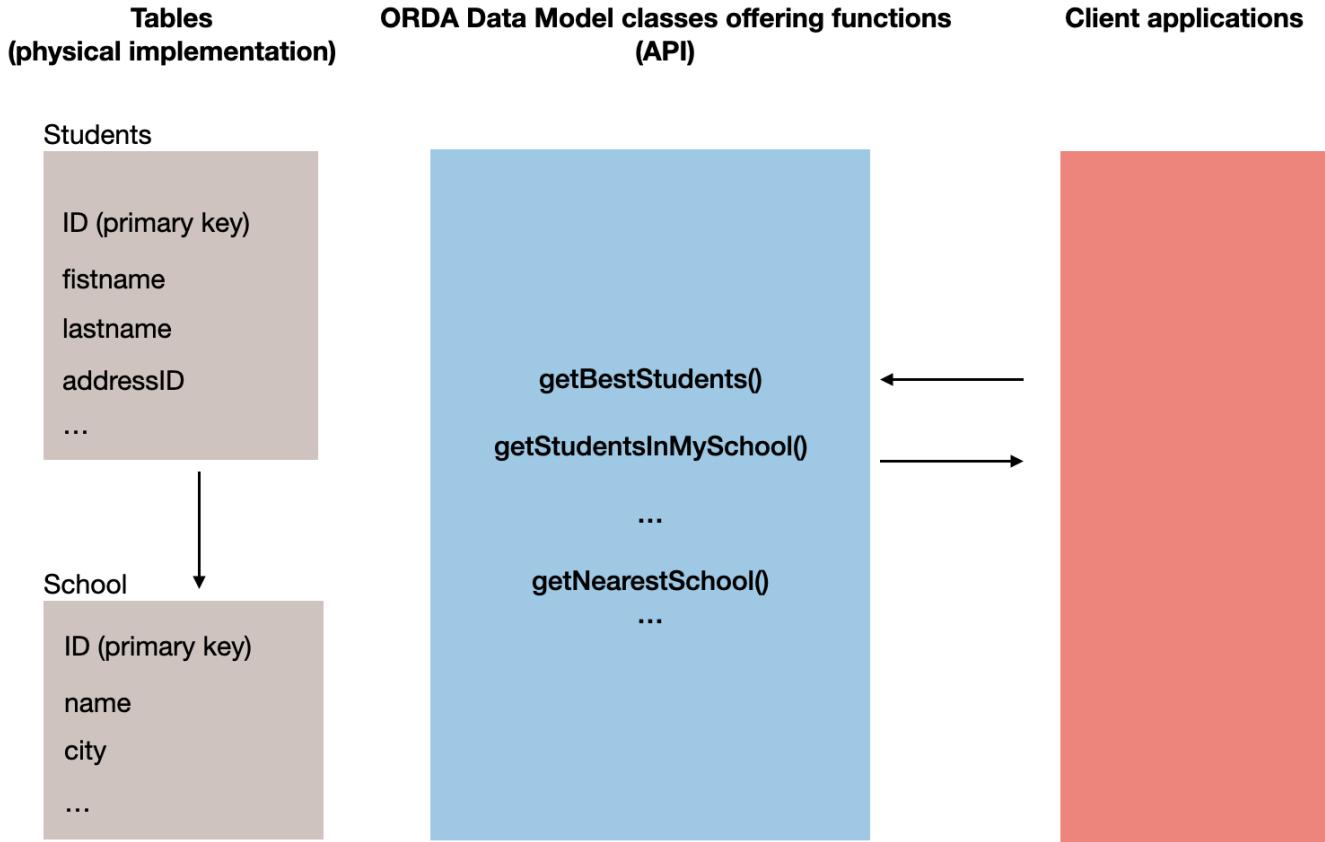
```
$nextHigh:=ds.Employee(1).getNextWithHigherSalary()
```

Les développeurs peuvent utiliser ces fonctions non seulement dans des datastores locaux, mais aussi dans des architectures client/serveur et distantes :

```
//$cityManager est la référence d'un datastore distant  
Form.comp.city:=$cityManager.City.getCityName(Form.comp.zipcode)
```

Grâce à cette fonctionnalité, toute la logique métier de votre application 4D peut être stockée comme une couche indépendante afin d'être facilement maintenue ou réutilisée avec un niveau de sécurité élevé :

- Elle vous permet de «masquer» la complexité globale de la structure physique sous-jacente et d'exposer uniquement des fonctions compréhensibles et prêtées à l'emploi.
- Si la structure physique évolue, il vous suffit d'adapter le code de la fonction et les applications clientes continueront de les appeler de manière transparente.
- By default, all of your data model class functions (including [computed attribute functions](#)) and [alias attributes](#) are not exposed to remote applications and cannot be called from REST requests. Vous devez déclarer explicitement chaque fonction publique et alias avec le mot-clé `exposed`.



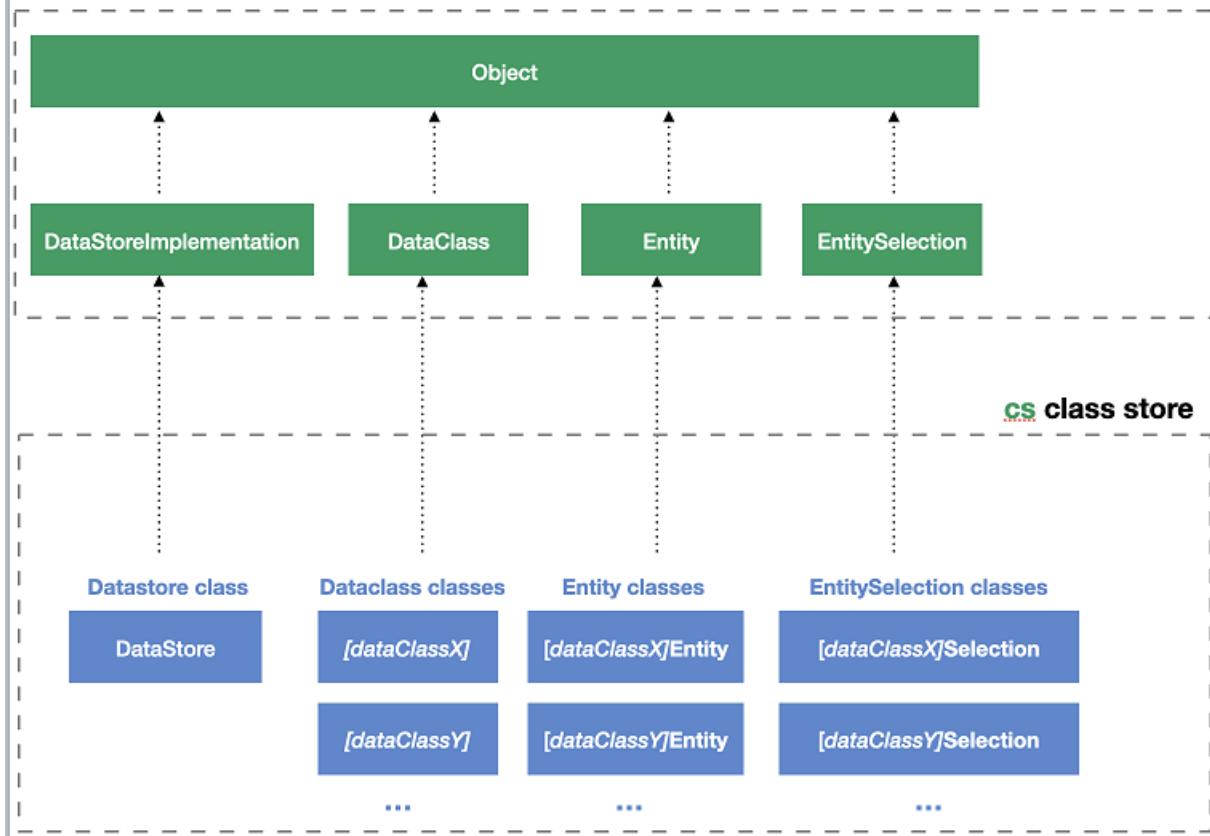
4D database (eventually exposed as a REST server)

De plus, 4D [crée préalablement et automatiquement](#) les classes pour chaque objet de modèle de données disponible.

Architecture

ORDA fournit des classes génériques exposées via le [class store](#) 4D, ainsi que des classes utilisateurs (étendant les classes génériques) exposées dans le [class store](#) cs :

4D class store



Toutes les classes de modèle de données ORDA sont exposées en tant que propriétés du class store `cs`. Les classes ORDA suivantes sont disponibles :

Class	Nom de l'exemple	Instanciée par
<code>cs.DataStore</code>	<code>cs.DataStore</code>	commande <code>ds</code>
<code>cs.DataClassName</code>	<code>cs.Employee</code>	<code>dataStore.DataClassName</code> , <code>dataStore["DataClassName"]</code>
<code>cs.DataClassNameEntity</code>	<code>cs.EmployeeEntity</code>	<code>dataClass.get()</code> , <code>dataClass.new()</code> , <code>entitySelection.first()</code> , <code>entitySelection.last()</code> , <code>entity.previous()</code> , <code>entity.next()</code> , <code>entity.first()</code> , <code>entity.last()</code> , <code>entity.clone()</code>
<code>cs.DataClassNameSelection</code>	<code>cs.EmployeeSelection</code>	<code>dataClass.query()</code> , <code>entitySelection.query()</code> , <code>dataClass.all()</code> , <code>dataClass.fromCollection()</code> , <code>dataClass.newSelection()</code> , <code>entitySelection.drop()</code> , <code>entity.getSelection()</code> , <code>entitySelection.and()</code> , <code>entitySelection.minus()</code> , <code>entitySelection.or()</code> , <code>entitySelection.orderBy()</code> , <code>entitySelection.orderByFormula()</code> , <code>entitySelection.slice()</code> , <code>Create entity selection</code>

Les classes utilisateur ORDA sont stockées sous forme de fichiers de classe standard (.4dm) dans le sous-dossier Classes du projet ([voir ci-dessous](#)).

De plus, les instances d'objet de classes utilisateurs du modèles de données ORDA bénéficient des propriétés et fonctions de leurs parents:

- un objet de classe Datastore peut appeler des fonctions de la [classe générique ORDA Datastore](#).
- un objet de classe Dataclass peut appeler des fonctions de la [classe générique ORDA Dataclass](#).
- un objet de classe Entity selection peut appeler des fonctions de la [classe générique ORDA Entity selection](#).
- un objet de classe Entity peut appeler des fonctions de la [classe générique ORDA Entity](#).

Description de la classe

► Historique

Classe DataStore

Une base de données 4D expose sa propre classe DataStore dans le class store `cs`.

- Extends: `4D.DataStoreImplementation`
- Nom de classe : `cs.DataStore`

Vous pouvez créer des fonctions dans la classe DataStore qui seront disponibles via l'objet `ds`.

Exemple

```
// cs.DataStore class

Class extends DataStoreImplementation

Function getDesc
    $0:="Database exposing employees and their companies"
```

Cette fonction peut alors être appelée :

```
$desc:=ds.getDesc() // "Database exposing..."
```

Classe DataClass

Chaque table exposée avec ORDA affiche une classe DataClass dans le class store `cs`.

- Extends : `4D.DataClass`
- **Nom de classe : `cs.DataClassName` (où `DataClassName` est le nom de la table)
- **Exemple ** : `cs.Employee`

Exemple

```
// cs.Company class

Class extends DataClass

// Retourne les entreprises dont le revenu est supérieur à la moyenne
// Retourne une sélection d'entités relative à l'entreprise DataClass

Function GetBestOnes()
    $sel:=This.query("revenues >= :1";This.all().average("revenues"));
    $0:=$sel
```

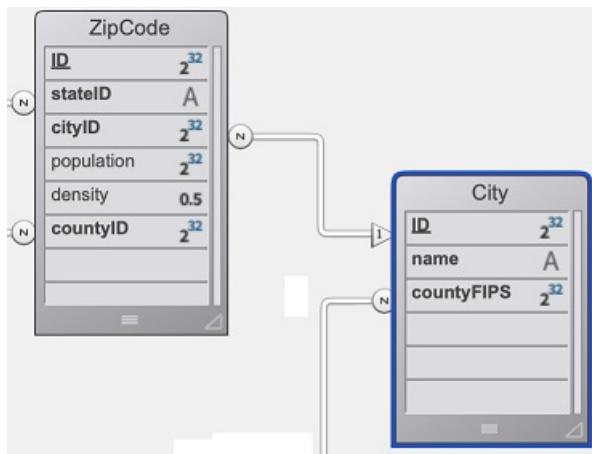
Vous pouvez ensuite obtenir une sélection d'entité des "meilleures" entreprises en exécutant le code suivant :

```
var $best : cs.CompanySelection
$best:=ds.Company.GetBestOnes()
```

Les champs calculés sont définis dans [la classe Entity](#).

Exemple avec un datastore distant

Le catalogue *City* suivant est exposé dans un datastore distant (vue partielle) :



La classe `City Class` fournit une API :

```
// cs.City class

Class extends DataClass

Function getCityName()
    var $1; $zipcode : Integer
    var $zip : 4D.Entity
    var $0 : Text

    $zipcode:=$1
    $zip:=ds.ZipCode.get($zipcode)
    $0:=""

    If ($zip#Null)
        $0:=$zip.city.name
    End if
```

L'application cliente ouvre une session sur le datastore distant :

```
$cityManager:=Open datastore(New object("hostname";"127.0.0.1:8111");"CityManager")
```

Une application cliente peut alors utiliser l'API pour obtenir la ville correspondant au code postal (par exemple) à partir d'un formulaire :

```
Form.comp.city:=$cityManager.City.getCityName(Form.comp.zipcode)
```

Classe EntitySelection

Chaque table exposée avec ORDA affiche une classe EntitySelection dans le class store `cs`.

- Extends : `4D.EntitySelection`
- Nom de classe : `DataClassNameSelection` (où `DataClassName` est le nom de la table)
- **Exemple** : `cs.EmployeeSelection`

Exemple

```
// Classe cs.EmployeeSelection

Classe extends EntitySelection

//Extrait, de cette sélection d'entité, les employés ayant un salaire supérieur à la moyenne

Function withSalaryGreaterThanAverage
    C_OBJECT($0)
    $0:=This.query("salary > :1";This.average("salary")).orderBy("salary")
```

Vous pouvez alors obtenir les employés dont le salaire est supérieur à la moyenne, dans une sélection d'entité, en exécutant le code suivant :

```
$moreThanAvg:=ds.Company.all().employees.withSalaryGreaterThanAverage()
```

Entity Class

Chaque table exposée avec ORDA affiche une classe Entity dans le class store `cs`.

- Extends : `4D.Entity`
- **Nom de classe** : `DataClassNameEntity` (où `DataClassName` est le nom de la table)
- **Exemple** : `cs.CityEntity`

Champs calculés

Les classes Entity vous permettent de définir des champs calculés à l'aide de mots-clés spécifiques :

- `Function get attributeName`
- `Function set attributeName`
- `Function query attributeName`
- `Function orderBy attributeName`

For information, please refer to the [Computed attributes](#) section.

Attributs de type alias

Entity classes allow you to define alias attributes, usually over related attributes, using the `Alias` keyword:

```
Alias attributeName targetPath
```

For information, please refer to the [Alias attributes](#) section.

Exemple

```
// cs.CityEntity class

Class extends Entity

Function getPopulation()
    $0:=This.zips.sum("population")

Function isBigCity(): Boolean
// La fonction getPopulation() est utilisable dans la classe
$0:=This.getPopulation()>50000
```

Vous pouvez ensuite appeler ce code :

```

var $cityManager; $city : Object

$cityManager:=Open datastore(New object("hostname";"127.0.0.1:8111");"CityManager")
$city:=$cityManager.City.getCity("Caguas")

If ($city.isBigCity())
    ALERT($city.name + " is a big city")
End if

```

Règles spécifiques

Lors de la création ou de la modification de classes de modèles de données, vous devez veiller aux règles décrites ci-dessous :

- Puisqu'ils sont utilisés pour définir des noms de classe DataClass automatiques dans le `class store` cs, les tables 4D doivent être nommées afin d'éviter tout conflit dans l'espace de nommage cs. En particulier :
 - Ne donnez pas le même nom à une table 4D et à une `classe d'utilisateurs` (user class). Si un tel cas se produit, le constructeur de la classe utilisateur devient inutilisable (un avertissement est retourné par le compilateur).
 - N'utilisez pas de nom réservé pour une table 4D (par exemple "DataClass").
- Lors de la définition d'une classe, assurez-vous que l'instruction `Class extends` correspond exactement au nom de la classe parente (sensible à la casse). Par exemple, `Class extends EntitySelection` pour une classe de sélection d'entité.
- Vous ne pouvez pas instancier un objet de classe de modèle de données avec le mot clé `new()` (une erreur est retournée). You must use a regular method as listed in the [Instantiated by column of the ORDA class table](#).
- Vous ne pouvez pas remplacer une fonction de classe ORDA native du `class store` 4D par une fonction de classe utilisateur de modèle de données.

Exécution préemptive

Lors de la compilation, les fonctions de classe du modèle de données sont exécutées :

- dans des process préemptifs ou coopératifs (en fonction du process appelant) dans des applications monoposte,
- dans des process préemptifs dans des applications client/serveur (sauf si le mot-clé `local` est utilisé, auquel cas il dépend du process appelant comme dans le cas d'un monoposte).

Si votre projet est conçu de façon à être exécuté en client/serveur, assurez-vous que le code de la fonction de classe du modèle de données est thread-safe. Si un code thread-unsafe est appelé, une erreur sera générée au moment de l'exécution (aucune erreur ne sera déclenchée au moment de la compilation puisque l'exécution coopérative est prise en charge dans les applications monoposte).

Champs calculés

Aperçu

Un champ calculé est un attribut de dataclass avec un type de données qui masque un calcul. [Les classes 4D standard](#) implémentent le concept de propriétés calculées avec des `fonctions d'accès` telles que `get` (getter) et `set` (setter). Les attributs de dataclass ORDA bénéficient de cette fonctionnalité et l'étendent avec deux fonctions supplémentaires : `query` et `orderBy`.

Un champ calculé nécessite au minimum une fonction `get` qui décrit comment sa valeur sera calculée. Lorsqu'une fonction `getter` est fournie à un attribut, 4D ne crée pas l'espace de stockage sous-jacent dans le datastore mais substitue le code de la fonction chaque fois que l'attribut est accédé. Si l'attribut n'est pas consulté, le code ne s'exécute jamais.

Un champ calculé peut également mettre en œuvre une fonction `set`, qui s'exécute chaque fois qu'une valeur est attribuée à l'attribut. La fonction `setter` décrit ce qui est à faire avec la valeur attribuée, généralement en la redirigeant

vers un ou plusieurs attributs de stockage ou, dans certains cas, vers d'autres entités.

Tout comme les champs de stockage, les champs calculés peuvent être inclus dans les requêtes. Par défaut, lorsqu'un champ calculé est utilisé dans une requête ORDA, il est calculé une fois par entité examinée. Dans certains cas, cela est suffisant. Cependant, pour de meilleures performances, notamment en client/serveur, les champs calculés peuvent implémenter une fonction de requête `query` qui s'appuie sur les attributs des dataclass et qui bénéficie de leurs index.

De même, les champs calculés peuvent être inclus dans des tris. Lorsqu'un champ calculé est utilisé dans un tri ORDA, l'attribut est calculé une fois par entité examinée. Tout comme dans les requêtes, les champs calculés peuvent mettre en œuvre une fonction `orderBy` qui substitue d'autres attributs pendant le tri, améliorant ainsi les performances.

Comment définir les champs calculés

Créez un champ calculé en définissant un accesseur `get` dans la [classe entity](#) de la dataclass. Le champ calculé sera automatiquement disponible dans les attributs de la dataclass et dans les attributs de l'entité.

D'autres fonctions de champs calculés (`set`, `query` et `orderBy`) peuvent également être définies dans la classe entity. Elles sont facultatives.

Dans les fonctions de champs calculés, `This` désigne l'entité. Les champs calculés peuvent être utilisés et traités comme n'importe quel champ de dataclass, c'est-à-dire qu'ils seront traités par les fonctions de [classe entity](#) ou de [classe entity selection](#).

Les champs calculés d'ORDA ne sont pas exposés ([exposed](#)) par défaut. Exposez un champ calculé en ajoutant le mot-clé `exposed` lors de la définition de la fonction `get`.

Les fonctions `get` et `set` peuvent avoir la propriété `local` pour optimiser le traitement client/serveur.

Function `get <attributeName>`

Syntaxe

```
{local} {exposed} Function get <attributeName>({$event : Object}) -> $result : type  
// code
```

La fonction `getter` est obligatoire pour déclarer le champ calculé `attributeName`. Chaque fois que l'on accède à `attributeName`, 4D évalue le code de la fonction `getter` et retourne la valeur `$result`.

Un champ calculé peut utiliser la valeur d'un ou plusieurs autres champs calculés. Les appels récursifs génèrent des erreurs.

La fonction `getter` définit le type de données du champ calculé grâce au paramètre `$result`. Les types de résultats suivants sont autorisés :

- Scalar (text, boolean, date, time, number)
- Object
- Image
- BLOB
- Entity (i.e. cs.EmployeeEntity)
- Entity selection (i.e. cs.EmployeeSelection)

Les propriétés du paramètre `$event` sont les suivantes :

Propriété	Type	Description
attributeName	Text	Nom du champ calculé
dataClassName	Text	Nom de la dataclass
kind	Text	"get"
result	Variant	Optionnel. Complétez cette propriété avec la valeur Null si vous souhaitez qu'un champ scalaire retourne Null

Exemples

- Le champ calculé *fullName* :

```
Function get fullName($event : Object)-> $fullName : Text
Case of
  : (This.firstName=NULL) & (This.lastName=NULL)
    $event.result:=NULL //utiliser le résultat pour retourner Null
  : (This.firstName=NULL)
    $fullName:=This.lastName
  : (This.lastName=NULL)
    $fullName:=This.firstName
Else
  $fullName:=This.firstName+" "+This.lastName
End case
```

- Un champ calculé peut être basé sur un attribut relatif à une entité :

```
Function get bigBoss($event : Object)-> $result: cs.EmployeeEntity
$result:=This.manager.manager
```

- Un champ calculé peut être basé sur un attribut relatif à une entity selection :

```
Function get coWorkers($event : Object)-> $result: cs.EmployeeSelection
If (This.manager.manager=NULL)
  $result:=ds.Employee.newSelection()
Else
  $result:=This.manager.directReports.minus(this)
End if
```

Function set <attributeName>

Syntaxe

```
{local} Function set <attributeName>($value : type {; $event : Object})
// code
```

La fonction *setter* s'exécute chaque fois qu'une valeur est attribuée à l'attribut. Cette fonction traite généralement la ou les valeurs d'entrée et le résultat est réparti entre un ou plusieurs autres attributs.

Le paramètre *\$value* reçoit la valeur attribuée à l'attribut.

Les propriétés du paramètre *\$event* sont les suivantes :

Propriété	Type	Description
attributeName	Text	Nom du champ calculé
dataClassName	Text	Nom de la dataclass
kind	Text	"set"
value	Variant	Valeur à gérer par le champ calculé

Exemple

```
Function set fullName($value : Text; $event : Object)
    var $p : Integer
    $p:=Position(" "; $value)
    This.firstname:=Substring($value; 1; $p-1) // "" if $p<0
    This.lastname:=Substring($value; $p+1)
```

Function query <attributeName>

Syntaxe

```
Function query <attributeName>($event : Object)
Function query <attributeName>($event : Object) -> $result : Text
Function query <attributeName>($event : Object) -> $result : Object
// code
```

Cette fonction prend en charge trois syntaxes :

- Avec la première syntaxe, vous traitez l'ensemble de la requête via la propriété de l'objet objet `$event.result`.
- Avec les deuxième et troisième syntaxes, la fonction retourne une valeur dans `$result` :
 - Si `$result` est Text, il doit s'agir d'une chaîne de requête valide
 - Si `$result` est Object, il doit contenir deux propriétés :

Propriété	Type	Description
<code>\$result.query</code>	Text	Chaîne de requête valide avec placeholders (:1, :2, etc.)
<code>\$result.parameters</code>	Collection	valeurs pour placeholders

La fonction `query` s'exécute à chaque fois qu'une requête utilisant le champ calculé est lancée. Il est utile de personnaliser et d'optimiser les requêtes en s'appuyant sur les attributs indexés. Lorsque la fonction `query` n'est pas implémentée pour un champ calculé, la recherche est toujours séquentielle (basée sur l'évaluation de toutes les valeurs à l'aide de la fonction `get <AttributeName>`).

Les fonctionnalités suivantes ne sont pas prises en charge : - l'appel d'une fonction `query` sur des champs calculés de type Entity ou Entity selection, - l'utilisation du mot clé `order by` dans la chaîne de requête résultante.

Les propriétés du paramètre `$event` sont les suivantes :

Propriété	Type	Description
attributeName	Text	Nom du champ calculé
dataClassName	Text	Nom de la dataclass
kind	Text	"query"
value	Variant	Valeur à gérer par le champ calculé
operator	Text	Opérateur de requête (voir également la fonction de classe <code>query</code>). Valeurs possibles : <ul style="list-style-type: none">• == (égal à, @ est un joker)• === (égal à, @ n'est pas un joker)• != (non égal à, @ est un joker)• !== (non égal à, @ n'est pas un joker)• < (inférieur à)• <= (less than or equal to)• > (supérieur à)• >= (supérieur ou égal à)• IN (inclus dans)• % (contient un mot-clé)
result	Variant	Valeur devant être gérée par le champ calculé. Passez <code>Null</code> dans cette propriété si vous voulez laisser 4D exécuter la requête par défaut (toujours séquentielle pour les champs calculés).

Si la fonction retourne une valeur dans `$result` et qu'une autre valeur est attribuée à la propriété `$event.result`, la priorité est donnée à `$event.result`.

Exemples

- Requête sur le champ calculé `fullName`.

```

Function query fullName($event : Object)->$result : Object

    var $fullname; $firstname; $lastname; $query : Text
    var $operator : Text
    var $p : Integer
    var $parameters : Collection

    $operator:=$event.operator
    $fullname:=$event.value

    $p:=Position(" "; $fullname)
    If ($p>0)
        $firstname:=Substring($fullname; 1; $p-1)+"@"
        $lastname:=Substring($fullname; $p+1)+"@"
        $parameters:=New collection($firstname; $lastname) //collection de deux éléments
    Else
        $fullname:=$fullname+"@"
        $parameters:=New collection($fullname) // collection d'un seul élément
    End if

    Case of
    : ($operator=="==") | ($operator=="===")
        If ($p>0)
            $query:="(firstName = :1 and lastName = :2) or (firstName = :2 and lastName = :1)"
        Else
            $query:="firstName = :1 or lastName = :1"
        End if
    : ($operator!="!=")
        If ($p>0)
            $query:="firstName != :1 and lastName != :2 and firstName != :2 and lastName != :1"
        Else
            $query:="firstName != :1 and lastName != :1"
        End if
    End case

    $result:=New object("query"; $query; "parameters"; $parameters)

```

A noter que l'utilisation de placeholders dans les requêtes basées sur la saisie de texte par l'utilisateur est recommandée pour des raisons de sécurité (voir la description de [query\(\)](#)).

Code d'appel, par exemple :

```
$emps:=ds.Employee.query("fullName = :1"; "Flora Pionsin")
```

- Cette fonction gère les requêtes sur le champ calculé `age` et retourne un objet avec des paramètres :

```

Function query age($event : Object)->$result : Object

    var $operator : Text
    var $age : Integer
    var $_ages : Collection

    $operator:=$event.operator

    $age:=Num($event.value) // entier
    $d1:=Add to date(Current date; -$age-1; 0; 0)
    $d2:=Add to date($d1; 1; 0; 0)
    $parameters:=New collection($d1; $d2)

Case of

    : ($operator=="==")
        $query:="birthday > :1 and birthday <= :2" // après jour1 et avant ou égal à jour2

    : ($operator=="===")

        $query:="birthday = :2" // d2 = deuxième date calculée (= jour anniversaire)

    : ($operator==">=")
        $query:="birthday <= :2"

    //... autres opérateurs

End case

If (Undefined($event.result))
    $result:=New object
    $result.query:=$query
    $result.parameters:=$parameters
End if

```

Code d'appel, par exemple :

```

// personnes âgées de 20 à 21 ans (-1 jour)
$twenty:=people.query("age = 20") // appelle le cas "==" 

// personnes âgées de 20 ans aujourd'hui
$twentyToday:=people.query("age === 20") // équivalent à people.query("age is 20")

```

Function orderBy <attributeName>

Syntaxe

```

Function orderBy <attributeName>($event : Object)
Function orderBy <attributeName>($event : Object)-> $result : Text

// code

```

La fonction `orderBy` s'exécute chaque fois que le champ calculé doit être ordonné. Elle permet de trier le champ calculé. Par exemple, vous pouvez trier `fullName` sur les prénoms puis les noms, ou inversement. Lorsque la fonction `orderBy` n'est pas implémentée pour un champ calculé, le tri est toujours séquentiel (basé sur l'évaluation de toutes les valeurs à l'aide de la fonction `get <AttributeName>`).

L'appel d'une fonction `orderBy` sur des champs calculés de type Entity class ou Entity selection class n'est pas pris en charge.

Les propriétés du paramètre `$event` sont les suivantes :

Propriété	Type	Description
<code>attributeName</code>	Text	Nom du champ calculé
<code>dataClassName</code>	Text	Nom de la dataclass
<code>kind</code>	Text	" <code>orderBy</code> "
<code>value</code>	Variant	Valeur à gérer par le champ calculé
<code>operator</code>	Text	" <code>desc</code> " or " <code>asc</code> " (default)
<code>descending</code>	Booléen	<code>true</code> pour l'ordre décroissant, <code>false</code> pour l'ordre croissant
<code>result</code>	Variant	Valeur devant être gérée par le champ calculé. Passez <code>Null</code> si vous voulez laisser 4D exécuter le tri par défaut.

Vous pouvez utiliser soit l'opérateur , soit la propriété `descending`. C'est essentiellement une question de style de programmation (voir les exemples).

Vous pouvez retourner la chaîne `orderBy` soit dans la propriété de l'objet `$event.result`, soit dans le résultat de la fonction `$result`. Si la fonction retourne une valeur dans `$result` et qu'une autre valeur est attribuée à la propriété `$event.result`, la priorité est donnée à `$event.result`.

Exemple

Vous pouvez saisir du code conditionnel :

```
Function orderBy fullName($event : Object)-> $result : Text
  If ($event.descending=True)
    $result:="firstName desc, lastName desc"
  Else
    $result:="firstName, lastName"
  End if
```

Vous pouvez également saisir du code compact :

```
Function orderBy fullName($event : Object)-> $result : Text
  $result:="firstName "+$event.operator+", lastName "+$event.operator
```

Le code conditionnel est nécessaire dans certains cas :

```
Function orderBy age($event : Object)-> $result : Text
  If ($event.descending=True)
    $result:="birthday asc"
  Else
    $result:="birthday desc"
  End if
```

Attributs de type alias

Aperçu

An alias attribute is built above another attribute of the data model, named `target` attribute. The `target` attribute can belong to a related dataclass (available through any number of relation levels) or to the same dataclass. An alias attribute stores no data, but the path to its `target` attribute. You can define as many alias attributes as you want in a dataclass.

Alias attributes are particularly useful to handle N to N relations. They bring more readability and simplicity in the code and in queries by allowing to rely on business concepts instead of implementation details.

How to define alias attributes

You create an alias attribute in a dataclass by using the `Alias` keyword in the [entity class](#) of the dataclass.

```
Alias <attributeName> <targetPath>
```

Syntaxe

```
{exposed} Alias <attributeName> <targetPath>
```

`attributeName` must comply with [standard rules for property names](#).

`targetPath` is an attribute path containing one or more levels, such as "employee.company.name". If the `target` attribute belongs to the same dataclass, `targetPath` is the attribute name.

An alias can be used as a part of a path of another alias.

A [computed attribute](#) can be used in an alias path, but only as the last level of the path, otherwise, an error is returned. For example, if "fullName" is a computed attribute, an alias with path "employee.fullName" is valid.

ORDA alias attributes are not exposed by default. You must add the `exposed` keyword before the `Alias` keyword if you want the alias to be available to remote requests.

Using alias attributes

Alias attributes are read-only (except when based upon a scalar attribute of the same dataclass, see the last example below). They can be used instead of their target attribute path in class functions such as:

Function
dataClass.query() , entitySelection.query()
entity.toObject()
entitySelection.toCollection()
entitySelection.extract()
entitySelection.orderBy()
entitySelection.orderByFormula()
entitySelection.average()
entitySelection.count()
entitySelection.distinct()
entitySelection.sum()
entitySelection.min()
entitySelection.max()
entity.diff()
entity.touchedAttributes()

Keep in mind that alias attributes are calculated on the server. In remote configurations, updating alias attributes in entities requires that entities are reloaded from the server.

Alias properties

Alias attribute `kind` is "alias".

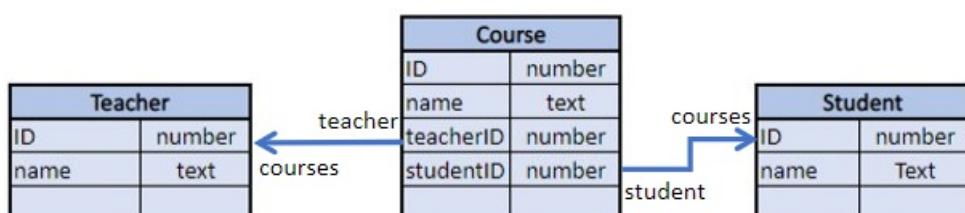
An alias attribute inherits its data `type` property from the target attribute:

- if the target attribute `kind` is "storage", the alias data type is of the same type,
- if the target attribute `kind` is "relatedEntity" or "relatedEntities", the alias data type is of the `4D.Entity` or `4D.EntitySelection` type ("classnameEntity" or "classnameSelection").

Alias attributes based upon relations have a specific `path` property, containing the path of their target attributes. Alias attributes based upon attributes of the same dataclass have the same properties as their target attributes (and no `path` property).

Exemples

Considering the following model:



In the Teacher dataclass, an alias attribute returns all students of a teacher:

```
// cs.TeacherEntity class

Class extends Entity

Alias students courses.student //relatedEntities
```

In the Student dataclass, an alias attribute returns all teachers of a student:

```
// cs.StudentEntity class

Class extends Entity

Alias teachers courses.teacher //relatedEntities
```

In the Course dataclass:

- an alias attribute returns another label for the "name" attribute
- an alias attribute returns the teacher name
- an alias attribute returns the student name

```
// cs.CourseEntity class

Class extends Entity

Exposed Alias courseName name //scalar
Exposed Alias teacherName teacher.name //scalar value
Exposed Alias studentName student.name //scalar value
```

You can then execute the following queries:

```
// Find course named "Archaeology"
ds.Course.query("courseName = :1";"Archaeology")

// Find courses given by the professor Smith
ds.Course.query("teacherName = :1";"Smith")

// Find courses where Student "Martin" assists
ds.Course.query("studentName = :1";"Martin")

// Find students who have M. Smith as teacher
ds.Student.query("teachers.name = :1";"Smith")

// Find teachers who have M. Martin as Student
ds.Teacher.query("students.name = :1";"Martin")
// Note that this very simple query string processes a complex
// query including a double join, as you can see in the queryPlan:
// "Join on Table : Course : Teacher.ID = Course.teacherID,
// subquery:[ Join on Table : Student : Course.studentID = Student.ID,
// subquery:[ Student.name === Martin]]"
```

You can also edit the value of the `courseName` alias:

```
// Rename a course using its alias attribute
$arch:=ds.Course.query("courseName = :1";"Archaeology")
$arch.courseName:="Archaeology II"
$arch.save() //courseName and name are "Archaeology II"
```

Fonctions exposées et non exposées

For security reasons, all of your data model class functions and alias attributes are not exposed (i.e., private) by default to remote requests.

Les requêtes à distance incluent :

- Les requêtes envoyées par des applications 4D distantes connectées via `Open datastore`
- Les requêtes REST

Les requêtes client/serveur 4D standard ne sont pas impactées. Les fonctions de classe de modèle de données sont toujours disponibles dans cette architecture.

Une fonction qui n'est pas exposée n'est pas disponible sur les applications distantes et ne peut être appelée sur aucune instance d'objet à partir d'une requête REST. Si une application distante tente d'accéder à une fonction non exposée, l'erreur «-10729 - Méthode membre inconnue» est retournée.

Pour permettre à une fonction de classe de modèle de données d'être appelée par une requête distante, vous devez la déclarer explicitement à l'aide du mot-clé `exposed`. La syntaxe formelle est la suivante :

```
// déclarer une fonction exposée
exposed Function <functionName>
```

Le mot-clé `exposed` ne peut être utilisé qu'avec les fonctions de classe du modèle de données. S'il est utilisé avec une fonction de [classe utilisateur standard](#), il est ignoré et une erreur est retournée par le compilateur.

Exemple

Vous voulez qu'une fonction exposée utilise une fonction privée dans une classe dataclass :

```
Class extends DataClass

//Fonction publique
exposed Function registerNewStudent($student : Object) -> $status : Object

var $entity : cs.StudentsEntity

$entity:=ds.Students.new()
$entity.fromObject($student)
$entity.school:=This.query("name=:1"; $student.schoolName).first()
$entity.serialNumber:=This.computeSerialNumber()
$status:=$entity.save()

//fonction (privée) non exposée
Function computeIDNumber()-> $id : Integer
//calculer un nouveau numéro d'ID
$id:=...
```

Lorsque le code est appelé :

```

var $remoteDS; $student; $status : Object
var $id : Integer

$remoteDS:=Open datastore(New object("hostname"; "127.0.0.1:8044"); "students")
$student:=New object("firstname"; "Mary"; "lastname"; "Smith"; "schoolName"; "Math school")

$status:=$remoteDS.Schools.registerNewStudent($student) // OK
$id:=$remoteDS.Schools.computeIDNumber() // Erreur "Unknown member method"

```

Fonctions locales

Par défaut dans l'architecture client/serveur, les fonctions de modèle de données ORDA sont exécutées sur le serveur. Il garantit généralement les meilleures performances puisque seuls la requête de fonction et le résultat sont envoyés sur le réseau.

Cependant, il peut arriver qu'une fonction soit entièrement exécutable côté client (par exemple, lorsqu'elle traite des données qui se trouvent déjà dans le cache local). Dans ce cas, vous pouvez enregistrer les requêtes sur le serveur et ainsi améliorer les performances de l'application en saisissant le mot-clé `local`. La syntaxe formelle est la suivante :

```

// déclarer une fonction à exécuter localement en client/serveur
local Function <functionName>

```

Avec ce mot-clé, la fonction sera toujours exécutée côté client.

Le mot-clé `local` ne peut être utilisé qu'avec les fonctions de classe du modèle de données. S'il est utilisé avec une fonction de [classe utilisateur standard](#), il est ignoré et une erreur est retournée par le compilateur.

A noter que la fonction fonctionnera même si elle nécessite d'accéder au serveur (par exemple si le cache ORDA est expiré). Toutefois, il est fortement recommandé de s'assurer que la fonction locale n'accède pas aux données sur le serveur, sinon l'exécution locale pourrait n'apporter aucun avantage en termes de performances. Une fonction locale qui génère de nombreuses requêtes au serveur est moins efficace qu'une fonction exécutée sur le serveur qui ne retournerait que les valeurs résultantes. Prenons l'exemple suivant, avec une fonction sur l'entité Schools :

```

// Obtenir les élèves les plus jeunes
// Utilisation inappropriée du mot-clé local
local Function getYoungest
    var $0 : Object
    $0:=This.students.query("birthDate >= :1"; !2000-01-01!).orderBy("birthDate desc").slice(0; 5)

```

- sans le mot clé `local`, le résultat est donné en une seule requête
- avec le mot-clé `local`, 4 requêtes sont nécessaires : une pour obtenir les élèves de l'entité Schools, une pour la `query()`, une pour le `orderBy()` et une pour la `slice()`. Dans cet exemple, l'utilisation du mot-clé `local` est inappropriée.

Exemples

Calcul de l'âge

Considérons une entité avec un attribut `birthDate`. Nous souhaitons définir une fonction `age()` qui serait appelée dans une list box. Cette fonction peut être exécutée sur le client, ce qui évite de déclencher une requête au serveur pour chaque ligne de la list box.

Dans la classe `StudentsEntity` :

```

Class extends Entity

local Function age() -> $age: Variant

If (This.birthDate#!00-00-00!)
    $age:=Year of(Current date)-Year of(This.birthDate)
Else
    $age:=Null
End if

```

Vérification des attributs

Nous souhaitons vérifier la cohérence des attributs d'une entité chargée sur le client et mise à jour par l'utilisateur, avant de demander au serveur de les enregistrer.

Sur la classe *StudentsEntity*, la fonction locale `checkData()` vérifie l'âge de l'étudiant :

```

Class extends Entity

local Function checkData() -> $status : Object

$status:=New object("success"; True)
Case of
    : (This.age())=Null
        $status.success:=False
        $status.statusText:="The birthdate is missing"

    :((This.age() <15) | (This.age()>30) )
        $status.success:=False
        $status.statusText:="The student must be between 15 and 30 – This one is "+String(This.age())
End case

```

Code d'appel :

```

var $status : Object

//Form.student est chargé avec tous ses attributs et mis à jour sur un Form
$status:=Form.student.checkData()
If ($status.success)
    $status:=Form.student.save() // appelle le serveur
End if

```

Prise en charge en IDE 4D

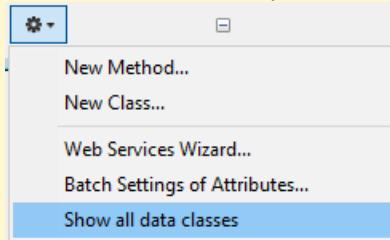
Fichiers de classe (class files)

Une classe utilisateur ORDA de modèle de données est définie en ajoutant, au [même emplacement que les fichiers de classe usuels](#) (c'est-à-dire dans le dossier `/Sources/Classes` du dossier projet), un fichier `.4dm` avec le nom de la classe. Par exemple, une classe d'entité pour la dataclass `Utilities` sera définie via un fichier `UtilitiesEntity.4dm`.

Créer des classes

4D crée préalablement et automatiquement des classes vides en mémoire pour chaque objet de modèle de données disponible.

Par défaut, les classes ORDA vides ne sont pas affichées dans l'Explorateur. Vous devez les afficher en sélectionnant Afficher toutes les dataclasses dans le menu d'options de l'Explorateur :



Les classes d'utilisateurs ORDA ont une icône différente des autres classes. Les classes vides sont grises :

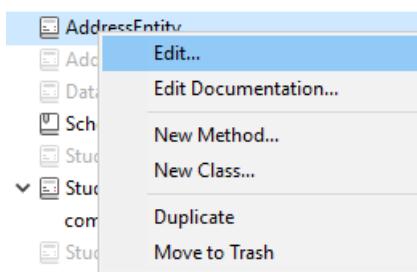
Pour créer un fichier de classe ORDA, il vous suffit de double-cliquer sur la classe prédéfinie correspondante dans l'Explorateur. 4D crée le fichier de classe et ajoute le code `extends`. Par exemple, pour une classe Entity :

Class extends Entity

Une fois qu'une classe est définie, son nom n'est plus grisé dans l'Explorateur.

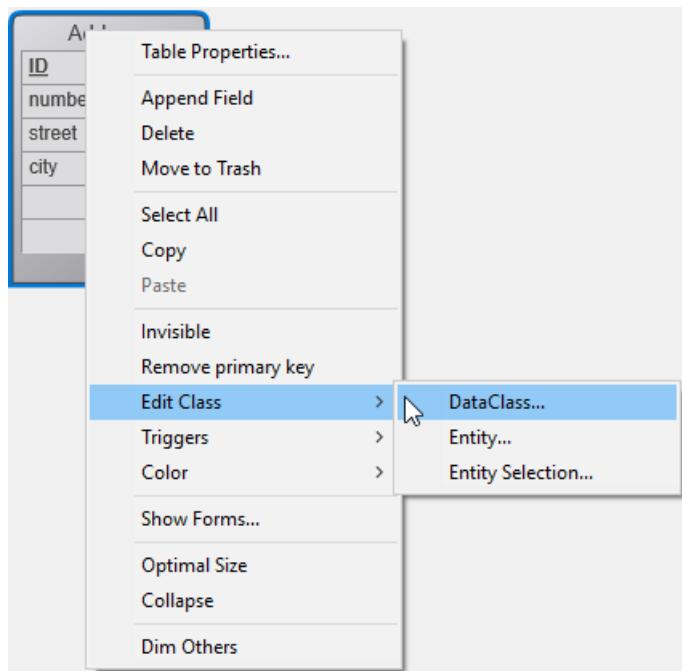
Modifier des classes

Pour ouvrir une classe ORDA définie dans l'éditeur de méthode 4D, sélectionnez ou double-cliquez sur un nom de classe ORDA et utilisez Editer... dans le menu contextuel/menu d'options de la fenêtre d'Explorateur :



Pour les classes ORDA basées sur le datastore local (`ds`), vous pouvez accéder directement au code de la classe depuis

la fenêtre de 4D Structure :



Éditeur de méthode

Dans l'éditeur de méthode de 4D, les variables saisies comme une classe ORDA bénéficient automatiquement des fonctionnalités d'auto-complétion. Exemple avec une variable de classe Entity :

A screenshot of the 4D Method Editor. The code being typed is:

```
var $student : cs.StudentsEntity  
$student:=ds.Students.all().first()  
$student.
```

An auto-completion dropdown is open at the end of the line, showing the following suggestions:

- ID
- name
- clone
- computeAverage** (highlighted with a red rectangle)
- diff
- drop
- first
- fromObject

Travailler avec des données

Dans ORDA, vous accédez aux données via des [entités](#) (entities) et des [sélections d'entités](#) (entity selections). Ces objets vous permettent de créer, mettre à jour, rechercher ou trier les données du datastore.

Créer une entité

Il existe deux façons de créer une nouvelle entité dans une dataclass :

- Les entités étant des références à des enregistrements de base de données, vous pouvez créer des entités en créant des enregistrements en utilisant le langage 4D "classique", puis les référencer avec des méthodes ORDA telles que `entity.next()` ou `entitySelection.first()`.
- Vous pouvez également créer une entité à l'aide de la méthode `dataClass.new()`.

Gardez à l'esprit que l'entité est créée uniquement en mémoire. Si vous souhaitez l'ajouter à la banque de données, vous devez appeler la méthode `entity.save()`.

Les attributs de l'entité sont directement disponibles en tant que propriétés de l'objet entité. Pour plus d'informations, reportez-vous à [Utilisation des attributs d'entité](#).

Par exemple, nous voulons créer une nouvelle entité dans la dataclass "Employee" dans le datastore courant avec "John" et "Dupont" affectés aux attributs de prénom et de nom :

```
var $myEntity : cs.EmployeeEntity
$myEntity:=ds.Employee.new() //Créer un nouvel objet de type entité
$myEntity.name:="Dupont" //assigner 'Dupont' à l'attribut 'name'
$myEntity.firstname:="John" //assigner 'John' à l'attribut 'firstname'
$myEntity.save() //sauvegarder l'entité
```

Une entité est définie uniquement dans le processus où elle a été créée. Vous ne pouvez pas, par exemple, stocker une référence à une entité dans une variable interprocess et l'utiliser dans un autre processus.

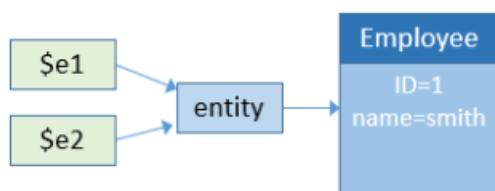
Entités et références

Une entité contient une référence à un enregistrement 4D. Différentes entités peuvent référencer le même enregistrement 4D. De plus, comme une entité peut être stockée dans une variable objet 4D, différentes variables peuvent contenir une référence à la même entité.

Si vous exécutez le code suivant :

```
var $e1; $e2 : cs.EmployeeEntity
$e1:=ds.Employee.get(1) //accéder à l'employé avec ID 1
$e2:=$e1
$e1.name:="Hammer"
//les variables $e1 et $e2 partagent la référence à la même entité
//$e2.name contient "Hammer"
```

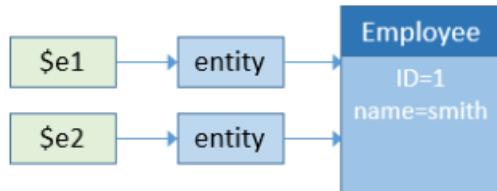
Ceci est illustré par le graphique suivant :



Maintenant, si vous exécutez :

```
var $e1; $e2 : cs.EmployeeEntity
$e1:=ds.Employee.get(1)
$e2:=ds.Employee.get(1)
$e1.name:="Hammer"
//la variable $e1 contient une référence vers une entité
//variable $e2 contient une autre référence vers une autre entité
//$e2.name contient "smith"
```

Ceci est illustré par le graphique suivant :



A noter cependant que les entités font référence au même enregistrement. Dans tous les cas, si vous appelez la méthode `entity.save()`, l'enregistrement sera mis à jour (sauf en cas de conflit, voir [Verrouillage d'entité](#)).

En fait, `$e1` et `$e2` ne sont pas l'entité elle-même, mais des références à l'entité. Cela signifie que vous pouvez les passer directement à n'importe quelle fonction ou méthode, et qu'elle agira comme un pointeur, et plus rapidement qu'un pointeur 4D. Par exemple :

```
For each($entity;$selection)
    do_Capitalize($entity)
End for each
```

Et la méthode est :

```
$entity:=$1
$name:=$entity.lastname
If(Not($name=NULL))
    $name:=Uppercase(Substring($name;1;1))+Lowercase(Substring($name;2))
End if
$entity.lastname:=$name
```

Vous pouvez gérer les entités comme n'importe quel autre objet dans 4D et passer leurs références directement en tant que [paramètres](#).

Avec les entités, il n'y a pas de notion de "enregistrement courant" comme dans le langage classique de 4D. Vous pouvez utiliser autant d'entités que nécessaire, en même temps. Il n'existe pas non plus de verrouillage automatique d'une entité (voir [Verrouillage d'une entité](#)). Lorsqu'une entité est chargée, elle utilise le mécanisme de [lazy loading](#), ce qui signifie que seules les informations nécessaires sont chargées. Néanmoins, en mode client/serveur, l'entité peut être automatiquement chargée directement si nécessaire.

Utilisation des attributs d'entités

Les attributs d'entité stockent les données et mappent les champs correspondants dans la table correspondante. Les attributs d'entité du type de stockage peuvent être définis ou obtenus sous forme de propriétés simples de l'objet entité, tandis que l'entité de type relatedEntity ou relatedEntities renverra une entité ou une sélection d'entité.

Pour plus d'informations sur le type d'attribut, reportez-vous au paragraphe [Attributs de stockage et de relation](#).

Par exemple, pour définir un attribut de stockage :

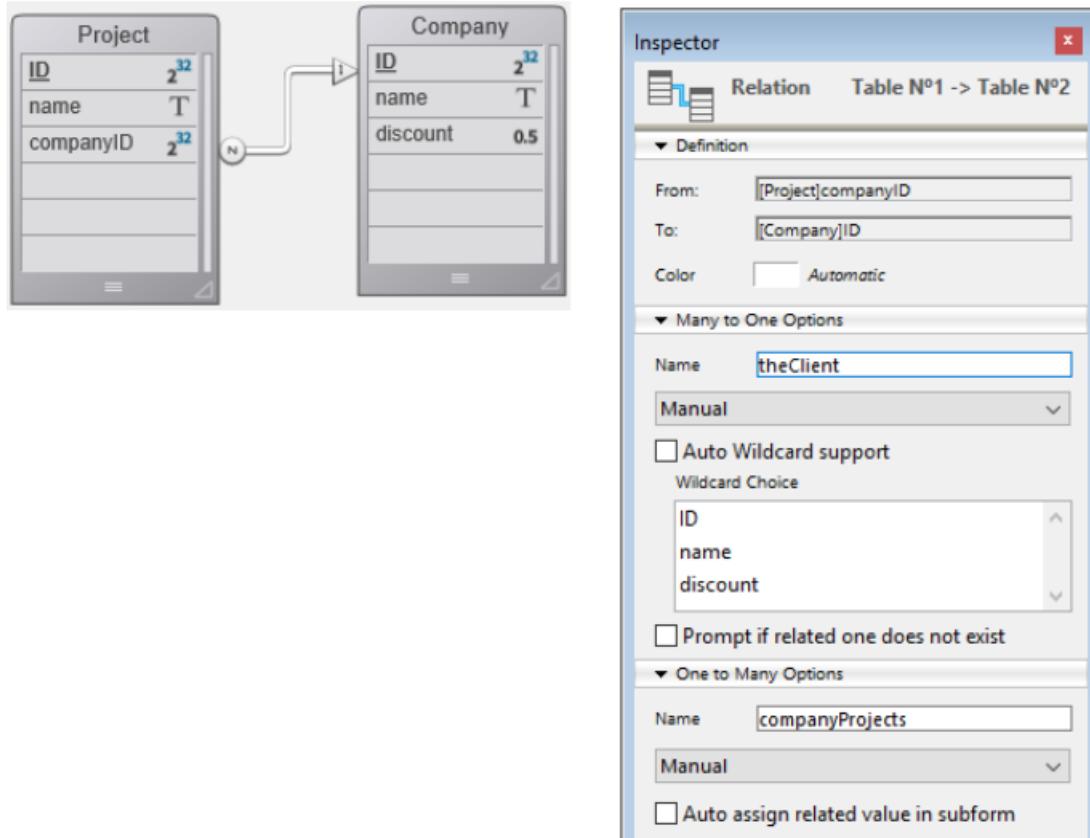
```

$entity:=ds.Employee.get(1) //obtenir l'attribut d'Employee avec l'ID 1
$name:=$entity.lastname //obtenir le nom de l'employé, par exemple "Smith"
$entity.lastname:="Jones" //définir le nom de l'employé
$entity.save() //sauvegarder les modifications

```

Les champs Blob des bases de données (les **blobs scalaires**) sont automatiquement convertis en attributs d'objets blob (**4D.Blob**) lorsqu'ils sont traités par ORDA. Lorsque vous sauvegardez un attribut d'objet blob, gardez à l'esprit que, contrairement à la taille de l'objet blob qui n'est limitée que par la mémoire disponible, la taille du champ Blob est limitée à 2 Go.

L'accès à un attribut associé dépend du type d'attribut. Par exemple, avec la structure suivante :



Vous pouvez accéder aux données via le ou les objets associé(s) :

```

$entity:=ds.Project.all().first().theClient //récupérer l'entité Company associée au projet
$EntitySel:=ds.Company.all().first().companyProjects //récupère la sélection de projets pour l'entreprise

```

Notez que dans l'exemple ci-dessus, *theClient* et *companyProjects* sont des attributs de relation et représentent une relation directe entre les deux dataclasses. Cependant, les attributs de relation peuvent également être créés sur des chemins via des relations à plusieurs niveaux, y compris des références circulaires. Par exemple, considérons la structure suivante :

The screenshot shows the Oracle ADF Inspector tool with the 'Relation' configuration for 'Table N°1 -> Table N°1'. The 'From' field is set to '[Employee]managerID' and the 'To' field is set to '[Employee]ID'. The 'Name' field is set to 'manager'. In the 'One to Many Options' section, 'Name' is set to 'directReports'. A diagram on the right shows a self-referencing relationship named 'manager'.

Chaque employé peut être un manager et peut avoir un manager. Pour obtenir le manager du manager d'un employé, vous pouvez simplement écrire :

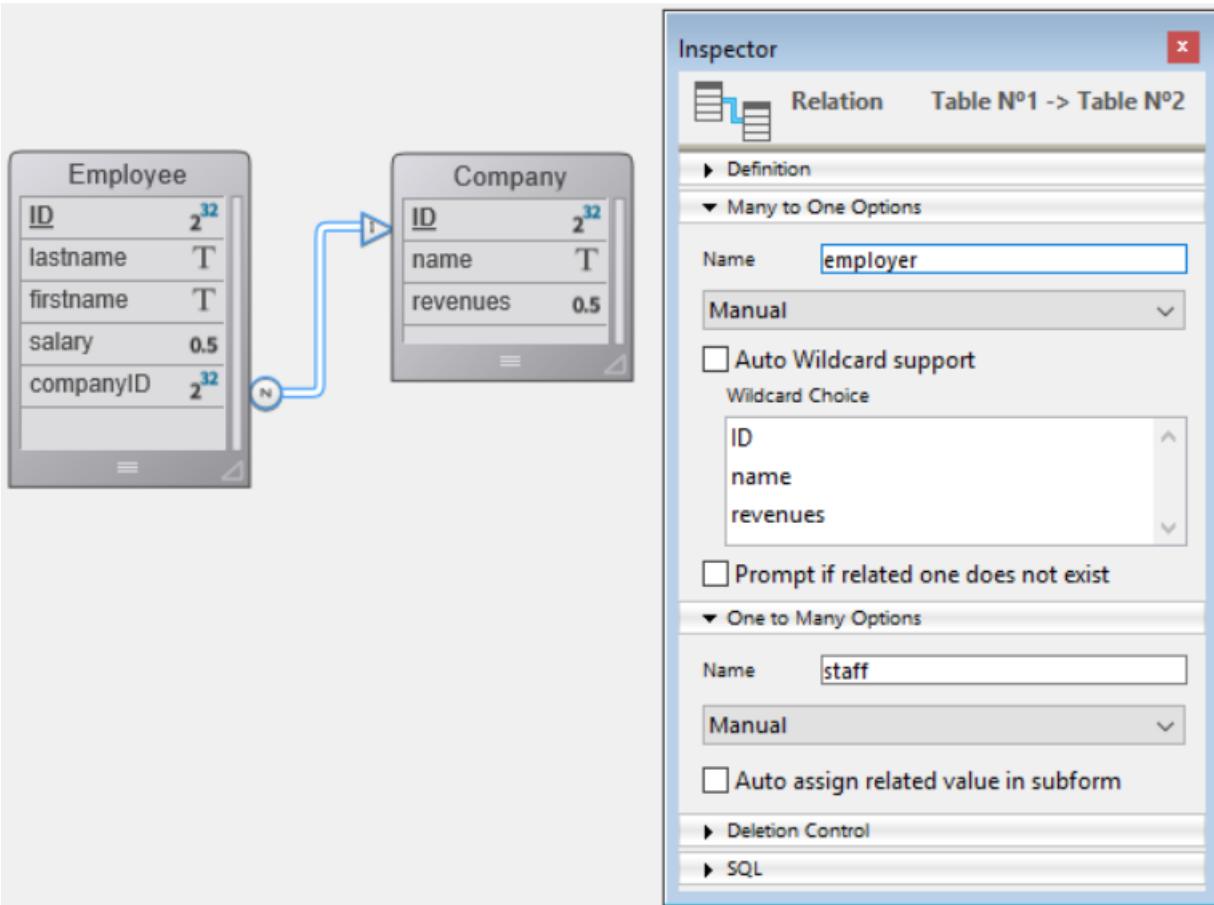
```
$myEmp:=ds.Employee.get(50)
$manLev2:=$myEmp.manager.manager.lastname
```

Assigner des valeurs aux attributs de relation

Dans l'architecture ORDA, les attributs de relation contiennent directement des données liées aux entités :

- Un attribut de relation de type N-> 1 (type relatedEntity) contient une entité
- Un attribut de relation de type 1-> N (type relatedEntities) contient une sélection d'entité

Regardons la structure (simplifiée) suivante :



Dans cet exemple, une entité de la dataclass "Employee" contient un objet de type Entité dans l'attribut "employer" (ou une valeur nulle). Une entité de la dataclass "Company" contient un objet de type EntitySelection dans l'attribut "staff" (ou une valeur nulle).

Dans ORDA, la propriété Automatic ou Manual des relations ne produit aucun effet.

Pour attribuer une valeur directement à l'attribut "employer", vous devez passer une entité existante de la dataclass "Company". Par exemple :

```
$emp:=ds.Employee.new() // créer un employé
$emp.lastname:="Smith" // attribuer une valeur à un attribut
$emp.employer:=ds.Company.query("name =:1";"4D")[0] //attribuer une entité de "company"
$emp.save()
```

4D fournit une fonctionnalité supplémentaire pour saisir un attribut de relation pour une entité N liée à une entité "1": vous passez directement la clé primaire de l'entité "1" lors de l'attribution d'une valeur à l'attribut de relation. Pour que cela fonctionne, passez des données de type Numérique ou Texte (la valeur de la clé primaire) à l'attribut de relation. 4D se charge alors automatiquement de rechercher l'entité correspondante dans la dataclass. Par exemple :

```
$emp:=ds.Employee.new()
$emp.lastname:="Wesson"
$emp.employer:=2 // attribuer une clé primaire à l'attribut relation
//4D recherche l'entreprise dont la clé primaire (dans ce cas, son ID) est 2
//et l'attribue à l'employé
$emp.save()
```

Ceci est particulièrement utile lorsque vous importez un grand nombre de données à partir d'une base de données relationnelle. Ce type d'import contient généralement une colonne "ID", qui référence une clé primaire que vous pouvez ensuite affecter directement à un attribut de relation.

Cela signifie également que vous pouvez attribuer des clés primaires dans les N entités sans que les entités correspondantes aient déjà été créées dans la 1e classe de datastore. Si vous affectez une clé primaire qui n'existe pas

dans la classe de datastore associée, elle est néanmoins stockée et affectée par 4D dès que cette entité "1" est créée.

Vous pouvez attribuer ou modifier la valeur d'un attribut d'entité associé "1" à partir de la dataclass "N" directement via l'attribut associé. Par exemple, si vous souhaitez modifier l'attribut de nom d'une entité "Company" associée d'une entité "Employee", vous pouvez écrire :

```
$emp:=ds.Employee.get(2) // charge l'entité Employee avec la clé primaire 2  
$emp.employer.name:="4D, Inc." //modifier l'attribut de nom de la société associée  
$emp.employer.save() //sauvegarder l'attribut associé  
//l'entité associée est mise à jour
```

Créer une sélection d'entité (entity selection)

Vous pouvez créer un objet de type `entity selection` comme suit :

- Lancez une requête sur les entités `dans une dataclass ou dans une sélection d'entités existante` ;
- Utilisez la fonction de dataclass `.all()` pour sélectionner toutes les entités d'une dataclass;
- Utilisez la commande `Create entity selection` ou la fonction de dataclass `.newSelection()` pour créer une sélection d'entités vide;
- Utilisez la fonction `.copy()` pour dupliquer une sélection d'entités existante;
- Utilisez l'une des diverses fonctions de `Entity selection class` qui retourne une nouvelle sélection d'entité, telle que `entitySelection.or()` ;
- Utilisez un attribut de relation de type "related entities" ("entités liées") (voir ci-dessous).

Vous pouvez créer et utiliser simultanément autant de sélections d'entités différentes que vous le souhaitez pour une dataclass. A noter qu'une sélection d'entité ne contient que des références à des entités. Différentes sélections d'entités peuvent contenir des références vers les mêmes entités.

Entity selections partageables ou modifiables

Une "entity selection" peut être partageable (lisible par plusieurs processus, mais non modifiable après sa création) ou modifiable (supporte la fonction `.add()`, mais utilisable uniquement par le processus actuel).

Propriétés

Une entity selection partageable a les caractéristiques suivantes :

- elle peut être stockée dans un objet partagé ou une collection partagée, et peut être passée comme paramètre entre plusieurs process ou workers ;
- elle peut être stockée dans plusieurs objets partagés ou collections partagées, ou dans un objet partagé ou une collection partagée qui appartient déjà à un groupe (elle n'a pas de *locking identifier*) ;
- elle ne permet pas d'ajouter de nouvelles entités. Essayer d'ajouter une entité à une entity selection partageable générera une erreur (1637 - Cette entity selection ne peut pas être modifiée). Pour ajouter une entité à une entity selection partageable, vous devez d'abord la transformer en une entity selection non partageable à l'aide de la fonction `.copy()`, avant d'appeler `.add()`.

La plupart des fonctions d'entity selection (telles que `.slice()`, `.and()` ...) prennent en charge les entity selection partageables car elles n'ont pas besoin de modifier l'entity selection d'origine (elles en retournent une nouvelle).

Une entity selection modifiable a les caractéristiques suivantes :

- elle ne peut pas être partagée entre les process, ni être stockée dans un objet partagé ou une collection partagée. Essayer de stocker une entity selection non partageable dans un objet partagé ou une collection partagée générera une erreur (-10721 - Type de valeur non pris en charge dans un objet partagé ou une collection partagée) ;
- elle accepte l'ajout de nouvelles entités, c'est-à-dire qu'elle supporte la fonction `.add()`.

Comment les définir ?

La nature partageable ou modifiable d'une entity selection est définie lors de sa création (elle ne peut pas être modifiée par la suite). Vous pouvez connaître la nature d'une entity selection à l'aide de la fonction `.isAlterable()` ou de la commande `OB Is shared`.

Une nouvelle entity selection est partageable dans les cas suivants :

- la nouvelle entity selection résulte d'une fonction de classe ORDA appliquée à une dataClass : `dataClass.all()`, `dataClass.fromCollection()`, `dataClass.query()`,
- la nouvelle entity selection est fondée sur une relation `entity.attributeName` (par exemple, "company.employees") lorsque `attributeName` est un attribut lié1 vers N mais que l'entity n'appartient pas à une entity selection.
- la nouvelle entity selection est explicitement copiée comme partageable avec `entitySelection.copy()` ou `OB Copy` (c'est-à-dire avec l'option `ck shared`).

Exemple :

```
$myComp:=ds.Company.get(2) // $myComp does not belong to an entity selection  
$employees:=$myComp.employees // $employees is shareable
```

A new entity selection is alterable in the following cases:

- la nouvelle "entity selection" crée un espace vide à l'aide de la fonction `dataClass.newSelection()` ou de la commande `Create entity selection`,
- la nouvelle "entity selection" est explicitement copiée comme modifiable avec `entitySelection.copy()` ou `OB Copy` (c'est-à-dire sans l'option `ck shared`).

Exemple :

```
$toModify:=ds.Company.all().copy() // $toModify is alterable
```

A new entity selection inherits from the original entity selection nature in the following cases:

- Most entity selection functions (such as `.slice()`, `.and()` ...).
- the new entity selection is based upon a relation:
 - `entity.attributeName` (par exemple, "company.employees") lorsque `attributeName` est un attribut lié1 vers N mais que l'entity appartient à une "entity selection" (de même nature que `getSelection`)
 - `entitySelection.attributeName` (e.g. "employees.employer") lorsque `attributeName` est un attribut lié (de même nature que celle de la "entity selection"),
 - `.extract()`, lorsque la collection résultante contient des sélections d'entités (de même nature que l'entity selection").

Voici quelques exemples :

```
$highSal:=ds.Employee.query("salary >= :1"; 1000000)  
// $highSal is shareable because of the query on dataClass  
$comp:=$highSal.employer // $comp is shareable because $highSal is shareable  
  
$lowSal:=ds.Employee.query("salary <= :1"; 10000).copy()  
// $lowSal is alterable because of the copy()  
$comp2:=$lowSal.employer // $comp2 is alterable because $lowSal is alterable
```

Sharing an entity selection between processes (example)

You work with two entity selections that you want to pass to a worker process so that it can send mails to appropriate persons:

```

var $paid; $unpaid : cs.InvoicesSelection
//We get entity selections for paid and unpaid invoices
$paid:=ds.Invoices.query("status=:1"; "Paid")
$unpaid:=ds.Invoices.query("status=:1"; "Unpaid")

//We pass entity selection references as parameters to the worker
CALL WORKER("mailing"; "sendMails"; $paid; $unpaid)

```

The `sendMails` method:

```

#DECLARE ($paid : cs.InvoicesSelection; $unpaid : cs.InvoicesSelection)
var $invoice : cs.InvoicesEntity

var $server; $transporter; $email; $status : Object

//Prepare emails
$server:=New object()
$server.host:="exchange.company.com"
$server.user:="myName@company.com"
$server.password:="my!password"
$transporter:=SMTP New transporter($server)
$email:=New object()
$email.from:="myName@company.com"

//Loops on entity selections
For each($invoice;$paid)
    $email.to:=$invoice.customer.address // email address of the customer
    $email.subject:="Payment OK for invoice # "+String($invoice.number)
    $status:=$transporter.send($email)
End for each

For each($invoice;$unpaid)
    $email.to:=$invoice.customer.address // email address of the customer
    $email.subject:="Please pay invoice # "+String($invoice.number)
    $status:=$transporter.send($email)
End for each

```

Sélections d'entités et attributs de stockage

Tous les attributs de stockage (texte, numérique, booléen, date) sont disponibles en tant que propriétés des sélections d'entités et en tant qu'entités. Lorsqu'il est utilisé avec une sélection d'entité, un attribut scalaire retourne une collection de valeurs scalaires. Par exemple :

```

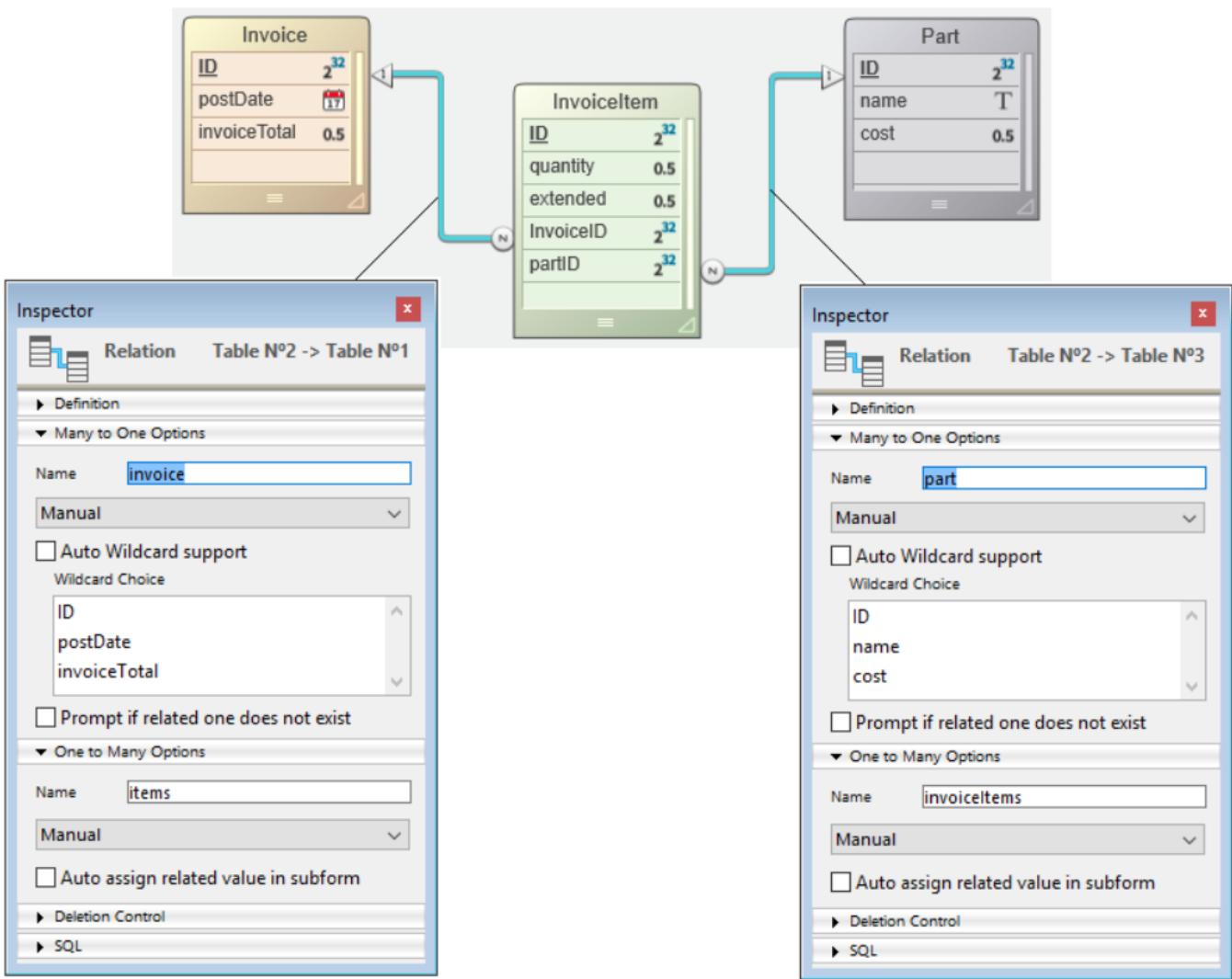
$locals:=ds.Person.query("city = :1";"San Jose") //entity selection of people
$localEmails:=$locals.emailAddress //collection of email addresses (strings)

```

This code returns in `$localEmails` a collection of email addresses as strings.

Sélections d'entités et attributs de relation

Outre la variété de méthodes de recherche, vous pouvez également utiliser des attributs de relation comme propriétés des sélections d'entités pour retourner de nouvelles sélections d'entités. Par exemple, considérons la structure suivante :



```
$myParts:=ds.Part.query("ID < 100") //Return parts with ID less than 100
$myInvoices:=$myParts.invoiceItems.invoice
//All invoices with at least one line item related to a part in $myParts
```

La dernière ligne renverra, dans \$myInvoices, une sélection d'entité de toutes les factures qui ont au moins un poste de facture lié à une partie de la sélection d'entités myParts. Lorsqu'un attribut de relation est utilisé comme propriété d'une entity selection, le résultat est toujours une autre entity selection, même si une seule entité est retournée. Lorsqu'un attribut de relation est utilisé comme propriété d'une sélection d'entité et qu'aucune entité n'est retournée, le résultat est une sélection d'entité vide et non nulle.

Verrouillage d'une entité

Vous devez souvent gérer d'éventuels conflits pouvant survenir lorsque plusieurs utilisateurs ou processus se chargent et tentent de modifier les mêmes entités en même temps. Le verrouillage des enregistrements est une méthodologie utilisée dans les bases de données relationnelles pour éviter les mises à jour incohérentes des données. Le concept consiste soit à verrouiller un enregistrement lors de sa lecture afin qu'aucun autre processus ne puisse le mettre à jour, soit à vérifier lors de la sauvegarde d'un enregistrement qu'un autre processus ne l'a pas modifié depuis sa lecture. Le premier est appelé verrouillage d'enregistrement pessimiste et garantit qu'un enregistrement modifié peut être écrit au détriment du verrouillage des enregistrements pour d'autres utilisateurs. Ce dernier est appelé verrouillage d'enregistrement optimiste et il échange la garantie des priviléges d'écriture sur l'enregistrement contre la flexibilité de décider des priviléges d'écriture uniquement si l'enregistrement doit être mis à jour. Dans le verrouillage d'enregistrement pessimiste, l'enregistrement est verrouillé même s'il n'est pas nécessaire de le mettre à jour. Dans le verrouillage d'enregistrement optimiste, la validité de la modification d'un enregistrement est fixée au moment de la mise à jour.

ORDA vous propose deux modes de verrouillage d'entité :

- un mode automatique "optimiste", adapté à la plupart des applications,
- un mode "pessimiste" permettant de verrouiller les entités avant d'y accéder.

Verrouillage optimiste automatique

Ce mécanisme automatique est basé sur le concept de "verrouillage optimiste" qui est particulièrement adapté aux problématiques des applications web. Ce concept se caractérise par les principes de fonctionnement suivants :

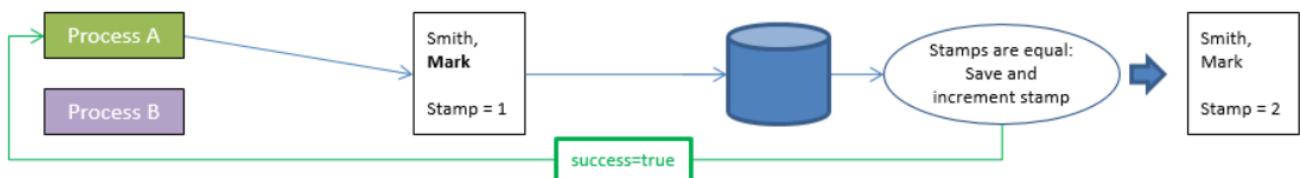
- Toutes les entités peuvent toujours être chargées en lecture-écriture; il n'y a pas de «verrouillage» *a priori* des entités.
- Chaque entité possède un marqueur de verrouillage interne qui est incrémenté à chaque fois qu'il est enregistré.
- Lorsqu'un utilisateur ou un process tente de sauvegarder une entité à l'aide de la méthode `entity.save()`, 4D compare la valeur du marqueur de l'entité à sauvegarder avec celle de l'entité trouvée dans les données (en cas de modification) :
 - Lorsque les valeurs correspondent, l'entité est enregistrée et la valeur du marqueur interne est incrémentée.
 - Lorsque les valeurs ne correspondent pas, cela signifie qu'un autre utilisateur a modifié cette entité entre-temps. La sauvegarde n'est pas effectuée et une erreur est retournée.

Le diagramme suivant illustre le verrouillage optimiste :

1. Deux process chargent la même entité.



2. Le premier process modifie l'entité et valide le changement. La méthode `entity.save()` est appelée. Le moteur 4D compare automatiquement la valeur du marqueur interne de l'entité modifiée avec celle de l'entité stockée dans les données. Puisqu'ils correspondent, l'entité est enregistrée et la valeur de son marqueur est incrémentée.



3. Le deuxième process modifie également l'entité chargée et valide ses modifications. La méthode `entity.save()` est appelée. Etant donné que la valeur de marqueur de l'entité modifiée ne correspond pas à celle de l'entité stockée dans les données, la sauvegarde n'est pas effectuée et une erreur est retournée.



Cela peut également être illustré par le code suivant :

```

$person1:=ds.Person.get(1) //Référence à l'entité
$person2:=ds.Person.get(1) //Autre référence à la même entité
$person1.name:="Bill"
$result:=$person1.save() //$/result.success=true, modification enregistrée
$person2.name:="William"
  
```

Dans cet exemple, nous attribuons à \$person1 une référence à l'entité "person" avec une clé de 1. Nous attribuons ensuite une autre référence de la même entité à la variable \$person2. Avec \$person1, nous modifions le prénom de la personne et sauvegardons l'entité. Lorsque nous essayons de faire de même avec \$person2, 4D vérifie que l'entité sur le disque est la même que lors de la première attribution de la référence dans \$person1. Puisqu'elles ne sont pas

identiques, 4D retourne "faux" dans la propriété "success" et ne sauvegarde pas la deuxième modification.

When this situation occurs, you can, for example, reload the entity from the disk using the `entity.reload()` method so that you can try to make the modification again. The `entity.save()` method also proposes an "automerge" option to save the entity in case processes modified attributes that were not the same.

Record stamps are not used in transactions because only a single copy of a record exists in this context. Whatever the number of entities that reference a record, the same copy is modified thus `entity.save()` operations will never generate stamp errors.

Verrouillage pessimiste

Vous pouvez verrouiller et déverrouiller des entités à la demande lorsque vous accédez aux données. Lorsqu'une entité est verrouillée par un process, elle est chargée en lecture/écriture dans ce process mais elle est verrouillée pour tous les autres process. L'entité peut être chargée uniquement en mode lecture seule dans ces process; ses valeurs ne peuvent pas être modifiées ou enregistrées.

This feature is based upon two methods of the `Entity` class:

- `entity.lock()`
- `entity.unlock()`

Pour plus d'informations, reportez-vous aux descriptions de ces méthodes.

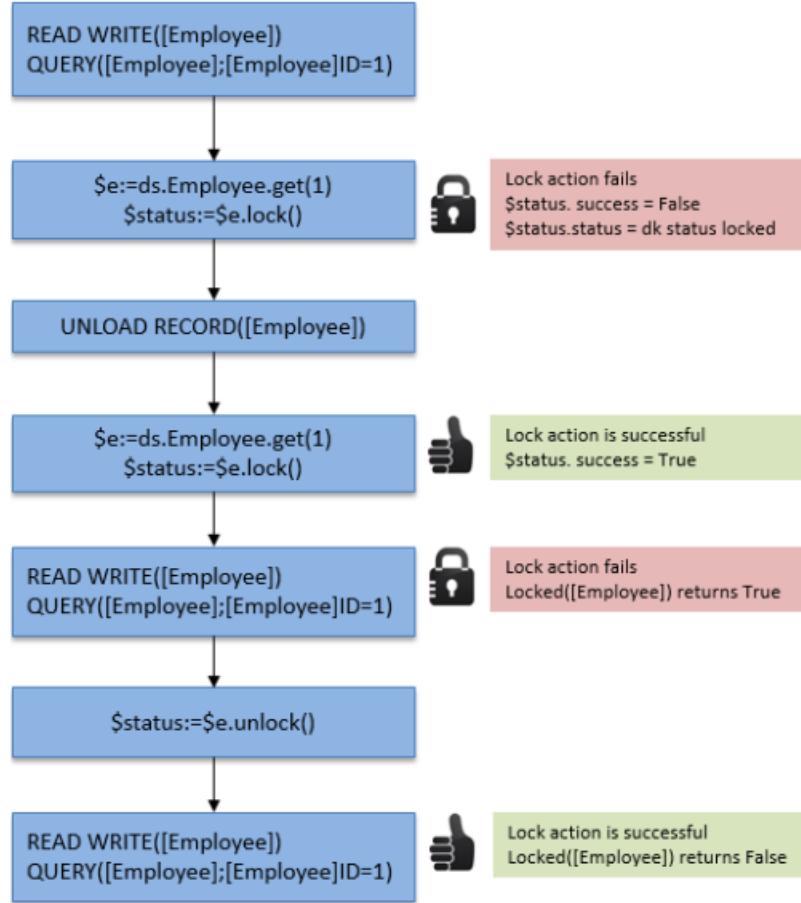
Pessimistic locks can also be handled through the [REST API](#).

Utilisation simultanée des verrouillages classiques 4D et des verrouillages pessimistes ORDA

L'utilisation des commandes classiques et ORDA pour le verrouillage des enregistrements est basé sur les principes suivants :

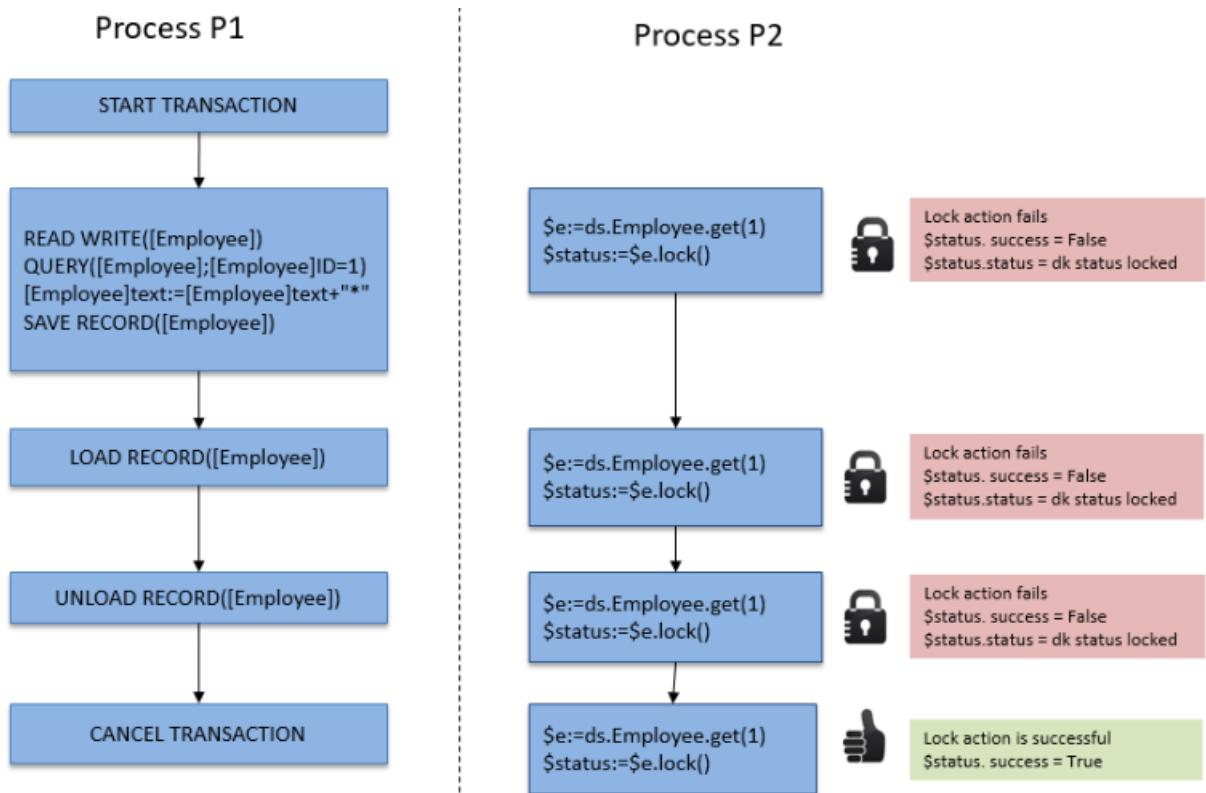
- Un verrouillage défini avec une commande 4D classique sur un enregistrement empêche ORDA de verrouiller l'entité correspondant à l'enregistrement.
- Un verrouillage défini avec ORDA sur une entité empêche les commandes 4D classiques de verrouiller l'enregistrement correspondant à l'entité.

Ces principes sont illustrés dans le diagramme suivant :



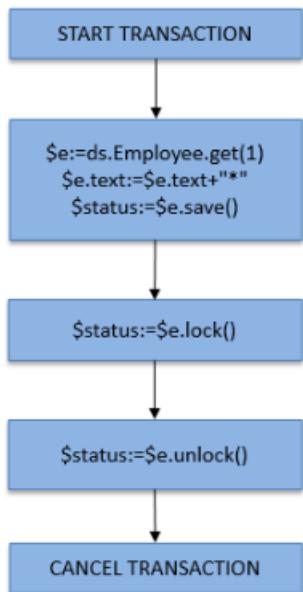
Les verrouillages de transaction s'appliquent également aux commandes classiques et aux commandes ORDA. Dans une application multiprocès ou multi-utilisateurs, un verrouillage défini dans une transaction sur un enregistrement par une commande classique aura pour effet d'empêcher tout autre processus de verrouiller les entités liées à cet enregistrement (ou inversement), jusqu'à ce que la transaction soit validée ou annulée.

- Exemple avec un verrouillage défini par une commande classique :

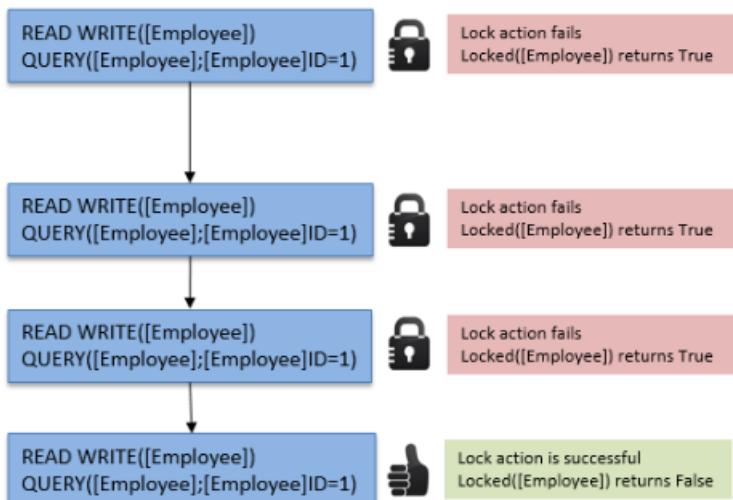


- Exemple avec un verrouillage défini par une méthode ORDA :

Process P1



Process P2



Utiliser un datastore distant

Un [datastore](#) exposé sur une application 4D Server est accessible simultanément via différents clients :

- Les applications 4D distantes utilisant ORDA pour accéder au datastore principal à l'aide de la commande `ds`. A noter que l'application 4D distante peut toujours accéder à la base de données en mode classique. Ces accès sont gérés par le serveur d'applications 4D.
- D'autres applications 4D (4D Remote, 4D Server) ouvrant une session sur le datastore distant via la commande [Open datastore](#). Ces accès sont transmis par le serveur HTTP REST.
- Les requêtes [4D for iOS ou 4D for Android](#) pour la mise à jour des applications mobiles. Ces accès sont remis par le serveur HTTP.

Ouverture des sessions

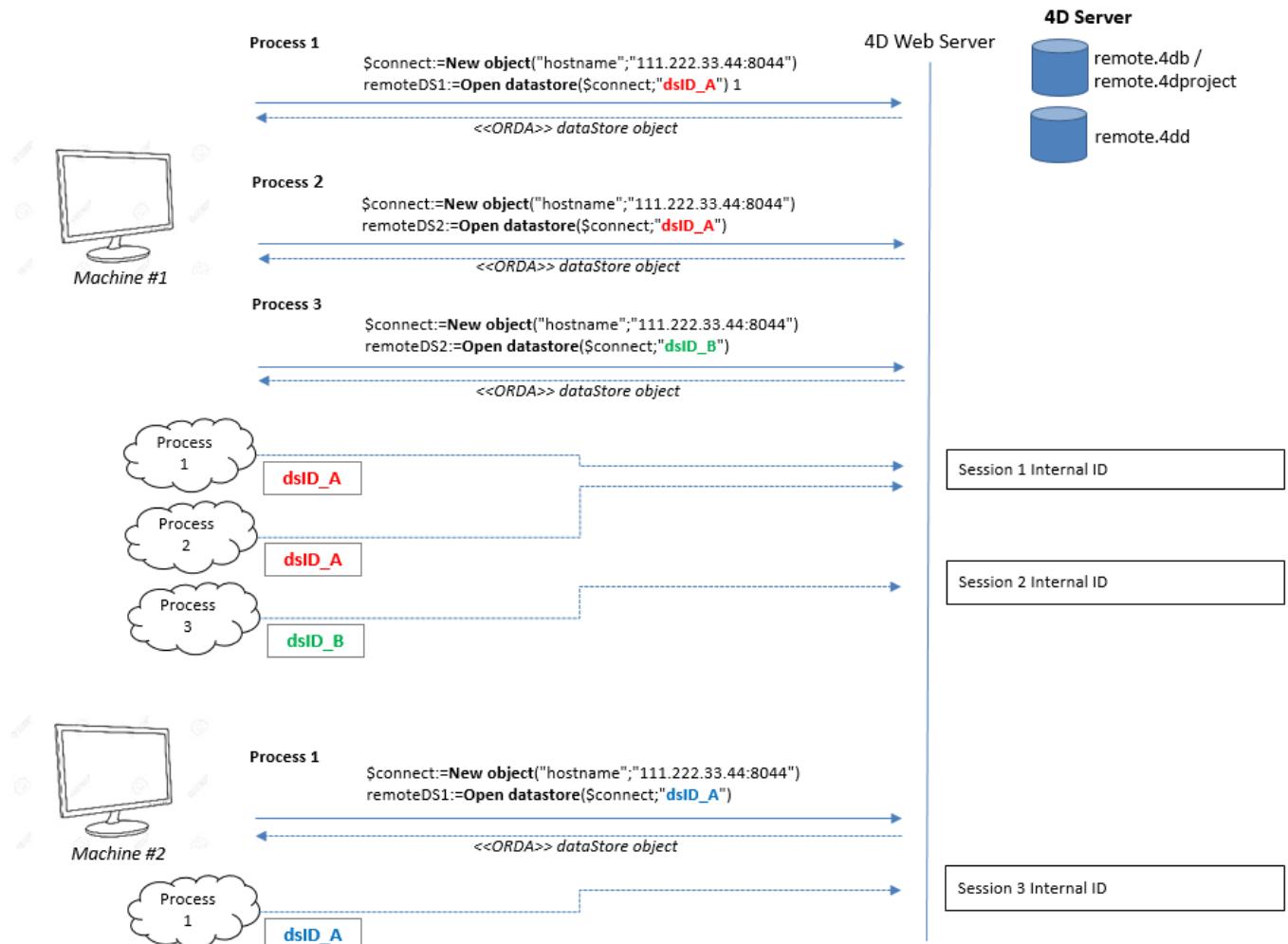
Lorsque vous travaillez avec un datastore distant référencé par des appels à la commande `Open datastore`, la connexion entre les process qui effectuent la requête et le datastore distant est gérée par des sessions.

Une session est créée sur le datastore distant pour gérer la connexion. Cette session est identifiée à l'aide d'un ID de session interne, associé au `localID` de l'application 4D. Cette session gère automatiquement l'accès aux données, aux sélections d'entités ou aux entités.

Le `localID` est local à la machine qui se connecte au datastore distant, ce qui signifie :

- Que si d'autres process de la même application doivent accéder au même datastore distant, ils peuvent utiliser le même `localID` et partager alors la même session.
- Que si un autre process de la même application ouvre le même datastore distant, mais avec un autre `localID`, il créera une nouvelle session sur le datastore distant.
- Que si un autre poste se connecte au même datastore distant avec le même `localID`, il créera une autre session avec un autre cookie.

Ces principes sont illustrés dans les graphiques suivants :



Pour les sessions ouvertes par des requêtes REST, veuillez consulter la page [Utilisateurs et sessions](#).

Visionnage des sessions

Les process qui gèrent les sessions d'accès aux datastore apparaissent dans la fenêtre d'administration de 4D Server :

- <process name>nom : "Gestionnaire REST : < nom du process >"
- type : type Worker Server HTTP
- session : le nom de session est le nom d'utilisateur passé à la commande `Open datastore`.

Dans l'exemple suivant, deux process sont en cours d'exécution pour la même session :

EmpComp - 4D Server Administration							
Monitor	Users (1)	Processes (23)	Maintenance	Application Server	SQL Server	HTTP Server	Real Time Monitor
Display processes by groups			Users processes (0)		4D Processes (16)		Spare processes (7)
Process name	Session / Info	Type	Num	State	CPU Time	Activity	
Client Manager	-	Application server	3	Waiting for event	00:00:01	0 %	
DB4D CRON	-	DB4D Server	0	Running	00:00:01	0 %	
DB4D Flush	-	DB4D Server	0	Running	00:00:01	0 %	
DB4D Index builder	-	DB4D Server	0	Running	00:00:01	0 %	
DB4D Server	-	DB4D Server	0	Running	00:00:01	0 %	
DB4D Sockets	-	DB4D Server	0	Running	00:00:01	0 %	
Garbage Handler	-	DB4D Server	0	Running	00:00:01	0 %	
HTTP Listener	-	Web Server	0	Running	00:00:01	0 %	
Internal Timer Process	-	Application server	2	Executing	00:00:01	0 %	
Logger	-	Logger process	0	Running	00:00:01	0 %	
Task managers	-	SQL Server	0	Running	00:00:01	0 %	
TCP connection listener	-	TCP Connection listener	0	Running	00:00:01	0 %	
TCP connection listener	-	SQL Server	0	Running	00:00:01	0 %	
User Interface	-	Application server	1	Waiting for event	00:00:02	2 %	
REST Handler: process1	marie-sophie	HTTP Server Worker	0	Running	00:00:08	90 %	
REST Handler: process2	marie-sophie	HTTP Server Worker	0	Running	00:00:08	89 %	

Verrouillage et transactions

Les fonctionnalités ORDA relatives au verrouillage d'entité et aux transactions sont gérées au niveau du process dans les datastore distants, tout comme en mode client/serveur ORDA :

- Si un process verrouille une entité à partir d'un datastore distant, l'entité est verrouillée pour tous les autres process, même lorsque ces process partagent la même session (voir [Verrouillage d'entités](#)). Si plusieurs entités pointant vers le même enregistrement ont été verrouillées dans un process, elles doivent toutes être déverrouillées dans le process pour supprimer le verrou. Si un verrou a été mis sur une entité, il est supprimé lorsqu'il n'existe plus de référence à cette entité en mémoire.
- Les transactions peuvent être lancées, validées ou annulées séparément sur chaque datastore distant à l'aide des méthodes `dataStore.startTransaction()`, `dataStore.cancelTransaction()`, et `dataStore.validateTransaction()`. Elles n'ont pas d'incidences sur les autres datastore.
- Les commandes classiques du langage 4D (`START TRANSACTION`, `VALIDATE TRANSACTION`, `CANCEL TRANSACTION`) s'appliquent uniquement au datastore principal (renvoyé par `ds`). Si une entité d'un datastore distant est verrouillée par une transaction dans un process, les autres process ne peuvent pas la mettre à jour, même si ces process partagent la même session.
- Les verrous sur les entités sont supprimés et les transactions sont annulées :
 - quand le processus est tué.
 - quand la session est fermée sur le serveur
 - lorsque la session est arrêtée à partir de la fenêtre d'administration du serveur.

Fermeture des sessions

Une session est automatiquement fermée par 4D lorsqu'il n'y a pas eu d'activité durant son timeout. Le timeout par défaut est de 60 mn mais cette valeur peut être paramétrée à l'aide du paramètre `connectionInfo` de la commande `Open datastore`.

Si une demande est envoyée au datastore distant après la fermeture de la session, elle est automatiquement recréée si possible (licence disponible, serveur non arrêté, etc.). A noter cependant que le contexte de la session des verrous et des transactions est perdu (voir ci-dessus).

Optimisation client/serveur

4D optimise automatiquement les requêtes ORDA qui utilisent des entity selections ou qui chargent des entités dans des configurations client/serveur (datastore accessible à distance à l'aide de `ds` ou de `Open datastore`). Ces optimisations accélèrent l'exécution de votre application 4D en réduisant drastiquement le volume d'informations transmises sur le réseau. Elles incluent :

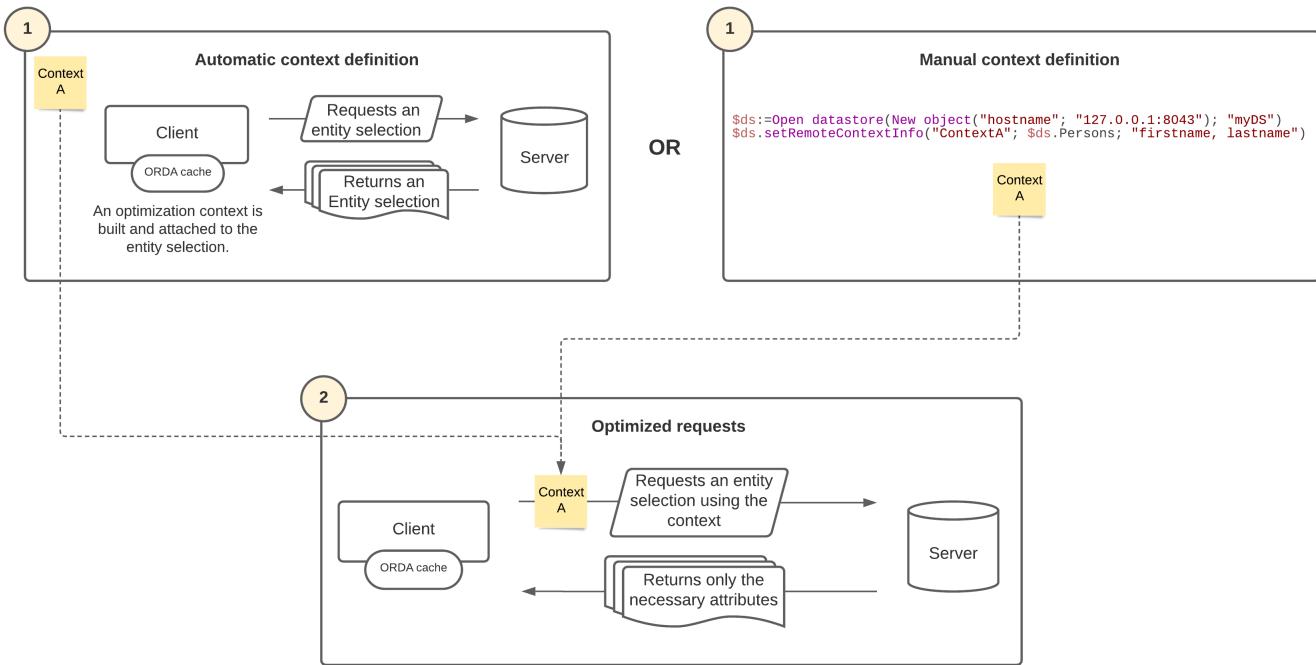
- le contexte d'optimisation
- le cache ORDA

Contexte

Le contexte d'optimisation est fondé sur ce qui suit :

- Lorsqu'un client demande une sélection d'entité au serveur, 4D "apprend" automatiquement attributs de la sélection d'entité sont réellement utilisés côté client lors de l'exécution du code, et génère un "contexte d'optimisation" correspondant. Ce contexte est relié à la sélection d'entité et stocke les attributs utilisés. Il sera mis à jour dynamiquement si d'autres attributs sont utilisés par la suite. Les méthodes et fonctions suivantes déclenchent la phase d'apprentissage :
 - `Créer une entity selection`
 - `dataClass.fromCollection()`
 - `dataClass.all()`
 - `dataClass.get()`
 - `dataClass.query()`
 - `entitySelection.query()`
- Les requêtes ultérieures envoyées au serveur sur la même sélection d'entité réutilisent automatiquement le contexte d'optimisation et lisent uniquement les attributs nécessaires depuis le serveur, ce qui accélère le traitement. Par exemple, dans une [list box de type entity selection](#), la phase d'apprentissage a lieu pendant l'affichage de la première ligne. L'affichage des lignes suivantes est optimisé. Les fonctions suivantes associent automatiquement le contexte d'optimisation de l'entity selection d'origine à l'entity selection retournée :
 - `entitySelection.and()`
 - `entitySelection.minus()`
 - `entitySelection.or()`
 - `entitySelection.orderBy()`
 - `entitySelection.slice()`
 - `entitySelection.drop()`
- Un contexte d'optimisation existant peut être passé en tant que propriété à une autre sélection d'entité de la même dataclass, ce qui permet d'éviter la phase d'apprentissage et d'accélérer l'application (voir [Utilisation de la propriété context](#) ci-dessous).
- Vous pouvez créer des contextes d'optimisation manuellement à l'aide de la fonction `dataStore.setRemoteContextInfo()` (voir [Préconfiguration des contextes](#)).

Client/Server Optimization Process



Exemple

Considérons le code suivant :

```
$sel:=$ds.Employee.query("firstname = ab@")
For each($e;$sel)
    $s:=$e.firstname+" "+$e.lastname+" works for "+$e.employer.name // $e.employer renvoie à la table Co
End for each
```

Grâce à l'optimisation, cette requête récupérera uniquement les données des attributs utilisés (prénom, nom, employeur, employeur.name) dans \$sel à partir de la deuxième itération de la boucle.

Réutiliser la propriété context

Vous pouvez tirer un meilleur parti de l'optimisation en utilisant la propriété context. Cette propriété référence un contexte d'optimisation "apris" pour une sélection d'entités. Elle peut être passée comme paramètre aux fonctions ORDA qui retournent de nouvelles sélections d'entités, afin que les sélections d'entités demandent directement au serveur les attributs utilisés, sans passer par la phase d'apprentissage.

Vous pouvez également créer des contextes à l'aide de la fonction `.setRemoteContextInfo()`.

All ORDA functions that handle entity selections support the context property (for example `dataClass.query()` or `dataClass.all()`). The same optimization context property can be passed to unlimited number of entity selections on the same dataclass. Il est toutefois important de garder à l'esprit qu'un contexte est automatiquement mis à jour lorsque de nouveaux attributs sont utilisés dans d'autres parties du code. Si le même contexte est réutilisé dans différents codes, il risque d'être surchargé et de perdre en efficacité.

A similar mechanism is implemented for entities that are loaded, so that only used attributes are requested (see the `dataClass.get()` function).

Exemple avec `dataClass.query()` :

```

var $sel1; $sel2; $sel3; $sel4; $querysettings; $querysettings2 : Object
var $data : Collection
$querysettings:=New object("context";"shortList")
$querysettings2:=New object("context";"longList")

$sel1:=ds.Employee.query("lastname = S@";$querysettings)
$data:=extractData($sel1) // In extractData method an optimization is triggered
// and associated to context "shortList"

$sel2:=ds.Employee.query("lastname = Sm@";$querysettings)
$data:=extractData($sel2) // In extractData method the optimization associated
// to context "shortList" is applied

$sel3:=ds.Employee.query("lastname = Smith";$querysettings2)
$data:=extractDetailedData($sel3) // In extractDetailedData method an optimization
// is triggered and associated to context "longList"

$sel4:=ds.Employee.query("lastname = Brown";$querysettings2)
$data:=extractDetailedData($sel4) // In extractDetailedData method the optimization
// associated to context "longList" is applied

```

Listbox basée sur une sélection d'entités

L'optimisation d'une sélection d'entités s'applique automatiquement aux listbox basées sur une sélection d'entités dans les configurations client/serveur, au moment d'afficher et de dérouler le contenu d'une listbox : seuls les attributs affichés dans la listbox sont demandés depuis le serveur.

Un contexte spécifique nommé "mode page" est également proposé lorsque l'entité courante de la sélection est chargée à l'aide de l'expression élément courant de la listbox (voir [List box de type collection ou entity selection](#)). Cette fonctionnalité vous permet de ne pas surcharger le contexte initial de la listbox dans ce cas précis, notamment si la "page" requiert des attributs supplémentaires. A noter que seule l'utilisation de l'expression Élément courant permettra de créer/utiliser le contexte de la page (l'accès via `entitySelection[index]` modifiera le contexte de la sélection d'entité).

Subsequent requests to server sent by entity browsing functions will also support this optimization. The following functions automatically associate the optimization context of the source entity to the returned entity:

- `entity.next()`
- `entity.first()`
- `entity.last()`
- `entity.previous()`

Par exemple, le code suivant charge l'entité sélectionnée et permet de naviguer dans la sélection d'entités. Les entités sont chargées dans un contexte séparé et le contexte initial de la listbox demeure inchangé :

```

$myEntity:=Form.currentElement //expression de l'élément courant
//... faire quelque chose
$myEntity:=$myEntity.next() //charge la prochaine entité à l'aide du même contexte

```

Preconfiguring contexts

An optimization context should be defined for every feature or algorithm of your application, in order to have the best performances. For example, a context can be used for queries on customers, another context for queries on products, etc.

If you want to deliver final applications with the highest level of optimization, you can preconfigure your contexts and thus save learning phases by following these steps:

1. Design your algorithms.
2. Run your application and let the automatic learning mechanism fill the optimization contexts.

3. Call the `dataStore.getRemoteContextInfo()` or `dataStore.getAllRemoteContexts()` function to collect contexts. You can use the `entitySelection.getRemoteContextAttributes()` and `entity.getRemoteContextAttributes()` functions to analyse how your algorithms use attributes.
4. In the final step, call the `dataStore.setRemoteContextInfo()` function to build contexts at application startup and [use them](#) in your algorithms.

ORDA cache

For optimization reasons, data requested from the server via ORDA is loaded in the ORDA remote cache (which is different from the 4D cache). The ORDA cache is organized by dataclass, and expires after 30 seconds.

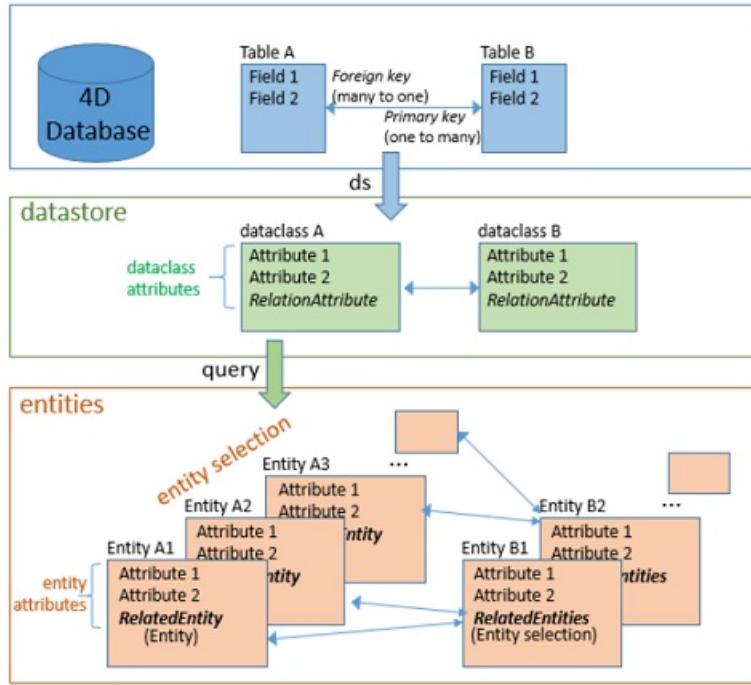
The data contained in the cache is considered as expired when the timeout is reached. Any access to expired data will send a request to the server. Expired data remains in the cache until space is needed.

By default, the ORDA cache is transparently handled by 4D. However, you can control its contents using the following ORDA class functions:

- `dataClass.setRemoteCacheSettings()`
- `dataClass.getRemoteCache()`
- `dataClass.clearRemoteCache()`

Glossaire

Aperçu des principaux concepts



Attribut

Un attribut est la plus petite cellule de stockage dans une base de données relationnelle (voir aussi [Attribut relationnel](#)). Ne confondez pas les attributs de la dataclass et les attributs d'entités :

- Dans un objet dataclass, chaque propriété est un attribut de dataclass qui correspond à un champ dans la table correspondante (même nom et même type).
- Dans un objet entity, les attributs d'entités sont des propriétés qui contiennent les valeurs pour les attributs du datastore correspondants.

Les *attributs* et les *propriétés* sont des concepts similaires. "Attribut" est utilisé pour désigner les propriétés de la dataclass qui stockent les données, tandis que "propriété" est plus générique et définit une donnée stockée dans un objet.

AttributePath

Un attributePath est le chemin d'un attribut à l'intérieur d'une dataclass ou d'une entité donnée. Voir aussi [propertyPath](#).

Class code

Code pour la (les) fonction(s) de classe utilisateurs.

Champ calculé

Un champ calculé ne stocke pas réellement d'informations. Il détermine plutôt sa valeur en fonction d'autres valeurs provenant de la même entité ou d'autres entités, champs ou fonctions. Lorsqu'un champ calculé est référencé, le

"calcul" sous-jacent est évalué pour déterminer la valeur. Les champs calculés peuvent même se voir attribuer des valeurs où le code défini par l'utilisateur détermine ce qu'il faut faire pendant l'attribution.

Data model class

Classe étendue disponible pour un objet modèle de données.

Data model object

Objets de base de données disponibles via le concept ORDA, c'est-à-dire datastore, dataclasses, entities (entités) et entity selections (sélections d'entités).

Data model function

Fonction d'une classe de modèle de données ORDA.

Dataclass

Une dataclass est un objet qui décrit les données. Les tables de la base de données fournies par le datastore sont gérées via des dataclasses. Chaque table de la base de données fournie par le datastore possède une dataclass correspondante portant le même nom. Chaque champ de la table est un attribut de la dataclass.

Une dataclass est reliée à un seul datastore.

DataClass class

Classe pour des objets dataclass spécifiques, dans laquelle vous pouvez ajouter des fonctions personnalisées.

Datastore

Un datastore est l'objet d'interface fourni par ORDA pour référencer une structure et accéder à ses données. La base de données principale, retournée par la commande `ds`, est disponible en tant que datastore (le datastore principal).

Un datastore fournit :

- une connexion à la base de données 4D
- un ensemble de dataclasses pour travailler avec la base de données

La base peut être une base locale 4D (le datastore principal), ou une base 4D Server exposée en ressource REST (un datastore distant).

Un datastore ne référence qu'une seule base de données. Il est toutefois possible d'ouvrir plusieurs datastores pour accéder à plusieurs bases.

DataStore class

Classe pour des objets datastore spécifiques, dans laquelle vous pouvez ajouter des fonctions personnalisées.

DataStoreImplementation

Nom interne de la classe générique DataStore dans le class store `4D`.

Copie profonde

Une copie profonde (deep copy) duplique un objet et toutes les références qu'il contient. Après une deep copy, une

collection copiée contient des éléments dupliqués et donc de nouvelles références de tous les éléments originaux. Voir également Copie superficielle.

ds

`ds` est la commande de langage 4D qui retourne une référence d'objet [datastore](#). Elle correspond au datastore disponible sur la base de données principale 4D.

Entity

Une entité est un objet qui correspond à un modèle de dataclass. Une entité contient les mêmes attributs que la dataclass.

Une entité peut être vue comme une instance de la dataclass, comme un enregistrement de la table correspondante à la dataclass dans son datastore associé. Cependant, une entité contient également des données connexes. Le but de l'entité est de gérer les données (créer, mettre à jour, supprimer).

Pour plus d'informations, voir le chapitre Entités.

Entity selection

Une sélection d'entités (entity selection) est un objet. Lorsqu'une requête est envoyée au datastore, une sélection d'entités est retournée. Une sélection d'entité est un ensemble de références à des entités liées à la même dataclass.

Une sélection d'entités contient :

- un ensemble de 0 à X références d'entités,
- une propriété `length` (toujours),
- les propriétés `queryPlan` et `queryPath` (si demandées lors de la requête).

Une sélection d'entités peut également être vide.

Generic class

Classe intégrée pour les objets ORDA tels que les entités ou les dataclasses. Les fonctions et propriétés des classes génériques sont automatiquement disponibles dans les classes utilisateur étendues, telles que `EmployeeEntity`.

Lazy loading

Comme les entités sont gérées comme des références, les données sont chargées uniquement lorsque cela est nécessaire, c'est-à-dire lorsqu'on y accède dans le code ou via des widgets d'interface. Ce principe d'optimisation est appelé lazy loading.

Datastore principal

L'objet Datastore correspondant à la base 4D ouverte (autonome ou client/serveur). Le datastore principal est retourné par la commande `ds`.

Méthode

Les objets ORDA tels que les "datastores", "dataclasses", "entity selections" et "entities" définissent les classes d'objets. Ils fournissent des méthodes spécifiques pour interagir directement avec eux. Ces méthodes sont aussi appelées des fonctions membres (member functions). Ces méthodes sont utilisées en étant appelées sur une instance de l'objet.

Par exemple, la méthode `query()` est une "member function" de dataclass. Si vous avez stocké un objet dataclass dans la variable `$myClass`, vous pouvez écrire :

```
$myClass.query("name = smith")
```

Type de données "Mixte"

Dans cette documentation, le type de données "Mixte" est utilisé pour désigner les différents types de valeurs qui peuvent être stockés dans les attributs d'une dataclass. Par exemple :

- number
- Texte
- null
- boolean
- date
- object
- collection
- image(*)

(*) le type *Image* n'est pas supporté par des méthodes statistiques telles que dans `entitySelection.max()`.

Verrouillage optimiste

En mode "verrouillage optimiste", les entités ne sont pas verrouillées explicitement avant d'être mises à jour. Chaque entité a un marqueur interne qui est automatiquement incrémenté chaque fois que l'entité est enregistrée sur le disque. Les méthodes `entity.save()` ou `entity.drop()` retourneront une erreur si le marqueur de l'entité chargée en mémoire et le marqueur de l'entité sur le disque ne correspondent pas, ou si l'entité a été supprimée. Le verrouillage optimiste est uniquement disponible dans l'implémentation ORDA. Voir aussi "verrouillage pessimiste".

Verrouillage pessimiste

Un "verrouillage pessimiste" signifie qu'une entité est verrouillée avant que l'on y accède, en utilisant la méthode `entity.lock()`. Les autres process ne peuvent ni mettre à jour ni supprimer l'entité tant qu'elle n'est pas déverrouillée. Le langage 4D classique n'autorise que les verrouillages pessimistes. Voir "Verrouillage optimiste".

Propriété

Voir [Attribut](#).

Les attributs et les propriétés sont des concepts similaires. "Attribut" est utilisé pour désigner les propriétés de la dataclass qui stockent les données, tandis que "propriété" est plus générique et définit une donnée stockée dans un objet.

PropertyPath

Un propertyPath est le chemin vers une propriété dans un objet donné. Si la propriété est imbriquée à plusieurs niveaux, chaque niveau est séparé par un point (".").

Regular class

Classe utilisateur non liée à un objet ORDA.

Related dataclass

Ce sont des dataclasses liées par des attributs de relation.

Attribut relationnel

Les attributs de relation sont utilisés pour conceptualiser les relations entre les dataclasses (N vers 1 et 1 vers N).

- Relation N vers 1 (la dataclassA fait référence à une occurrence de la dataclassB) : un attribut de relation est disponible dans dataclassA et fait référence à une instance de dataclassB.
- Relation 1 vers N (une occurrence de dataclassB fait référence à plusieurs occurrences de dataclassA) : un attribut de relation est disponible dans la dataclassB et fait référence à plusieurs instances de la dataclassA.

Une dataclass peut avoir des attributs de relation récursifs.

Dans une entité, la valeur d'un attribut de relation peut être une entité ou une sélection d'entité.

Related entities

Une entité associée peut être considérée comme l'instance d'un attribut de relation dans une dataclass.

Les sélections d'entités peuvent faire référence à des entités relatives selon les attributs de relation définis dans les dataclasses correspondantes.

Remote datastore

Une base de données 4D ouverte sur 4D ou 4D Server (disponible via HTTP) et exposée en tant que ressource REST. Cette base de données peut être référencée localement en tant que Datastore à partir d'autres postes de travail, où un locaID lui est attribué. Le datastore distant peut être utilisé à travers les concepts ORDA (datastore, dataclass, sélection d'entités, etc.). Cette utilisation est soumise à un système de licence.

Session

Lorsque l'application 4D se connecte à un datastore distant, une session est créée sur le 4D Server (HTTP). Un cookie de session est généré et associé à l'ID du datastore local.

Chaque fois qu'une nouvelle session est ouverte, une licence est utilisée. Chaque fois qu'une session est fermée, la licence est libérée.

Les sessions inactives sont automatiquement fermées après un délai. Le timeout par défaut est de 48 heures, il peut être défini par le développeur (il doit être ≥ 60 minutes).

Copie superficielle (Shallow copy)

Une copie superficielle (shallow copy) ne fait que dupliquer la structure des éléments et conserve les mêmes références internes. Après une copie superficielle, deux collections partageront les éléments individuels. Voir également Copie profonde.

Stamp

Utilisé dans la technologie du verrouillage "optimiste". Toutes les entités ont un compteur interne, le marqueur, qui est incrémenté chaque fois que l'entité est sauvegardée. En comparant automatiquement les marqueurs entre une entité sauvegardée et sa version stockée sur disque, 4D peut empêcher les modifications simultanées sur les mêmes entités.

Attribut de stockage

Un attribut de stockage (parfois appelé attribut scalaire) est le type d'attribut le plus basique dans une classe de datastore et correspond le plus directement à un champ dans une base de données relationnelle. Un attribut de stockage contient une seule valeur pour chaque entité de la classe.

Blob

La classe Blob vous permet de créer et de manipuler des [objets blob](#) (`4D.Blob`).

Sommaire

[`4D.Blob.new\(\) : 4D.Blob`](#)

[`4D.Blob.new\(blobScal : Blob \) : 4D.Blob`](#)

[`4D.Blob.new\(blobObj : 4D.Blob \) : 4D.Blob`](#)

crée un nouvel objet `4D.Blob` encapsulant, de manière facultative, une copie des données à partir d'un autre blob (blob scalaire ou `4D.Blob`)

[`.size : Real`](#)

retourne la taille d'un `4D.Blob`, exprimée en octets.

[`.slice\(\) : 4D.Blob`](#)

[`.slice\(start : Real \) : 4D.Blob`](#)

[`.slice\(start : Real; end : Real \) : 4D.Blob`](#)

crée et retourne un `4D.Blob` qui référence les données d'un sous-ensemble du blob sur lequel il est appelé. Le blob d'origine n'est pas modifié.

`4D.Blob.new()`

► Historique

`4D.Blob.new() : 4D.Blob`

`4D.Blob.new(blobScal : Blob) : 4D.Blob`

`4D.Blob.new(blobObj : 4D.Blob) : 4D.Blob`

Paramètres	Type		Description
blob	Blob ou <code>4D.Blob</code>	->	Blob pour copie
Résultat	<code>4D.Blob</code>	<-	Nouveau <code>4D.Blob</code>

Description

`4D.Blob.new` crée un nouvel objet `4D.Blob` encapsulant, de manière facultative, une copie des données à partir d'un autre blob (blob scalaire ou `4D.Blob`).

Si le paramètre `blob` est omis, la méthode retourne un `4D.Blob` vide.

`.size`

[`.size : Real`](#)

Description

La propriété `.size` retourne la taille d'un `4D.Blob`, exprimée en octets.

`.slice()`

► Historique

[`.slice\(\) : 4D.Blob`](#)

[`.slice\(start : Real \) : 4D.Blob`](#)

.slice(*start* : Real; *end* : Real) : 4D.Blob

Paramètres	Type		Description
start	Real	->	indice du premier octet à inclure dans le nouveau 4D.Blob .
end	Real	->	indice du premier octet qui ne sera pas inclus dans le nouveau 4D.Blob
Résultat	4D.Blob	<-	Nouveau 4D.Blob

Description

.slice() crée et retourne un 4D.Blob qui référence les données d'un sous-ensemble du blob sur lequel il est appelé. Le blob d'origine n'est pas modifié. Le paramètre start est un indice dans le blob, indiquant le premier octet à inclure dans le nouveau 4D.Blob . Si vous indiquez une valeur négative, 4D la traite comme un décalage de la fin du blob vers le début. Par exemple, -10 correspondrait à l'avant-dernier octet du blob. La valeur par défaut est 0. Si vous indiquez une valeur pour start supérieure à la taille du blob source, la taille du 4D.Blob retourné est 0, et il ne contient aucune donnée.

Le paramètre end est un indice dans le blob indiquant le premier octet qui ne sera pas inclus dans le nouveau 4D.Blob (c'est-à-dire que l'octet situé exactement à cet indice ne sera pas inclus). Si vous indiquez une valeur négative, 4D la traite comme un décalage de la fin du blob vers le début. Par exemple, -10 correspondrait à l'avant-dernier octet du blob. La valeur par défaut est la taille du blob.

Exemple

```
var $myBlob : 4D.Blob

// Stocker du texte dans un 4D.Blob
CONVERT FROM TEXT("Hello, World!"); "UTF-8"; $myBlob)
$is4DBlob:=OB Instance of($myBlob; 4D.Blob); //True

$myString:=Convert to text($myBlob; "UTF-8")
// $myString contient "Hello, World!"

// Créer un nouveau 4D.Blob à partir de $myBlob
$myNewBlob:=$myBlob.slice(0; 5)

$myString:=Convert to text($myNewBlob; "UTF-8")
// $myString contient "Hello"
```

Class

Lorsqu'une classe utilisateur est [définie](#) dans le projet, elle est chargée dans l'environnement de langage 4D. Une classe est un objet lui-même, de la classe "Class", qui a des propriétés et une fonction.

Sommaire

.name : Text

contient le nom de l'objet `4D.Class`

.new(*param* : any { ;...*paramN* }) : 4D.Class

crée et retourne un objet `cs.className` qui est une nouvelle instance de la classe sur laquelle il est appelé

.superclass : 4D.Class

retourne la classe parente de la classe

.name

► Historique

.name : Text

Description

La propriété `.name` contient le nom de l'objet `4D.Class`. Les noms de classe sont sensibles à la casse.

Cette propriété est en lecture seule.

.new()

► Historique

.new(*param* : any { ;...*paramN* }) : 4D.Class

Paramètres	Type		Description
param	any	->	Paramètre(s) à passer à la fonction constructeur
Résultat	4D.Class	<-	Nouvel objet de la classe

Description

La fonction `.new()` crée et retourne un objet `cs.className` qui est une nouvelle instance de la classe sur laquelle il est appelé. Cette fonction est automatiquement disponible sur toutes les classes à partir du class store `cs`.

Vous pouvez passer un ou plusieurs paramètre(s) optionnel(s) `param`, qui seront passés à la fonction [class constructor](#) (le cas échéant) dans la définition de la classe `className`. A l'intérieur de la fonction constructeur, `This` est lié au nouvel objet en cours de construction.

Si `.new()` si appelé sur une classe qui n'existe pas, une erreur est retournée.

Exemples

Pour créer une nouvelle instance de la classe Person :

```
var $person : cs.Person  
$person:=cs.Person.new() //créer la nouvelle instance  
//$person contient les fonctions de la classe
```

Pour créer une nouvelle instance de la classe Person avec des paramètres :

```
//Class: Person.4dm  
Class constructor($firstname : Text; $lastname : Text; $age : Integer)  
    This.firstName:=$firstname  
    This.lastName:=$lastname  
    This.age:=$age
```

```
//Dans une méthode  
var $person : cs.Person  
$person:=cs.Person.new("John";"Doe";40)  
//$person.firstName = "John"  
//$person.lastName = "Doe"  
//$person.age = 40
```

.superclass

► Historique

.superclass : 4D.Class

Description

La propriété `.superclass` retourne la classe parente de la classe. Une superclasse peut être un objet `4D.Class`, ou un objet `cs.className`. Si la classe n'a pas de classe parente, la propriété renvoie `null`.

Une superclasse de classe utilisateur est déclarée dans une classe à l'aide du mot-clé `Class extend<superclass>`.

Cette propriété est en lecture seule.

Exemples

```
$sup:=4D.File.superclass //Document  
$sup:=4D.Document.superclass //Object  
$sup:=4D.Object.superclass //null  
  
// Si vous avez créé une classe MyFile  
// avec `Class extends File`  
$sup:=cs.MyFile.superclass //File
```

Voir également : [Super](#)

Collection

La classe Collection gère les variables de type [Collection](#).

Une collection est initialisée avec :

New collection {{ ...value : any }} : Collection
crée une nouvelle collection vide ou préremplie

New shared collection {{ ...value : any }} : Collection
crée une nouvelle collection partagée vide ou pré-remplie et retourne sa référence

Exemple

```
var $colVar : Collection //création d'une variable 4D de type collection  
$colVar:=New collection ///initialisation de la collection et assignation à la variable 4D
```

Sommaire

.average({propertyPath : Text }) : Real
retourne la moyenne arithmétique des valeurs définies dans la collection

.clear() : Collection
supprime tous les éléments de la collection et retourne une collection vide

.combine(col2 : Collection {; index : Integer }) : Collection
insère les éléments de col2 elements la fin ou à la position index de la collection et retourne la collection modifiée

.concat(value : any { ;...valueN }) : Collection
retourne une nouvelle collection avec le contenu du paramètre value ajouté à la fin de la collection d'origine

.copy() : Collection
.copy(option : Integer) : Collection
.copy(option : Integer ; groupWithCol : Collection) : Collection
.copy(option : Integer ; groupWithObj : Object) : Collection
retourne une copie profonde (deep copy) de la collection

.count({ propertyPath : Text }) : Real
retourne le nombre d'éléments non null dans la collection

.countValues(value : any {; propertyPath : Text }) : Real
retourne le nombre d'occurrences de value dans la collection

.distinct({ option : Integer}) : Collection
.distinct(propertyPath : Text {; option : Integer }) : Collection

.equal(collection2 : Collection {; option : Integer }) : Boolean
compare la collection avec collection2

.every(methodName : Text { ;...param : any }) : Boolean
.every(startFrom : Integer ; methodName : Text { ;...param : any }) : Boolean

retourne vrai si tous les éléments de la collection ont été évalués à vrai par le test implémenté dans la méthode *methodName* passée en paramètre

Paramètre(s) à passer à *methodName*

crée et retourne une nouvelle collection contenant les valeurs de *propertyPath* extraites depuis la collection d'objets d'origine

`.fill(value : any) : Collection`

`.fill(value : any ; startFrom : Integer { ; end : Integer }) : Collection`

remplit les éléments de la collection avec *value*, optionnellement depuis l'élément *startFrom* et jusqu'à l'élément *end* (non inclus), et retourne la collection résultante

`.filter(methodName : Text { ; ...param : any }) : Collection`

retourne une nouvelle collection contenant tous les éléments de la collection d'origine pour lesquels le résultat de la méthode *methodName* est true

`.find(methodName : Text { ; ...param : any }) : any`

`.find(startFrom : Integer ; methodName : Text { ; ...param : any }) : any`

retourne la première valeur dans la collection pour laquelle *methodName* retourne true

`.findIndex(methodName : Text { ; ...param : any }) : Integer`

`.findIndex(startFrom : Integer ; methodName : Text { ; ...param : any }) : Integer`

retourne le numéro, dans la collection, du premier élément pour lequel *methodName* retourne true

`.indexOf(toSearch : expression { ; startFrom : Integer }) : Integer`

recherche l'expression *toSearch* parmi les éléments de la collection et retourne le numéro d'élément de la première occurrence trouvée, ou -1 si aucune occurrence n'a été trouvée

`.indices(queryString : Text { ; ...value : any }) : Collection`

retourne les positions, dans la collection d'origine, des éléments répondant au(x) critère(s) de recherche de *queryString*

`.insert(index : Integer ; element : any) : Collection`

insère *element* dans la collection à la position spécifiée par *index* et retourne la collection modifiée

`.join(delimiter : Text { ; option : Integer }) : Text`

`.lastIndexOf(toSearch : expression { ; startFrom : Integer }) : Integer`

recherche l'expression *toSearch* parmi les éléments de la collection et retourne le numéro d'élément de la dernière occurrence trouvée

[`.length : Integer`

`] (#length)`

retourne le nombre d'éléments contenus dans la collection.

`.map(methodName : Text { ; ...param : any }) : Collection`

crée une nouvelle collection basée sur le résultat de l'exécution de la méthode *methodName* sur chaque élément de la collection d'origine

`.max({ propertyPath : Text }) : any`

`.min({ propertyPath : Text }) : any`

`.orderBy() : Collection`

`.orderBy(pathStrings : Text) : Collection`

`.orderBy(pathObjects : Collection) : Collection`
`.orderBy(ascOrDesc : Integer) : Collection`

retourne une nouvelle collection contenant tous les éléments de la collection d'origine triés selon les critères définis

`.orderByMethod(methodName : Text { ; ...extraParam : expression }) : Collection`

retourne une nouvelle collection contenant tous les éléments de la collection d'origine triés selon les critères définis par *methodName*

`.pop() : any`

supprime le dernier élément de la collection et le retourne comme résultat de la fonction

`.push(element : any { ;...elementN }) : Collection`

`.query(queryString : Text ; ...value : any) : Collection`

`.query(queryString : Text ; querySettings : Object) : Collection`

retourne tous les éléments d'une collection d'objets qui correspondent aux critères de recherche

`.reduce(methodName : Text) : any`

`.reduce(methodName : Text ; initialValue : any { ; ...param : expression }) : any`

applique la méthode callback *methodName* à un accumulateur et à chaque élément de la collection (de gauche à droite) pour le réduire à une valeur unique

`.remove(index : Integer { ; howMany : Integer }) : Collection`

supprime un ou plusieurs élément(s) de la position d'*index* spécifiée dans la collection et retourne la collection modifiée

`.resize(size : Integer { ; defaultValue : any }) : Collection`

définit la longueur de la collection sur la nouvelle taille spécifiée et retourne la collection redimensionnée

`.reverse() : Collection`

retourne une copie complète de la collection avec tous ses éléments dans l'ordre inverse

`.shift() : any`

`.slice(startFrom : Integer { ; end : Integer }) : Collection`

retourne une partie d'une collection dans une nouvelle collection

`.some(methodName : Text { ; ...param : any }) : Boolean`

`.some(startFrom : Integer ; methodName : Text { ; ...param : any }) : Boolean`

retourne true si au moins un élément de la collection a réussi un test

`.sort(methodName : Text { ; ...extraParam : any }) : Collection`

trie les éléments de la collection d'origine

`.sum({ propertyPath : Text }) : Real`

`.unshift(value : any { ;...valueN : any }) : Collection`

insère la ou les *valeur(s)* données au début de la collection

New collection

New collection `{(...value : any)}` : Collection

Paramètres	Type		Description
value	Number, Text, Date, Time, Boolean, Object, Collection, Picture, Pointer	->	Valeur(s) de collection
Résultat	Collection	<-	Nouvelle collection

Description

La commande `New collection` crée une nouvelle collection vide ou préremplie et retourne sa référence.

Si vous ne passez aucun paramètre, `New collection` crée une collection vide et retourne sa référence.

Vous devez affecter la référence retournée à une variable 4D de type Collection.

Gardez à l'esprit que les instructions `var : Collection` ou `C_COLLECTION` déclarent une variable de type `Collection` mais ne créent aucune collection.

En option, vous pouvez préremplir la nouvelle collection en utilisant une ou plusieurs valeur(s) (`value(s)`) en tant que paramètre(s).

Sinon, vous pouvez ajouter ou modifier des éléments ultérieurement par affectation. Par exemple :

```
myCol[10]:="My new element"
```

Si le nouvel indice de l'élément est au-delà du dernier élément existant de la collection, la collection est automatiquement redimensionnée et tous les nouveaux éléments intermédiaires sont attribués une valeur null.

Vous pouvez passer n'importe quel nombre de valeurs de n'importe quel type pris en charge (number, text, date, picture, pointer, object, collection...). Contrairement aux tableaux, les collections peuvent mélanger des données de différents types.

Vous devez prêter attention aux problèmes de conversion suivants :

- Si vous passez un pointeur, il est conservé "tel quel"; il est évalué à l'aide de la commande `JSON Stringify`
- Les dates sont stockées sous la forme de date « aaaa-mm-jj » ou des chaînes au format « AAAA-MM-JJTHH:ss.SSSZ: mm », selon la configuration actuelle « dates à l'intérieur des objets » de la base de données. Lors de la conversion de dates 4D en texte avant de les stocker dans la collection, par défaut le programme prend en compte le fuseau horaire local. Vous pouvez modifier ce comportement à l'aide du sélecteur `Dates inside objects` de la commande `SET DATABASE PARAMETER`.
- Si vous passez une heure, elle est stockée sous la forme d'un nombre de millisecondes (Réel).

Exemple 1

Vous souhaitez créer une nouvelle collection vide et l'assigner à une variable collection 4D :

```
var $myCol : Collection
$myCol:=New collection
//$/myCol=[]
```

Exemple 2

Vous souhaitez créer une collection pré-remplie :

```
var $filledColl : Collection
$filledColl:=New collection(33;"mike";"november";->myPtr;Current date)
//$/filledColl=[33,"mike","november","->myPtr","2017-03-28T22:00:00.000Z"]
```

Exemple 3

Vous souhaitez créer une nouvelle collection puis ajouter un élément :

```
var $coll : Collection
$coll:=New collection("a";"b";"c")
//$coll=["a","b","c"]
$coll[9]:="z" //ajouter un 10e élément avec la valeur "z"
$vcolSize:=$coll.length //10
//$coll=["a","b","c",null,null,null,null,null,"z"]
```

New shared collection

► Historique

New shared collection {(...value : any)} : Collection

Paramètres	Type		Description
value	Number, Text, Date, Time, Boolean, Shared object, Shared collection	->	Valeur(s) de la collection partagée
Résultat	Collection	<-	New shared collection

Description

La commande `New shared collection` crée une nouvelle collection partagée vide ou pré-remplie et retourne sa référence .

L'ajout et la modification d'éléments dans une collection partagée doivent être encadrés par une structure `Use...End`, sinon une erreur est générée. La lecture d'un élément hors `Use...End` est toutefois possible.

Pour plus d'informations sur les collections partagées, reportez-vous à la page [Objets et collections partagés](#).

Si vous ne passez aucun paramètre, `New shared collection` crée une collection vide et retourne sa référence.

Vous devez affecter la référence renournée à une variable 4D de type Collection.

Gardez à l'esprit que les instructions `var : Collection` ou `C_COLLECTION` déclarent une variable de type `Collection` mais ne créent aucune collection.

En option, vous pouvez préremplir la nouvelle collection en utilisant une ou plusieurs valeur(s) (`value(s)`) en tant que paramètre(s). Sinon, vous pouvez ajouter ou modifier des éléments ultérieurement via l'assignation en notation objet (cf. exemple).

Si le nouvel indice de l'élément est au-delà du dernier élément existant de la collection, la collection est automatiquement redimensionnée et tous les nouveaux éléments intermédiaires prennent la valeur null.

Vous pouvez passer tout nombre de valeurs de n'importe quel type pris en charge :

- nombre (réel, entier...). Les valeurs numériques sont toujours stockées sous forme de réels.
- text
- boolean
- date
- heure (stockée en nombre de milliseconds - réel)
- null
- shared object(*)
- shared collection(*)

Contrairement aux collections standard (non partagées), les collections partagées ne prennent pas en charge les

images, les pointeurs et les objets ou collections non partagés.

(*) Lorsqu'un objet partagé ou une collection partagée est ajouté(e) comme élément à une collection partagée, il/elle hérite de son *locking identifier*. Pour plus d'informations sur ce point, reportez-vous au manuel Concepts du langage 4D.

Exemple

```
$mySharedCol:=New shared collection("alpha";"omega")
Use($mySharedCol)
  $mySharedCol[1]:="beta"
End use
```

.average()

► Historique

.average({*propertyPath* : Text }) : Real

Paramètres	Type		Description
<i>propertyPath</i>	Text	->	Chemin de propriété d'objet à utiliser pour évaluer les valeurs
Résultat	Real, Undefined	<-	Moyenne arithmétique des valeurs de la collection

Description

La fonction `.average()` retourne la moyenne arithmétique des valeurs définies dans la collection.

Seuls les éléments ayant une valeur numérique sont pris en compte pour le calcul (les autres types d'éléments sont ignorés).

Si la collection contient des objets, passez le paramètre *propertyPath* si vous souhaitez désigner la propriété dont vous voulez connaître la moyenne.

`.average()` retourne `undefined` si :

- la collection est vide,
- la collection ne contient pas d'éléments numériques,
- *propertyPath* n'est pas trouvé dans la collection.

Exemple 1

```
var $col : Collection
$col:=New collection(10;20;"Monday";True;6)
$vAvg:=$col.average() //12
```

Exemple 2

```
var $col : Collection
$col:=New collection
$col.push(New object("name";"Smith";"salary";10000))
$col.push(New object("name";"Wesson";"salary";50000))
$col.push(New object("name";"Gross";"salary";10500))
$vAvg:=$col.average("salary") //23500
```

.clear()

► Historique

.clear() : Collection

Paramètres	Type		Description
Résultat	Collection	<-	Collection d'origine dont tous les éléments ont été supprimés

Description

La fonction `.clear()` supprime tous les éléments de la collection et retourne une collection vide.

Cette fonction modifie la collection d'origine.

Exemple

```
var $col : Collection
$col:=New collection(1;2;5)
$col.clear()
$vSize:=$col.length //vSize=0
```

.combine()

► Historique

.combine(*col2* : Collection {; *index* : Integer }) : Collection

Paramètres	Type		Description
<i>col2</i>	Collection	->	Collection à combiner
<i>index</i>	Integer	->	Emplacement où insérer les éléments à combiner (défaut=length+1)
Résultat	Collection	<-	Collection d'origine incluant les éléments combinés

Description

La fonction `.combine()` insère les éléments de *col2* éléments la fin ou à la position *index* de la collection et retourne la collection modifiée. A la différence de la fonction `.insert()`, `.combine()` ajoute chaque valeur de *col2* dans la collection d'origine, et non en tant qu'élément unique de collection.

Cette fonction modifie la collection d'origine.

Par défaut, les éléments de *col2* sont ajoutés à la fin de la collection d'origine. Vous pouvez passer dans *index* le numéro de l'élément après lequel vous souhaitez que les éléments de *col2* soient insérés dans la collection.

Attention : A noter que la numérotation des éléments de collection débute à 0.

- Si *index* > la longueur de la collection, l'*index* de départ réel sera fixé à la longueur de la collection.
- Si *index* < 0, il est recalculé comme *index*:=*index*+*length* (il est considéré comme le décalage par rapport à la fin de la collection).
- Si la valeur calculée est négative, *index* prend la valeur 0.

Exemple

```

var $c; $fruits : Collection
$c:=New collection(1;2;3;4;5;6)
$fruits:=New collection("Orange";"Banana";"Apple";"Grape")
$c.combine($fruits;3) // [1,2,3,"Orange","Banana","Apple","Grape",4,5,6]

```

.concat()

► Historique

.concat(*value* : any { ;...*valueN* }) : Collection

Paramètres	Type		Description
<i>value</i>	Number, Text, Object, Collection, Date, Time, Boolean, Picture	->	Valeur(s) à concaténer. Si <i>value</i> est une collection, tous les éléments de la collection sont ajoutés à la collection d'origine
Résultat	Collection	<-	Nouvelle collection contenant les valeurs d'origine et les valeurs ajoutées

Description

La fonction `.concat()` retourne une nouvelle collection avec le contenu du paramètre *value* ajouté à la fin de la collection d'origine.

Cette fonction ne modifie pas la collection d'origine.

Si *value* est une collection, tous ses éléments sont ajoutés comme nouveaux éléments à la fin de la collection d'origine.
Si *value* n'est pas une collection, son contenu est ajouté comme nouvel élément.

Exemple

```

var $c : Collection
$c:=New collection(1;2;3;4;5)
$fruits:=New collection("Orange";"Banana";"Apple";"Grape")
$fruits.push(New object("Intruder";"Tomato"))
$c2:=$c.concat($fruits) // [1,2,3,4,5,"Orange","Banana","Apple","Grape",{"Intruder":"Tomato"}]
$c2:=$c.concat(6;7;8) // [1,2,3,4,5,6,7,8]

```

.copy()

► Historique

.copy() : Collection

.copy(*option* : Integer) : Collection

.copy(*option* : Integer ; *groupWithCol* : Collection) : Collection

.copy(*option* : Integer ; *groupWithObj* : Object) : Collection

Paramètres	Type		Description
<i>option</i>	Integer	->	ck resolve pointers : résoudre les pointeurs avant la copie, ck shared : retourner une collection partagée
<i>groupWithCol</i>	Collection	->	Collection partagée à grouper avec la collection résultante
<i>groupWithObj</i>	Object	->	Objet partagé à grouper avec la collection résultante
Résultat	Collection	<-	Copie de la collection d'origine (deep copy)

Description

La fonction `.copy()` retourne une copie profonde (deep copy) de la collection. *Deep copy* signifie que les objets ou les collections présents dans la collection d'origine sont dupliqués et ne partagent pas leur référence avec la collection qui est retournée.

Cette fonction ne modifie pas la collection d'origine.

S'il est passé, le paramètre *option* peut contenir l'une des constantes suivantes (ou les deux) :

option	Description
<code>ck resolve pointers</code>	Si la collection d'origine contient des valeurs de type pointeur, par défaut la copie contient également les pointeurs. Toutefois, vous pouvez résoudre les pointeurs au moment de la copie, en passant <code>ck resolve pointers</code> . Dans ce cas, chaque pointeur contenu dans la collection est évalué lors de la copie et sa valeur déréférencée est utilisée.
<code>ck shared</code>	Par défaut, <code>copy()</code> retourne une collection standard (non partagée), même si la fonction s'applique à une collection partagée. Passez la constante <code>ck shared</code> pour créer une collection partagée. Dans ce cas, vous pouvez utiliser le paramètre <code>groupWith</code> pour associer la collection partagée avec un(e) autre collection/objet (voir ci-dessous).

Les paramètres `groupWithCol` ou `groupWithObj` vous permettent de désigner une collection ou un objet auquel/à laquelle la collection résultante sera associée.

Exemple 1

Nous souhaitons copier la collection standard (non partagée) `$lastnames` dans l'objet partagé `$sharedObject`. Pour ce faire, nous devons créer une copie partagée de la collection (`$sharedLastnames`).

```
var $sharedObject : Object
var $lastnames;$sharedLastnames : Collection
var $text : Text

$sharedObject:=New shared object

$text:=Document to text(Get 4D folder(Current resources folder)+"lastnames.txt")
$lastnames:=JSON Parse($text) // $lastnames est une collection standard

$sharedLastnames:=$lastnames.copy(ck shared) // $sharedLastnames est une collection partagée

// Nous pouvons désormais placer $sharedLastnames dans $sharedObject
Use($sharedObject)
    $sharedObject.lastnames:=$sharedLastnames
End use
```

Exemple 2

Nous souhaitons combiner `$sharedColl1` et `$sharedColl2`. Etant donné qu'ils appartiennent à différents groupes partagés, une combinaison directe pourrait générer une erreur. Nous devons effectuer une copie partagée de `$sharedColl1` et désigner `$sharedColl2` comme étant un groupe partagé pour la copie.

```
var $sharedColl1;$sharedColl2;$copyColl : Collection

$sharedColl1:=New shared collection(New shared object("lastname";"Smith"))
$sharedColl2:=New shared collection(New shared object("lastname";"Brown"))

// $copyColl appartient au nouveau groupe partagé comme $sharedColl2
$copyColl:=$sharedColl1.copy(ck shared:$sharedColl2)
Use($sharedColl2)
    $sharedColl2.combine($copyColl)
End use
```

Exemple 3

Nous avons une collection standard (`$lastnames`) et nous souhaitons la placer dans le Storage de l'application. Pour ce faire, nous devons préalablement créer une copie partagée (`$sharedLastnames`).

```
var $lastnames;$sharedLastnames : Collection
var $text : Text

$text:=Document to text(Get 4D folder(�Current resources folder)+"lastnames.txt")
$lastnames:=JSON Parse($text) // $lastnames est une collection standard

$sharedLastnames:=$lastnames.copy(ck shared) // copie partagée

Use(Storage)
  Storage.lastnames:=$sharedLastnames
End use
```

Exemple 4

Cet exemple illustre l'utilisation de l'option `ck resolve pointers` :

```
var $col : Collection
var $p : Pointer
$p:==>$what

$col:=New collection
$col.push(New object("alpha";"Hello";"num";1))
$col.push(New object("beta";"You";"what";$p))

$col2:=$col.copy()
$col2[1].beta:="World!"
ALERT($col[0].alpha+" "+$col2[1].beta) // affiche "Hello World!"

$what:="You!"
$col3:=$col2.copy(ck resolve pointers)
ALERT($col3[0].alpha+" "+$col3[1].what) // affiche "Hello You!"
```

.count()

► Historique

`.count({ propertyPath : Text }) : Real`

Paramètres	Type		Description
propertyPath	Text	->	Chemin de propriété d'objet à utiliser pour évaluer les valeurs
Résultat	Réel	<-	Nombre d'éléments dans la collection

Description

La fonction `.count()` retourne le nombre d'éléments non null dans la collection.

Si la collection contient des objets, vous pouvez passer le paramètre `propertyPath`. Dans ce cas, seuls les éléments qui contiennent le `propertyPath` sont comptabilisés.

Exemple

```

var $col : Collection
var $count1;$count2 : Real
$col:=New collection(20;30;Null;40)
$col.push(New object("name";"Smith";"salary";10000))
$col.push(New object("name";"Wesson";"salary";50000))
$col.push(New object("name";"Gross";"salary";10500))
$col.push(New object("lastName";"Henry";"salary";12000))
$count1:=$col.count() // $count1=7
$count2:=$col.count("name") // $count2=3

```

.countValues()

► Historique

.countValues(*value* : any {; *propertyPath* : Text }) : Real

Paramètres	Type		Description
<i>value</i>	Text, Number, Boolean, Date, Object, Collection	->	Valeur à compter
<i>propertyPath</i>	Text	->	Chemin de propriété d'objet à utiliser pour évaluer les valeurs
Résultat	Réel	<-	Nombre d'occurrences de la valeur

Description

La fonction `.countValues()` retourne le nombre d'occurrences de *value* dans la collection.

Vous pouvez passer dans *value* :

- une valeur scalaire (texte, numérique, booléen, date),
- une référence d'objet ou de collection.

Pour qu'un élément soit comptabilisé, le type de *value* doit être égal à celui de l'élément ; la fonction utilise l'opérateur d'égalité.

Le paramètre optionnel *propertyPath* vous permet de compter des valeurs à l'intérieur d'une collection d'objets : passez dans *propertyPath* le chemin de la propriété dont vous souhaitez comptabiliser le nombre de valeurs.

Cette fonction ne modifie pas la collection d'origine.

Exemple 1

```

var $col : Collection
var $vCount : Integer
$col:=New collection(1;2;5;5;5;3;6;4)
$vCount:=$col.countValues(5) // $vCount=3

```

Exemple 2

```

var $col : Collection
var $vCount : Integer
$col:=New collection
$col.push(New object("name";"Smith";"age";5))
$col.push(New object("name";"Wesson";"age";2))
$col.push(New object("name";"Jones";"age";3))
$col.push(New object("name";"Henry";"age";4))
$col.push(New object("name";"Gross";"age";5))
$vCount:=$col.countValues(5;"age") //vCount=2

```

Exemple 3

```

var $numbers; $letters : Collection
var $vCount : Integer

$letters:=New collection("a";"b";"c")
$numbers:=New collection(1;2;$letters;3;4;5)

$vCount:=$numbers.countValues($letters) //vCount=1

```

.distinct()

► Historique

.distinct(*{option : Integer}*) : Collection
 .distinct(*propertyPath : Text {; option : Integer}*) : Collection

Paramètres	Type		Description
option	Integer	->	ck diacritical : évaluation diacritique ("A" # "a" par exemple)
propertyPath	Text	->	Chemin de l'attribut dont vous souhaitez obtenir les valeurs distinctes
Résultat	Collection	<-	Nouvelle collection contenant uniquement les valeurs distinctes

Description

La fonction `.distinct()` renvoie une collection contenant uniquement les valeurs distinctes (différentes) dans la collection d'origine.

Cette fonction ne modifie pas la collection d'origine.

La collection retournée est automatiquement triée. Les valeurs Null ne sont pas renvoyées.

Par défaut, une évaluation non diacritique est effectuée. Si vous souhaitez que l'évaluation soit sensible à la casse ou pour différencier des caractères accentués et non-accentués, passez la constante `ck diacritical` dans le paramètre *option*.

Si la collection contient des objets, vous pouvez passer le paramètre *propertyPath* afin d'indiquer la propriété d'objet dont vous souhaitez obtenir les valeurs distinctes.

Exemple

```

var $c; $c2 : Collection
$c:=New collection
$c.push("a";"b";"c";"A";"B";"c";"b";"b")
$c.push(New object("size";1))
$c.push(New object("size";3))
$c.push(New object("size";1))
$c2:=$c.distinct() // $c2=[{"a":1,"b":1,"c":1,"size":1}, {"a":2,"b":2,"c":2,"size":3}, {"a":3,"b":3,"c":3,"size":1}]
$c2:=$c.distinct(ck diacritical) // $c2=[{"a":1,"A":1,"B":2,"c":1,"size":1}, {"a":2,"b":2,"c":2,"size":3}, {"a":3,"b":3,"c":3,"size":1}]
$c2:=$c.distinct("size") // $c2=[1,3]

```

.equal()

► Historique

.equal(*collection2* : Collection {; *option* : Integer }) : Boolean

Paramètres	Type		Description
collection2	Collection	->	Collection à comparer
option	Integer	->	ck diacritical : évaluation diacritique ("A" # "a" par exemple)
Résultat	Booléen	<-	Vrai si les collections sont identiques, sinon faux

Description

La fonction `.equal()` compare la collection avec *collection2* et retourne vrai si elles sont identiques (deep comparison).

Par défaut, une évaluation non diacritique est effectuée. Si vous souhaitez que l'évaluation soit sensible à la casse ou pour différencier des caractères accentués et non-accentués, passez la constante `ck diacritical` dans le paramètre *option*.

Les éléments contenant valeurs Null ne sont pas équivalents aux éléments Undefined.

Exemple

```

var $c; $c2 : Collection
var $b : Boolean

$c:=New collection(New object("a";1;"b";"orange");2;3)
$c2:=New collection(New object("a";1;"b";"orange");2;3;4)
$b:=$c.equal($c2) // false

$c:=New collection(New object("1";"a";"b";"orange");2;3)
$c2:=New collection(New object("a";1;"b";"orange");2;3)
$b:=$c.equal($c2) // false

$c:=New collection(New object("a";1;"b";"orange");2;3)
$c2:=New collection(New object("a";1;"b";"0Range");2;3)
$b:=$c.equal($c2) // true

$c:=New collection(New object("a";1;"b";"orange");2;3)
$c2:=New collection(New object("a";1;"b";"0Range");2;3)
$b:=$c.equal($c2;ck diacritical) //false

```

.every()

► Historique

.every(*methodName* : Text { ;...*param* : any }) : Boolean

.every(*startFrom* : Integer ; *methodName* : Text { ;...*param* : any }) : Boolean

Paramètres	Type		Description
<i>startFrom</i>	Integer	->	Elément à partir duquel débuter l'évaluation
<i>methodName</i>	Text	->	Nom de la méthode à appeler pour l'évaluation
<i>param</i>	Mixed	->	Paramètre(s) à passer à nomMéthode
Résultat	Booléen	<-	Vrai si tous les éléments sont évalués à vrai

Description

La fonction `.every()` retourne vrai si tous les éléments de la collection ont été évalués à vrai par le test implémenté dans la méthode *methodName* passée en paramètre.

Dans *methodName*, passez le nom de la méthode à utiliser pour évaluer les éléments de la collection, ainsi que son ou ses paramètre(s) dans *param* (optionnel). *methodName* peut effectuer tout test, avec ou sans paramètres. La méthode reçoit un paramètre de type `Object` dans `$1` et doit passer true dans `$1.result` pour chaque élément ayant satisfait aux conditions du test.

methodName reçoit les paramètres suivants :

- dans `$1.value` : valeur de l'élément à évaluer
- in `$2`: param
- in `$N...`: paramN...

methodName doit fixer le(s) paramètre(s) suivant(s) :

- `$1.result` (booléen) : true si l'évaluation de la valeur de l'élément est réussie, sinon `false`.
- `$1.stop` (booléen, optionnel) : true pour stopper le rétroappel de méthode. La valeur renournée est la dernière calculée.

Dans tous les cas, au premier élément retournant `false` dans `$1.result`, la fonction `.every()` cesse d'appeler *methodName* et retourne `false`.

Par défaut, `.every()` évalue l'ensemble de la collection. Optionnellement, vous pouvez passer dans *startFrom* le numéro de l'élément auquel démarrer l'évaluation.

- Si *startFrom* \geq la longueur de la collection, `false` est retourné, ce qui signifie que la collection n'est pas testée.
- Si *startFrom* < 0 , la fin de la collection est considérée comme point de départ du calcul de la position(`startFrom:=startFrom+length`).
- Si *startFrom* = 0, l'ensemble de la collection est évalué (défaut).

Exemple 1

```
var $c : Collection
var $b : Boolean
$c:=New collection
$c.push(5;3;1;4;6;2)
$b:=$c.every("NumberGreaterThan0") //retourne true
$c.push(-1)
$b:=$c.every("NumberGreaterThan0") //retourne false
```

Avec la méthode `NumberGreaterThan0` suivante :

```
$1.result:=$1.value>0
```

Exemple 2

Cet exemple vérifie que tous les éléments de la collection sont de type réel :

```

var $c : Collection
var $b : Boolean
$c:=New collection
$c.push(5;3;1;4;6;2)
$b:=$c.every("TypeLookUp";Is real) //$/b=true
$c:=$c.push(New object("name";"Cleveland";"zc";35049))
$c:=$c.push(New object("name";"Blountsville";"zc";35031))
$b:=$c.every("TypeLookUp";Is real) //$/b=false

```

Avec la méthode *TypeLookUp* suivante :

```

#DECLARE ($toEval : Object ; $param : Integer) //$/1; $2
If(Value type($toEval.value)=$param)
    $toEval.result:=True
End if

```

.extract()

► Historique

Paramètre(s) à passer à *methodName*

Paramètres	Type		Description
propertyPath	Text	->	Chemin de propriété d'objet dont les valeurs doivent être extraites dans la nouvelle collection
targetpath	Text	->	Chemin ou nom de propriété cible
option	Integer	->	ck keep null : inclure les propriétés null dans la collection retournée (ignorées par défaut). Paramètre ignoré si <i>targetPath</i> est passé.
Résultat	Collection	<-	Nouvelle collection contenant les valeurs extraites

Description

La fonction `.extract()` crée et retourne une nouvelle collection contenant les valeurs de *propertyPath* extraites depuis la collection d'objets d'origine.

Cette fonction ne modifie pas la collection d'origine.

Le contenu de la collection retournée dépend du paramètre *targetPath* :

- Si le paramètre *targetPath* est omis, `.extract()` remplit la nouvelle collection avec les valeurs de *propertyPath* de la collection d'origine.
Par défaut, les éléments pour lesquels *propertyPath* est null ou undefined sont ignorés dans la collection résultante. Vous pouvez passer la constante `ck keep null` dans le paramètre *option* pour intégrer ces valeurs comme des éléments null dans la collection retournée.
- Si un ou plusieurs paramètre(s) *targetPath* sont passés, `.extract()` remplit la nouvelle collection avec les propriétés *propertyPath* et chaque élément de la nouvelle collection est un objet contenant les propriétés *targetPath* dont les valeurs sont celles des propriétés *propertyPath* correspondantes. Les valeurs null sont conservées (le paramètre *option* est ignoré avec cette syntaxe).

Exemple 1

```

var $c : Collection
$c:=New collection
$c.push(New object("name";"Cleveland"))
$c.push(New object("zip";5321))
$c.push(New object("name";"Blountsville"))
$c.push(42)
$c2:=$c.extract("name") // $c2=[Cleveland,Blountsville]
$c2:=$c.extract("name";ck keep null) // $c2=[Cleveland,null,Blountsville,null]

```

Exemple 2

```

var $c : Collection
$c:=New collection
$c.push(New object("zc";35060))
$c.push(New object("name";Null;"zc";35049))
$c.push(New object("name";"Cleveland";"zc";35049))
$c.push(New object("name";"Blountsville";"zc";35031))
$c.push(New object("name";"Adger";"zc";35006))
$c.push(New object("name";"Clanton";"zc";35046))
$c.push(New object("name";"Clanton";"zc";35045))
$c2:=$c.extract("name";"City") // $c2=[{City:null},{City:Cleveland},{City:Blountsville},{City:Adger},{City:Clanton},{City:Clanton}]
$c2:=$c.extract("name";"City";"zc";"Zip") // $c2=[{Zip:35060},{City:null,Zip:35049},{City:Cleveland,Zip:35031},{City:Adger,Zip:35006},{City:Clanton,Zip:35046},{City:Clanton,Zip:35045}]

```

.fill()

► Historique

`.fill(value : any) : Collection`
`.fill(value : any ; startFrom : Integer { ; end : Integer }) : Collection`

Paramètres	Type		Description
value	Number, Text, Object, Collection, Date, Boolean	->	Valeur de remplissage
startFrom	Integer	->	Numéro de l'élément de départ (inclus)
end	Integer	->	Position de fin (non incluse)
Résultat	collection	<-	Collection d'origine avec valeurs de remplissage

Description

La fonction `.fill()` remplit les éléments de la collection avec `value`, optionnellement depuis l'élément `startFrom` et jusqu'à l'élément `end` (non inclus), et retourne la collection résultante.

Cette fonction modifie la collection d'origine.

- Si `startFrom` est omis, `value` est appliquée à tous les éléments de la collection (`startFrom=0`).
- Si `startFrom` est passé et `end` est omis, `value` est appliquée à tous les éléments de la collection à partir de `startFrom` jusqu'au dernier élément (`end=length`).
- Si les deux paramètres `startFrom` et `end` sont passés, `value` est appliquée aux éléments de la collection à partie de `startFrom` jusqu'à l'élément `end`.

En cas d'incohérence, les règles suivantes sont appliquées :

- Si `startFrom < 0`, le paramètre est recalculé comme `startFrom:=startFrom+length` (la fin de la collection est considérée comme point de départ du calcul de la position). Si la valeur recalculée est négative, `startFrom` prend la

valeur 0.

- Si `end < 0`, le paramètre est recalculé comme `end:=end+length`.
- Si `end < startFrom` (valeurs passées ou recalculées), la fonction ne fait rien.

Exemple

```
var $c : Collection
$c:=New collection(1;2;3;"Lemon";Null;"";4;5)
$c.fill("2") // $c:=[2,2,2,2,2,2,2]
$c.fill("Hello";5) // $c=[2,2,2,2,Hello,Hello,Hello]
$c.fill(0;1;5) // $c=[2,0,0,0,0>Hello,Hello,Hello]
$c.fill("world";1;-5) // -5+8=3 -> $c=[2,"world","world",0,0>Hello,Hello,Hello]
```

.filter()

► Historique

`.filter(methodName : Text { ; ...param : any }) : Collection`

Paramètres	Type		Description
methodName	Text	->	Nom de la méthode à appeler pour filtrer la collection
param	Mixed	->	Paramètre(s) à passer à <code>methodName</code>
Résultat	Collection	<-	Nouvelle collection contenant les éléments filtrés (shallow copy)

Description

La fonction `.filter()` retourne une nouvelle collection contenant tous les éléments de la collection d'origine pour lesquels le résultat de la méthode `methodName` est true. Cette méthode retourne une *shallow copy* (copie superficielle), ce qui signifie que les objets ou les collections présents dans les deux collections partagent la même référence. Si la collection d'origine est une collection partagée, la collection retournée est également une collection partagée.

Cette fonction ne modifie pas la collection d'origine.

Dans `methodName`, passez le nom de la méthode à utiliser pour évaluer les éléments de la collection, ainsi que son ou ses paramètre(s) dans `param` (optionnel). `methodName` peut effectuer tout test, avec ou sans paramètres. Dans `methodName`, passez le nom de la méthode à utiliser pour évaluer les éléments de la collection, ainsi que son ou ses paramètre(s) dans `param` (optionnel).

`methodName` reçoit les paramètres suivants :

- dans `$1.value` : valeur de l'élément à évaluer
- dans `$2` : `param`
- dans `$N...` : `param2...paramN`

`methodName` doit fixer le(s) paramètre(s) suivant(s) :

- `$1.result` (booléen) : true si l'élément satisfait à la condition de filtrage et doit être conservé.
- `$1.stop` (booléen, optionnel) : true pour stopper le rétroappel de méthode. La valeur retournée est la dernière calculée.

Exemple 1

Vous voulez obtenir la collection des éléments de type texte dont la longueur est inférieure à 6 :

```

var $col;$colNew : Collection
$col:=New collection("hello";"world";"red horse";66;"tim";"san jose";"miami")
$colNew:=$col.filter("LengthLessThan";6)
// $colNew=["hello","world","tim","miami"]

```

Le code de la méthode *LengthLessThan* est le suivant :

```

C_OBJECT($1)
C_LONGINT($2)
If(Value type($1.value)=Is text)
    $1.result:=(Length($1.value))<$2
End if

```

Exemple 2

Vous voulez filtrer les éléments de la collection en fonction de leur type :

```

var $c;$c2;$c3 : Collection
$c:=New collection(5;3;1;4;6;2)
$c.push(New object("name";"Cleveland";"zc";35049))
$c.push(New object("name";"Blountsville";"zc";35031))
$c2:=$c.filter("TypeLookUp";Is real) // $c2=[5,3,1,4,6,2]
$c3:=$c.filter("TypeLookUp";Is object)
// $c3=[{name:Cleveland,zc:35049},{name:Blountsville,zc:35031}]

```

Le code de *TypeLookUp* est :

```

C_OBJECT($1)
C_LONGINT($2)
If(0B Get type($1;"value")=$2)

    $1.result:=True
End if

```

.find()

► Historique

.find(*methodName* : Text { ; ...*param* : any }) : any
 .find(*startFrom* : Integer ; *methodName* : Text { ; ...*param* : any }) : any

Paramètres	Type		Description
<i>startFrom</i>	Integer	->	Elément à partir duquel débuter la recherche
<i>methodName</i>	Text	->	Nom de la méthode à appeler pour la recherche
<i>param</i>	any	->	Paramètre(s) à passer à <i>methodName</i>
Résultat	any	<-	Première valeur trouvée (Undefined si non trouvée)

Description

La fonction `.find()` retourne la première valeur dans la collection pour laquelle *methodName* retourne true.

Cette fonction ne modifie pas la collection d'origine.

Dans *methodName*, passez le nom de la méthode à utiliser pour évaluer les éléments de la collection, ainsi que son ou

ses paramètre(s) dans *param* (optionnel). *methodName* peut effectuer tout test, avec ou sans paramètres. Dans *methodName*, passez le nom de la méthode à utiliser pour évaluer les éléments de la collection, ainsi que son ou ses paramètre(s) dans *param* (optionnel).

methodName reçoit les paramètres suivants :

- dans *\$1.value* : valeur de l'élément à évaluer
- dans *\$2* : *param*
- dans *\$N...* : *param2...paramN*

methodName doit fixer le(s) paramètre(s) suivant(s) :

- *\$1.result* (booléen) : true si l'élément satisfait à la condition de recherche et doit être conservé.
- *\$1.stop* (booléen, optionnel) : true pour stopper le rétroappel de méthode. La valeur retournée est la dernière calculée.

Par défaut, `.find()` effectue une recherche dans la totalité de la collection. Optionnellement, vous pouvez passer dans *startFrom* le numéro de l'élément auquel démarrer la recherche.

- Si *startFrom* >= la longueur de la collection, -1 est retourné, ce qui signifie que la recherche n'est pas effectuée.
- Si *startFrom* < 0, la fin de la collection est considérée comme point de départ du calcul de la position (*startFrom:=startFrom+length*). Note : Même si *startFrom* est négatif, la recherche est effectuée de la gauche vers la droite.
- Si *startFrom* = 0, l'ensemble de la collection est évalué (défaut).

Exemple 1

Vous souhaitez obtenir le premier élément dont la taille est inférieure à 5 caractères :

```
var $col : Collection
$col:=New collection("hello";"world";4;"red horse";"tim";"san jose")
$value:=$col.find("LengthLessThan";5) //$/value="tim"
```

Le code de la méthode *LengthLessThan* est le suivant :

```
var $1 : Object
var $2 : Integer
If(Value type($1.value)=Is text)
    $1.result:=(Length($1.value))<$2
End if
```

Exemple 2

Vous souhaitez trouver un nom de ville dans une collection :

```
var $c : Collection
var $c2 : Object
$c:=New collection
$c.push(New object("name";"Cleveland";"zc";35049))
$c.push(New object("name";"Blountsville";"zc";35031))
$c.push(New object("name";"Adger";"zc";35006))
$c.push(New object("name";"Clanton";"zc";35046))
$c.push(New object("name";"Clanton";"zc";35045))
$c2:=$c.find("FindCity";"Clanton") //$/c2={name:Clanton,zc:35046}
```

Le code de la méthode *FindCity* est :

```

var $1 : Object
var $2 : Text
$1.result:=$1.value.name=$2 //name est un nom de propriété d'objets dans la collection

```

.findIndex()

► Historique

`.findIndex(methodName : Text { ; ...param : any }) : Integer`
`.findIndex(startFrom : Integer ; methodName : Text { ; ...param : any }) : Integer`

Paramètres	Type		Description
startFrom	Integer	->	Elément à partir duquel débuter la recherche
methodName	Text	->	Nom de la méthode à appeler pour la recherche
param	any	->	Paramètre(s) à passer à <i>methodName</i>
Résultat	Integer	<-	Numéro du premier élément trouvé (-1 si non trouvé)

Description

La fonction `.findIndex()` retourne le numéro, dans la collection, du premier élément pour lequel *methodName* retourne true.

Cette fonction ne modifie pas la collection d'origine.

Dans *methodName*, passez le nom de la méthode à utiliser pour évaluer les éléments de la collection, ainsi que son ou ses paramètre(s) dans *param* (optionnel). Dans *methodName*, passez le nom de la méthode à utiliser pour évaluer les éléments de la collection, ainsi que son ou ses paramètre(s) dans *param* (optionnel). *methodName* peut effectuer tout test, avec ou sans paramètres.

methodName reçoit les paramètres suivants :

- dans `$1.value` : valeur de l'élément à évaluer
- dans `$2` : *param*
- dans `$N...` : *param2...paramN*

methodName doit fixer le(s) paramètre(s) suivant(s) :

- `$1.result` (booléen) : true si l'élément satisfait à la condition de recherche et doit être conservé.
- `$1.stop` (booléen, optionnel) : true pour stopper le rétroappel de méthode. La valeur retournée est la dernière calculée.

Par défaut, `.findIndex()` effectue une recherche dans la totalité de la collection. Optionnellement, vous pouvez passer dans *startFrom* le numéro de l'élément auquel démarrer la recherche.

- Si *startFrom* >= la longueur de la collection, -1 est retourné, ce qui signifie que la recherche n'est pas effectuée.
- Si *startFrom* < 0, la fin de la collection est considérée comme point de départ du calcul de la position (`startFrom:=startFrom+length`). Note : Même si *startFrom* est négatif, la recherche est effectuée de la gauche vers la droite.
- Si *startFrom* = 0, l'ensemble de la collection est évalué (défaut).

Exemple

Vous souhaitez trouver la position du premier nom de ville dans la collection :

```

var $c : Collection
var $val2;$val3 : Integer
$c:=New collection
$c.push(New object("name";"Cleveland";"zc";35049))
$c.push(New object("name";"Blountsville";"zc";35031))
$c.push(New object("name";"Adger";"zc";35006))
$c.push(New object("name";"Clanton";"zc";35046))
$c.push(New object("name";"Clanton";"zc";35045))
$val2:=$c.findIndex("FindCity";"Clanton") // $val2=3
$val3:=$c.findIndex($val2+1;"FindCity";"Clanton") // $val3=4

```

Le code de la méthode *FindCity* est :

```

var $1 : Object
var $2 : Text
$1.result:=$1.value.name=$2

```

.indexOf()

► Historique

`.indexOf(toSearch : expression { ; startFrom : Integer }) : Integer`

Paramètres	Type		Description
toSearch	expression	->	Expression à rechercher dans la collection
startFrom	Integer	->	Elément à partir duquel débuter la recherche
Résultat	Integer	<-	Numéro de la première occurrence de toSearch dans la collection, -1 si non trouvée

Description

La fonction `.indexOf()` recherche l'expression *toSearch* parmi les éléments de la collection et retourne le numéro d'élément de la première occurrence trouvée, ou -1 si aucune occurrence n'a été trouvée.

Cette fonction ne modifie pas la collection d'origine.

Dans *toSearch*, passez l'expression à rechercher dans la collection. Vous pouvez passer :

- une valeur scalaire (texte, numérique, booléen, date),
- la valeur null,
- une référence d'objet ou de collection.

toSearch doit correspondre exactement à l'élément recherché (les mêmes règles que pour l'opérateur d'égalité sont appliquées).

Optionnellement, vous pouvez passer le numéro de l'élément auquel démarrer la recherche dans *startFrom*.

- Si *startFrom* >= la longueur de la collection, -1 est retourné, ce qui signifie que la recherche n'est pas effectuée.
- Si *startFrom* < 0, la fin de la collection est considérée comme point de départ du calcul de la position (*startFrom:=startFrom+length*). Note : Même si *startFrom* est négatif, la recherche est effectuée de la gauche vers la droite.
- Si *startFrom* = 0, l'ensemble de la collection est évalué (défaut).

Exemple

```

var $col : Collection
var $i : Integer
$col:=New collection(1;2;"Henry";5;3;"Albert";6;4;"Alan";5)
$i:=$col.indexOf(3) // $i=4
$i:=$col.indexOf(5;5) // $i=9
$i:=$col.indexOf("al@") // $i=5
$i:=$col.indexOf("Hello") // $i=-1

```

.indices()

► Historique

.indices(*queryString* : Text { ; ...*value* : any }) : Collection

Paramètres	Type		Description
<i>queryString</i>	Text	->	Critère(s) de recherche
<i>value</i>	any	->	Valeur(s) à comparer lors de l'utilisation de paramètre(s) dans la chaîne
Résultat	Collection	<-	Numéro(s) d'élément(s) de la collection répondant au(x) critère(s) de recherche

Description

La fonction `.indices()` fonctionne exactement comme `.query()` mais retourne les positions, dans la collection d'origine, des éléments répondant au(x) critère(s) de recherche de *queryString*, et non les éléments eux-mêmes. Les positions sont renvoyées dans un ordre croissant.

Cette fonction ne modifie pas la collection d'origine.

Le paramètre *queryString* doit respecter la syntaxe suivante :

```
propertyPath comparator value {logicalOperator propertyPath comparator value}
```

Pour une description détaillée de la construction de recherches à l'aide des paramètres *queryString* et *value* veuillez vous reporter à la description de la fonction `dataclass.query()`.

Exemple

```

var $c; $icol : Collection
$c:=New collection
$c.push(New object("name";"Cleveland";"zc";35049))
$c.push(New object("name";"Blountsville";"zc";35031))

$c.push(New object("name";"Adger";"zc";35006))
$c.push(New object("name";"Clanton";"zc";35046))
$c.push(New object("name";"Clanton";"zc";35045))
$icol:=$c.indices("name = :1";"Cleveland") // $icol=[0]
$icol:=$c.indices("zc > 35040") // $icol=[0,3,4]

```

.insert()

► Historique

.insert(*index* : Integer ; *element* : any) : Collection

Paramètres	Type		Description
index	Integer	->	Où insérer l'élément
element	any	->	Elément à insérer dans la collection
Résultat	Collection	<-	Collection d'origine incluant l'élément inséré

Description

La fonction `.insert()` insère *element* dans la collection à la position spécifiée par *index* et retourne la collection modifiée.

Cette fonction modifie la collection d'origine.

Dans *index*, passez le numéro de l'élément après lequel vous souhaitez que le paramètre *element* soit inséré.

Attention : A noter que la numérotation des éléments de collection débute à 0.

- Si *index* > la longueur de la collection, l'*index* de départ réel sera fixé à la longueur de la collection.
- Si *index* < 0, il est recalculé comme *index*:=*index*+*length* (la fin de la collection est considérée comme point de départ du calcul de la position).
- Si la valeur recalculée est négative, *index* prend la valeur 0.

Vous pouvez passer tout type d'élément accepté par les collections, y compris une autre collection.

Exemple

```
var $col : Collection
$col:=New collection("a";"b";"c";"d") // $col=["a","b","c","d"]
$col.insert(2;"X") // $col=["a","b","X","c","d"]
$col.insert(-2;"Y") // $col=["a","b","X","Y","c","d"]
$col.insert(-10;"Hi") // $col=[["Hi","a","b","X","Y","c","d"]]
```

.join()

► Historique

`join(delimiter : Text { ; option : Integer }) : Text`

Paramètres	Type		Description
delimiter	Text	->	Séparateur à utiliser entre les éléments
option	Integer	->	<code>ck ignore null or empty</code> : ignorer les chaînes null ou vides dans le résultat
Résultat	Text	<-	Chaîne contenant tous les éléments de la collection, séparés par <i>delimiter</i>

Description

La fonction `.unshift()` insère la ou les *valeur(s)* données au début de la collection et retourne la collection modifiée.

Cette fonction ne modifie pas la collection d'origine.

Par défaut, les éléments null ou vides de la collection sont inclus dans la chaîne résultante. Passez la constante `ck ignore null or empty` dans le paramètre *option* si vous souhaitez les exclure de la chaîne résultante.

Exemple

```

var $c : Collection
var $t1;$t2 : Text
$c:=New collection(1;2;3;"Paris";Null;"";4;5)
$t1:=$c.join("|") //1|2|3|Paris|null||4|5
$t2:=$c.join("|";ck ignore null or empty) //1|2|3|Paris|4|5

```

.lastIndexOf()

► Historique

.lastIndexOf(*toSearch* : expression { ; *startFrom* : Integer }) : Integer

Paramètres	Type		Description
<i>toSearch</i>	expression	->	Elément à chercher dans la collection
<i>startFrom</i>	Integer	->	Elément à partir duquel débuter la recherche
Résultat	Integer	<-	Numéro de la dernière occurrence de <i>toSearch</i> dans la collection, -1 si non trouvé

Description

La fonction `.lastIndexOf()` recherche l'expression *toSearch* parmi les éléments de la collection et retourne le numéro d'élément de la dernière occurrence trouvée, ou -1 si aucune occurrence n'a été trouvée.

Cette fonction ne modifie pas la collection d'origine.

Dans *toSearch*, passez l'expression à rechercher dans la collection. Vous pouvez passer :

- une valeur scalaire (texte, numérique, booléen, date),
- la valeur null,
- une référence d'objet ou de collection.

Optionnellement, vous pouvez passer le numéro de l'élément auquel démarrer la recherche inversée dans *startFrom*.

Optionnellement, vous pouvez passer le numéro de l'élément auquel démarrer la recherche inversée dans *startFrom*.

- Si *startFrom* >= taille de la collection (*coll.length-1*), l'ensemble de la collection est évalué (défaut).
- Si *startFrom* < 0, le paramètre est recalculé comme *startFrom:=startFrom+length* (la fin de la collection est considérée comme point de départ du calcul de la position). Si la position calculée est négative, -1 est retourné (la collection n'est pas évaluée). Note : Même si *startFrom* est négatif, la recherche est effectuée de la droite vers la gauche.
- Si *startFrom* = 0, -1 est retourné, ce qui signifie que la recherche n'est pas effectuée.

Exemple

```

var $col : Collection
var $pos1;$pos2;$pos3;$pos4;$pos5 : Integer
$col:=Split string("a,b,c,d,e,f,g,h,i,j,e,k,e;","") // $col.length=13
$pos1:=$col.lastIndexOf("e") //retourne 12
$pos2:=$col.lastIndexOf("e";6) //retourne 4
$pos3:=$col.lastIndexOf("e";15) //retourne 12
$pos4:=$col.lastIndexOf("e";-2) //retourne 10
$pos5:=$col.lastIndexOf("x") //retourne -1

```

.length

► Historique

.length : Integer

Description

La propriété `.length` retourne le nombre d'éléments contenus dans la collection. La propriété `.length` est initialisée à la création de la collection. Elle est automatiquement mise à jour en cas d'ajout ou de suppression d'éléments. Cette propriété est en lecture seulement (vous ne pouvez pas l'utiliser pour modifier la taille de la collection).

Exemple

```
var $col : Collection // $col.length est initialisée à 0
$col:=New collection("one";"two";"three") // $col.length est mise à jour et vaut 3
$col[4]:="five" // $col.length vaut 5
$vSize:=$col.remove(0;3).length // $vSize=2
```

.map()

► Historique

`.map(methodName : Text { ; ...param : any }) : Collection`

Paramètres	Type		Description
methodName	Text	->	Nom de la méthode à appeler pour transformer les éléments de la collection
param	any	->	Paramètre(s) à passer à la méthode
Résultat	Collection	<-	Collection de valeurs transformées

Description

La fonction `.map()` crée une nouvelle collection basée sur le résultat de l'exécution de la méthode `methodName` sur chaque élément de la collection d'origine. Optionnellement, vous pouvez passer des paramètres à `methodName` via le paramètre `param`. `.map()` retourne toujours une collection de taille égale à celle de la collection d'origine.

Cette fonction ne modifie pas la collection d'origine.

Dans `methodName`, passez le nom de la méthode à utiliser pour évaluer les éléments de la collection, ainsi que son ou ses paramètre(s) dans `param` (optionnel). Dans `methodName`, passez le nom de la méthode à utiliser pour évaluer les éléments de la collection, ainsi que son ou ses paramètre(s) dans `param` (optionnel).

`methodName` reçoit les paramètres suivants :

- dans `$1.value` (tout type) : valeur de l'élément à évaluer
- dans `$2` (tout type) : `param`
- dans `$N...` (tout type) : `paramN...`

`methodName` doit fixer le(s) paramètre(s) suivant(s) :

- `$1.result` (tout type) : nouvelle valeur transformée à ajouter à la collection résultante
- `$1.stop` (booléen, optionnel) : true pour stopper le rétroappel de méthode. La valeur retournée est la dernière calculée.

Exemple

```
var $c; $c2 : Collection
$c:=New collection(1;4;9;10;20)
$c2:=$c.map("Percentage";$c.sum())
// $c2=[2.27, 9.09, 20.45, 22.73, 45.45]
```

Avec la méthode `NumberGreaterThan0` suivante :

```
var $1 : Object
var $2 : Real
$1.result:=Round(( $1.value/$2)*100;2)
```

.max()

► Historique

`.max({ propertyPath : Text }) : any`

Paramètres	Type		Description
propertyPath	Text	->	Chemin de propriété d'objet à utiliser pour évaluer les valeurs
Résultat	Boolean, Text, Number, Collection, Object, Date	<-	Valeur maximum de la collection

Description

Si la collection contient différents types de valeurs, la fonction `.max()` retournera la plus grande valeur du premier type d'élément dans l'ordre de la liste de types (cf. description de `.sort()`).

Cette fonction ne modifie pas la collection d'origine.

Si la collection contient des objets, passez le paramètre `propertyPath` pour indiquer la propriété d'objet dont vous souhaitez obtenir la valeur maximale.

Si la collection est vide, `.max()` retourne `Undefined`.

La fonction `.some()` retourne true si au moins un élément de la collection a réussi un test implémenté dans la méthode `methodName` fournie.

Exemple

```
var $col : Collection
$col:=New collection(200;150;55)
$col.push(New object("name";"Smith";"salary";10000))
$col.push(New object("name";"Wesson";"salary";50000))
$col.push(New object("name";"Alabama";"salary";10500))
$max:=$col.max() //{name:Alabama,salary:10500}
$maxSal:=$col.max("salary") //50000
$maxName:=$col.max("name") //Wesson"
```

.min()

► Historique

`.min({ propertyPath : Text }) : any`

Paramètres	Type		Description
propertyPath	Text	->	Chemin de propriété d'objet à utiliser pour évaluer les valeurs
Résultat	Boolean, Text, Number, Collection, Object, Date	<-	Valeur minimum de la collection

Description

Si la collection contient différents types de valeurs, la fonction `.mix()` retournera la plus petite valeur du premier type d'élément dans l'ordre de la liste de types (cf. description de `.sort()`).

Cette fonction ne modifie pas la collection d'origine.

Si la collection contient des objets, passez le paramètre `propertyPath` pour indiquer la propriété d'objet dont vous souhaitez obtenir la valeur minimum.

Si la collection est vide, `.min()` retourne `Undefined`.

La fonction `.some()` retourne true si au moins un élément de la collection a réussi un test implémenté dans la méthode `methodName` fournie.

Exemple

```
var $col : Collection
$col:=New collection(200;150;55)
$col.push(New object("name";"Smith";"salary";10000))
$col.push(New object("name";"Wesson";"salary";50000))
$col.push(New object("name";"Alabama";"salary";10500))
$min:=$col.min() //55
$minSal:=$col.min("salary") //10000
$minName:=$col.min("name") //"Alabama"
```

.orderBy()

► Historique

`.orderBy()` : Collection

`.orderBy(pathStrings : Text)` : Collection

`.orderBy(pathObjects : Collection)` : Collection

`.orderBy(ascOrDesc : Integer)` : Collection

Paramètres	Type		Description
pathStrings	Text	->	Chemin(s) de propriété(s) à utiliser pour trier la collection
pathObjects	Collection	->	Collection d'objets critère
ascOrDesc	Integer	->	<code>ck ascending</code> ou <code>ck descending</code> (valeurs scalaires)
Résultat	Collection	<-	Copiée triée de la collection (shallow copy)

Description

La fonction `.orderBy()` retourne une nouvelle collection contenant tous les éléments de la collection d'origine triés selon les critères définis.

Cette fonction retourne une **shallow copy (copie superficielle), ce qui signifie que les objets ou les collections présents dans les deux collections partagent la même référence. Si la collection d'origine est une collection partagée, la collection renvoyée est également une collection partagée.

Cette fonction ne modifie pas la collection d'origine.

Si vous ne passez aucun paramètre, la fonction trie les valeurs scalaires de la collection par ordre croissant (les autres types d'éléments tels que les objets ou les collections sont renvoyés sans être triés). Vous pouvez modifier ce tri automatique par défaut en passant la constante `ck ascending` ou `ck descending` dans le paramètre `ascOrDesc` (voir ci-dessous).

Vous pouvez également passer des critères afin de configurer le tri des éléments de la collection. Trois syntaxes sont

prises en charge pour ce paramètre :

- *pathStrings* : Texte (formule). Syntaxe : `propertyPath1 {desc ou asc}, propertyPath2 {desc ou asc},...` (défaut : asc). *pathStrings* contient une formule constituée de 1 à N chemin(s) de propriété(s) et (optionnellement) ordres de tri, séparés par des virgules. L'ordre dans lequel les propriétés sont passées détermine la priorité de tri des éléments de la collection. Par défaut, les propriétés sont triées pas ordre croissant. Vous pouvez définir l'ordre de tri de chaque propriété dans la formule de critère, séparée du chemin de propriété par un simple espace : passez "asc" pour trier par ordre croissant ou "desc" pour un ordre décroissant.
- *pathObjects* : Collection. Vous pouvez ajouter autant d'objets dans la collection *pathObjects* que nécessaire. Par défaut, les propriétés sont triées par ordre croissant ("descending" est faux). Chaque élément de la collection contient un objet structuré de la manière suivante :

```
{  
    "propertyPath": string,  
    "descending": boolean  
}
```

- *ascOrDesc* : Entier. Passez une des constantes suivantes du thème Objets et collections :

Constante	Type	Valeur	Commentaire
ck ascending	Longint	0	Les éléments sont triés par ordre croissant (défaut)
ck descending	Longint	1	Les éléments sont triés par ordre décroissant

Cette syntaxe trie uniquement les valeurs scalaires de la collection (les autres types d'éléments comme les objets ou les collections sont retournés non triés).

Si la collection contient des éléments de différents types, ils sont d'abord groupés par type et triés par la suite. Si *attributePath* est un attribut d'objet qui contient des valeurs de types différents, elles sont groupées par type et triées ensuite.

1. null
2. booléens
3. chaînes
4. nombres
5. objets
6. collections
7. dates

Exemple 1

Tri d'une collection d'objets basé sur une formule de texte avec noms de propriétés :

```
var $c; $c2; $3 : Collection  
$c:=New collection  
For($vCounter;1;10)  
    $c.push(Random)  
End for  
$c2:=$c.orderBy(ck ascending)  
$c3:=$c.orderBy(ck descending)
```

Exemple 2

Tri d'une collection d'objets sur des propriétés :

```

var $c; $c2 : Collection
$c:=New collection
For($vCounter;1;10)
    $c.push(New object("id";$vCounter;"value";Random))
End for
$c2:=$c.orderBy("value desc")
$c2:=$c.orderBy("value desc, id")
$c2:=$c.orderBy("value desc, id asc")

```

Tri d'une collection d'objets via une collection d'objets critères :

```

var $c; $c2 : Collection
$c:=New collection
$c.push(New object("name";"Cleveland";"phones";New object("p1";"01";"p2";"02")))
$c.push(New object("name";"Blountsville";"phones";New object("p1";"00";"p2";"03")))

$c2:=$c.orderBy("phones.p1 asc")

```

Exemple 3

Tri avec un chemin de propriété :

```

var $crit; $c; $c2 : Collection
$crit:=New collection
$c:=New collection
For($vCounter;1;10)
    $c.push(New object("id";$vCounter;"value";Random))
End for
$crit.push(New object("propertyPath";"value";"descending";True))
$crit.push(New object("propertyPath";"id";"descending";False))
$c2:=$c.orderBy($crit)

```

Tri avec un chemin de propriété :

```

var $crit; $c; $c2 : Collection
$c:=New collection
$c.push(New object("name";"Cleveland";"phones";New object("p1";"01";"p2";"02")))
$c.push(New object("name";"Blountsville";"phones";New object("p1";"00";"p2";"03")))
$crit:=New collection(New object("propertyPath";"phones.p2";"descending";True))
$c2:=$c.orderBy($crit)

```

.orderByMethod()

► Historique

.orderByMethod(*methodName* : Text { ; ...*extraParam* : expression }) : Collection

Paramètres	Type		Description
methodName	Text	->	Nom de la méthode utilisée pour trier la collection
extraParam	expression	->	Paramètre(s) à passer à la méthode
Résultat	Collection	<-	Copie triée de la collection (shallow copy)

Description

La fonction `.orderByMethod()` retourne une nouvelle collection contenant tous les éléments de la collection d'origine triés selon les critères définis par *methodName*.

Cette fonction retourne une **shallow copy (copie superficielle), ce qui signifie que les objets ou les collections présents dans les deux collections partagent la même référence. Si la collection d'origine est une collection partagée, la collection retournée est également une collection partagée.

Cette fonction ne modifie pas la collection d'origine.

Dans *methodName*, passez le nom d'une méthode qui compare deux valeurs et retourne `true` dans `$1.result` si la première valeur est inférieure à la seconde valeur. Si nécessaire, vous pouvez passer des paramètres supplémentaires à la méthode via *extraParam*.

- *methodName* reçoit les paramètres suivants :
 - \$1 (object), où :
 - `$1.value` (tout type) : valeur du premier élément à comparer
 - `$1.value2` (tout type) : valeur du second élément à comparer
 - \$2...\$N (tout type) : paramètres supplémentaires (*extraParam*)
- *methodName* doit fixer le paramètre suivant :
 - `$1.result` (booléen) : vrai si `$1.value < $1.value2`, faux sinon

Exemple 1

Voici le code de la méthode *NumAscending* :

```
var $c; $c2; $c3 : Collection
$c:=New collection
$c.push("33";"4";"1111";"222")
$c2:=$c.orderBy() // $c2=[1111,"222","33","4"], alphabetical order
$c3:=$c.orderByMethod("NumAscending") // $c3=[4,"33","222","1111"]
```

Vous souhaitez trier une collection de chaînes selon leur longueur :

```
$1.result:=Num($1.value)<Num($1.value2)
```

Exemple 2

Voici le code de la méthode *WordLength* :

```
var $fruits; $c2 : Collection
$fruits:=New collection("Orange";"Apple";"Grape";"pear";"Banana";"fig";"Blackberry";"Passion fruit")
$c2:=$fruits.orderByMethod("WordLength")
// $c2=[Passion fruit, Blackberry, Orange, Banana, Apple, Grape, pear, fig]
```

Vous souhaitez trier une collection par code de caractère ou par langage :

```
$1.result:=Length(String($1.value))>Length(String($1.value2))
```

Exemple 3

La méthode *sortCollection* :

```

var $strings1; $strings2 : Collection
$strings1:=New collection("Alpha";"Charlie";"alpha";"bravo";"Bravo";"charlie")

//using the character code:
$strings2:=$strings1.orderByMethod("sortCollection";sk character codes)
// result : ["Alpha","Bravo","Charlie","alpha","bravo","charlie"]

//using the language:
$strings2:=$string1s.orderByMethod("sortCollection";sk strict)
// result : ["alpha","Alpha","bravo","Bravo","charlie","Charlie"]

```

Paramètre(s) à passer à *methodName*

```

var$1Object
var$2Integer // option de tri

$1.result:=(Compare strings($1.value;$1.value2;$2)<0)

```

.pop()

► Historique

.pop() : any

Paramètres	Type		Description
Résultat	any	<-	Dernier élément de collection

Description

Lorsqu'il est appliqué à une collection vide, `.pop()` retourne *undefined*.

Cette fonction modifie la collection d'origine.

`.pop()`, combiné à `push()`, peut être utilisé pour implémenter une fonctionnalité last in first out de traitement des données empilées :

Exemple

La fonction `.pop()` supprime le dernier élément de la collection et le retourne comme résultat de la fonction.

```

var $stack : Collection
$stack:=New collection // $stack=[]
$stack.push(1;2) // $stack=[1,2]
$stack.pop() // $stack=[1] retourne 2
$stack.push(New collection(4;5)) // $stack=[[1,[4,5]]
$stack.pop() // $stack=[1] retourne [4,5]
$stack.pop() // $stack=[] retourne 1

```

.push()

► Historique

`.push(element : any { ;...elementN }) : Collection`

Paramètres	Type		Description
element	Mixed	->	Élément(s) à ajouter à la collection
Résultat	Collection	<-	Collection originale contenant des éléments ajoutés

Description

Vous souhaitez trier la collection résultante :

Cette fonction modifie la collection d'origine.

Exemple 1

```
var $col : Collection
$col:=New collection(1;2) // $col=[1,2]
$col.push(3) // $col=[1,2,3]
$col.push(6;New object("firstname";"John";"lastname";"Smith"))
// $col=[1,2,3,6,{firstname:John, lastname:Smith}]
```

Exemple 2

Vous souhaitez trier une collection de chaînes contenant des nombres par valeur plutôt que par ordre alphabétique :

```
var $col; $sortedCol : Collection
$col:=New collection(5;3;9) // $col=[5,3,9]
$sortedCol:=$col.push(7;50).sort()
// $col=[5,3,9,7,50]
// $sortedCol=[3,5,7,9,50]
```

.query()

► Historique

```
.query( queryString : Text ; ...value : any ) : Collection
.query( queryString : Text ; querySettings : Object ) : Collection
```

Paramètres	Type		Description
queryString	Text	->	Critère(s) de recherche
value	Mixed	->	Valeur(s) à comparer lors de l'utilisation de paramètre(s) dans la chaîne
querySettings	Object	->	Options de requête : paramètres, attributs
Résultat	Collection	<-	Élément(s) correspondant à queryString dans la collection

Description

La fonction `.query()` retourne tous les éléments d'une collection d'objets qui correspondent aux critères de recherche définis par `queryString` et (éventuellement) `value` ou `querySettings`. Si la collection d'origine est une collection partagée, la collection renournée est également une collection partagée.

Cette fonction ne modifie pas la collection d'origine.

Le paramètre `queryString` doit respecter la syntaxe suivante :

```
propertyPath comparator value {logicalOperator propertyPath comparator value}
```

Cet exemple retourne les personnes dont le nom contient "in" :

Les formules ne sont pas prises en charge par la fonction `collection.query()`, ni dans le paramètre `queryString` ni en tant que paramètre objet `formula`.

Exemple 1

```
var $c; $c2; $c3 : Collection
$c:=New collection
$c.push(New object("name";"Cleveland";"zc";35049))
$c.push(New object("name";"Blountsville";"zc";35031))
$c.push(New object("name";"Adger";"zc";35006))
$c.push(New object("name";"Clanton";"zc";35046))
$c.push(New object("name";"Clanton";"zc";35045))
$c2:=$c.query("name = :1";"Cleveland") //{$c2=[{name:Cleveland,zc:35049}]}
$c3:=$c.query("zc > 35040") //{$c3=[{name:Cleveland,zc:35049},{name:Clanton,zc:35046},{name:Clanton,zc:35045}]}
```

Exemple 2

```
var $c : Collection
$c:=New collection
$c.push(New object("name";"Smith";"dateHired";!22-05-2002!;"age";45))
$c.push(New object("name";"Wesson";"dateHired";!30-11-2017!))
$c.push(New object("name";"Winch";"dateHired";!16-05-2018!;"age";36))

$c.push(New object("name";"Sterling";"dateHired";!10-5-1999!;"age";Null))
$c.push(New object("name";"Mark";"dateHired";!01-01-2002!))
$c.push(New object("name";"Winch";"dateHired";!16-05-2018!;"age";36))

$c.push(New object("name";"Sterling";"dateHired";!10-5-1999!;"age";Null))
$c.push(New object("name";"Mark";"dateHired";!01-01-2002!))
```

Cet exemple retourne des personnes dont le nom ne commence pas par une chaîne d'une variable (saisie par l'utilisateur, par exemple) :

```
$col:=$c.query("name = :1";"@in@")
//{$col=[{name:Winch...},{name:Sterling...}]}
```

Cet exemple retourne des personnes dont l'âge n'est pas connu (propriété définie sur null ou indéfinie) :

```
$col:=$c.query("name # :1;$aString+@")
//if $astring="W"
//{$col=[{name:Smith...},{name:Sterling...},{name:Mark...}]}
```

Cet exemple retourne des personnes embauchées il y a plus de 90 jours :

```
$col:=$c.query("age=null") //placeholders not allowed with "null"
//{$col=[{name:Wesson...},{name:Sterling...},{name:Mark...}]}
```

Vous trouverez plus d'exemples de requêtes dans la page `dataClass.query()`.

```
$col:=$c.query("dateHired < :1;(Current date-90))
//$col=[{name:Smith...},{name:Sterling...},{name:Mark...}] si la date du jour est 01/10/2018
```

Exemple 3

Vous trouverez plus d'exemples de requêtes dans la page `dataClass.query()`.

.reduce()

► Historique

`.reduce(methodName : Text) : any`

`.reduce(methodName : Text ; initialValue : any { ; ...param : expression }) : any`

Paramètres	Type		Description
methodName	Text	->	Nom de la fonction à appeler pour traiter les éléments de collection
initialValue	Text, Number, Object, Collection, Date, Boolean	->	Valeur à utiliser comme premier argument du premier appel de <i>methodName</i>
param	expression	->	Paramètre(s) à passer à <i>methodName</i>
Résultat	Text, Number, Object, Collection, Date, Boolean	<-	Résultat de la valeur de l'accumulateur

Description

La fonction `.reduce()` applique la méthode callback *methodName* à un accumulateur et à chaque élément de la collection (de gauche à droite) pour le réduire à une valeur unique.

Cette fonction ne modifie pas la collection d'origine.

Dans *methodName*, passez le nom de la méthode à utiliser pour évaluer les éléments de la collection, ainsi que son ou ses paramètres dans *param* (facultatif). *methodName* prend chaque élément de la collection et effectue toutes les opérations souhaitées pour accumuler le résultat dans `$1.accumulator`, qui est retourné dans `$1.value`.

Vous pouvez passer la valeur pour initialiser l'accumulateur dans *initialValue*. S'il est omis, `$1.accumulator` commence par *Undefined*.

methodName reçoit les paramètres suivants :

- dans `$1.value` : valeur de l'élément à évaluer
- dans `$2` : *param*
- dans `$N...` : *paramN...*

methodName doit fixer le(s) paramètre(s) suivant(s) :

- `$1.accumulator`: valeur à modifier par la fonction et qui est initialisée par *initialValue*.
- `$1.stop` (booléen, optionnel) : true pour stopper le rétroappel de méthode. La valeur retournée est la dernière calculée.

Exemple 1

```
C_COLLECTION($c)
$c:=New collection(5;3;5;1;3;4;4;6;2;2)
$r:=$c.reduce("Multiply";1) //retourne 86400
```

Cet exemple permet de réduire plusieurs éléments de collection en un seul élément :

```
If(Value type($1.value)=Is real)
    $1.accumulator:=$1.accumulator*$1.value
End if
```

Exemple 2

Avec la méthode *Flatten* suivante :

```
var $c;$r : Collection
$c:=New collection
$c.push(New collection(0;1))
$c.push(New collection(2;3))
$c.push(New collection(4;5))
$c.push(New collection(6;7))
$r:=$c.reduce("Flatten") //r=[0,1,2,3,4,5,6,7]
```

Avec la méthode *NumberGreaterThan0* suivante :

```
If($1.accumulator=NULL)
    $1.accumulator:=New collection
End if
$1.accumulator.combine($1.value)
```

.remove()

► Historique

.remove(*index* : Integer { ; *howMany* : Integer }) : Collection

Paramètres	Type		Description
<i>index</i>	Integer	->	Élément à partir duquel la suppression peut commencer
<i>howMany</i>	Integer	->	Nombre d'éléments à supprimer, ou 1 élément si omis
Résultat	Collection	<-	Collection d'origine sans élément(s) supprimé(s)

Description

Dans *index*, passez la position où vous souhaitez supprimer l'élément de la collection.

Cette fonction modifie la collection d'origine.

La fonction `.remove()` supprime un ou plusieurs élément(s) de la position d'*index* spécifiée dans la collection et retourne la collection modifiée.

Attention : A noter que la numérotation des éléments de collection débute à 0. Si *index* est supérieur à la longueur de la collection, l'*index* de départ réel sera fixé à la longueur de la collection.

- Si *index* < 0, il est recalculé comme *index:=index+lenth* (il est considéré comme le décalage par rapport à la fin de la collection).
- Si la valeur recalculée est négative, *index* prend la valeur 0.
- Si valeur recalculée > longueur de la collection, *index* prend comme valeur la taille de la collection.

Dans *howMany*, passez le nombre d'éléments à supprimer de l'*index*. Si *howMany* n'est pas spécifié, un élément est supprimé.

Si vous essayez de supprimer un élément d'une collection vide, la méthode ne fait rien (aucune erreur n'est générée).

Exemple

```
var $col : Collection
$col:=New collection("a";"b";"c";"d";"e";"f";"g";"h")
$col.remove(3) // $col=["a","b","c","e","f","g","h"]
$col.remove(3;2) // $col=["a","b","c","g","h"]
$col.remove(-8;1) // $col=["b","c","g","h"]
$col.remove(-3;1) // $col=["b","g","h"]
```

.resize()

► Historique

.resize(*size* : Integer { ; *defaultValue* : any }) : Collection

Paramètres	Type		Description
size	Integer	->	Nouvelle taille de la collection
defaultValue	Number, Text, Object, Collection, Date, Boolean	->	Valeur par défaut pour remplir de nouveaux éléments
Résultat	Collection	<-	Collection d'origine redimensionnée

Description

La fonction `.resize()` définit la longueur de la collection sur la nouvelle taille spécifiée et retourne la collection redimensionnée.

Cette fonction modifie la collection d'origine.

- Si *size* est inférieure à la longueur de la collection, les éléments excédentaires sont retirés de la collection.
- Si *size* > longueur de la collection, *size* est la nouvelle longueur de la collection.

Par défaut, les nouveaux éléments sont remplis par des valeurs `null`. Vous pouvez indiquer la valeur à remplir dans les éléments ajoutés à l'aide du paramètre *defaultValue*.

Exemple

```
var $c : Collection
$c:=New collection
$c.resize(10) // $c=[null,null,null,null,null,null,null,null,null,null]

$c:=New collection
$c.resize(10;0) // $c=[0,0,0,0,0,0,0,0,0,0]

$c:=New collection(1;2;3;4;5)
$c.resize(10;New object("name";"X")) // $c=[1,2,3,4,5,{name:X},{name:X},{name:X},{name:X},{name:X}]

$c:=New collection(1;2;3;4;5)
$c.resize(2) // $c=[1,2]
```

.reverse()

► Historique

.reverse() : Collection

Paramètres	Type		Description
Résultat	Collection	<-	Copie inversée de la collection

Description

La fonction `.reverse()` retourne une copie complète de la collection avec tous ses éléments dans l'ordre inverse. Si la collection d'origine est une collection partagée, la collection retournée est également une collection partagée.

Cette fonction ne modifie pas la collection d'origine.

Exemple

```
var $c; $c2 : Collection
$c:=New collection(1;3;5;2;4;6)
$c2:=$c.reverse() // $c2=[6,4,2,5,3,1]
```

.shift()

► Historique

`.shift()` : any

Paramètres	Type		Description
Résultat	any	<-	Premier élément de collection

Description

Si la collection est vide, cette méthode ne fait rien.

Cette fonction modifie la collection d'origine.

Si la collection est vide, cette méthode ne fait rien.

Exemple

```
var $c : Collection
var $val : Variant
$c:=New collection(1;2;4;5;6;7;8)
$val:=$c.shift()
// $val=1
// $c=[2,4,5,6,7,8]
```

.slice()

► Historique

`.slice(startFrom : Integer { ; end : Integer })` : Collection

Paramètres	Type		Description
startFrom	Integer	->	Numéro de l'élément de départ (inclus)
end	Integer	->	Position de fin (non incluse)
Résultat	Collection	<-	Nouvelle collection contenant des éléments scindées (copie superficielle)

Description

La fonction `.slice()` retourne une partie d'une collection dans une nouvelle collection, sélectionnée de l'index *startFrom* à l'index de *fin* (fin non incluse). Cette fonction retourne une *copie superficielle* de la collection. Si la collection d'origine est une collection partagée, la collection retournée est également une collection partagée.

Cette fonction ne modifie pas la collection d'origine.

La collection retournée contient l'élément spécifié par *startFrom* et tous les éléments suivants jusqu'à l'élément spécifié par *end* (mais non compris). Si seul le paramètre *startFrom* est spécifié, la collection retournée contient tous les éléments de *startFrom* au dernier élément de la collection d'origine.

- Si *startFrom* < 0, le paramètre est recalculé comme *startFrom:=startFrom+length* (la fin de la collection est considérée comme point de départ du calcul de la position).
- Si la valeur calculée est négative, *startFrom* prend la valeur 0.
- Si *end* < 0 , le paramètre est recalculé comme *end:=end+length*.
- Si *end* < *startFrom* (valeurs passées ou recalculées), la fonction ne fait rien.

Exemple

```
var $c; $nc : Collection
$c:=New collection(1;2;3;4;5)
$nc:=$c.slice(0;3) // $nc=[1,2,3]
$nc:=$c.slice(3) // $nc=[4,5]
$nc:=$c.slice(1;-1) // $nc=[2,3,4]
$nc:=$c.slice(-3;-2) // $nc=[3]
```

.some()

► Historique

`.some(methodName : Text { ; ...param : any }) : Boolean`

`.some(startFrom : Integer ; methodName : Text { ; ...param : any }) : Boolean`

Paramètres	Type		Description
startFrom	Integer	->	Elément à partir duquel débuter l'évaluation
methodName	Text	->	Nom de la méthode à appeler pour l'évaluation
param	Mixed	->	Paramètre(s) à passer à <i>methodName</i>
Résultat	Booléen	<-	Vrai si au moins un élément a réussi le test

Description

La fonction `.push()` ajoute un ou plusieurs *élément(s)* à la fin de l'instance de collection et retourne la collection modifiée.

Dans *methodName*, passez le nom de la méthode à utiliser pour évaluer les éléments de la collection, ainsi que son ou ses paramètre(s) dans *param* (optionnel). *methodName* peut effectuer tout test, avec ou sans paramètres. Dans *methodName*, passez le nom de la méthode à utiliser pour évaluer les éléments de la collection, ainsi que son ou ses paramètre(s) dans *param* (optionnel).

methodName reçoit les paramètres suivants :

- dans `$1.value` : valeur de l'élément à évaluer
- in `$2`: *param*
- dans `$N...` : *param2...paramN*

methodName doit fixer le(s) paramètre(s) suivant(s) :

- `$1.result` (booléen) : true si l'évaluation de la valeur de l'élément est réussie, sinon false.

- `$1.stop` (booléen, optionnel) : true pour stopper le rétroappel de méthode. La valeur renvoyée est la dernière calculée.

La fonction `.lastIndexOf()` recherche l'expression `toSearch` parmi les éléments de la collection et retourne le numéro d'élément de la dernière occurrence trouvée, ou -1 si aucune occurrence n'a été trouvée.

By default, `.every()` tests the whole collection. Optionally, you can pass in `startFrom` the index of the element from which to start the test.

- Si `startFrom >=` la longueur de la collection, False est retourné, ce qui signifie que la collection n'est pas testée.
- Si `startFrom < 0`, il est considéré comme le décalage depuis la fin de la collection.
- Si `startFrom = 0`, l'ensemble de la collection est évalué (défaut).

Exemple

```
var $c : Collection
var $b : Boolean
$c:=New collection
$c.push(-5;-3;-1;-4;-6;-2)
$b:=$c.some("NumberGreaterThan0") // retourne false
$c.push(1)
$b:=$c.some("NumberGreaterThan0") // retourne true

$c:=New collection
$c.push(1;-5;-3;-1;-4;-6;-2)
$b:=$c.some("NumberGreaterThan0") // $b=true
$b:=$c.some(1;"NumberGreaterThan0") // $b=false
```

`methodName` doit fixer le paramètre suivant :

```
$1.result:=$1.value>0
```

.sort()

► Historique

`.sort(methodName : Text { ; ...extraParam : any }) : Collection`

Paramètres	Type		Description
methodName	Text	->	Nom de la méthode utilisée pour trier la collection
extraParam	any	->	Paramètre(s) à passer à la méthode
Résultat	Collection	<-	Collection d'origine triée

Description

La fonction `.sort()` trie les éléments de la collection d'origine et retourne également la collection triée.

Cette fonction modifie la collection d'origine.

Si `.sort()` est appelé sans paramètre, seules les valeurs scalaires (numérique, texte, date, booléens) sont triées. Les éléments sont triés par défaut par ordre croissant, en fonction de leur type.

Si vous souhaitez trier les éléments de la collection dans un autre ordre ou trier n'importe quel type d'élément, vous devez fournir, dans `methodName`, une méthode de comparaison qui compare deux valeurs et retourne true dans `$1.result` si la première valeur est inférieure à la deuxième valeur. Si nécessaire, vous pouvez passer des paramètres supplémentaires à la méthode via `extraParam`.

- `methodName` reçoit les paramètres suivants :

- \$1 (object), où :
 - \$1.value (tout type) : valeur du premier élément à comparer
 - \$1.value2 (tout type) : valeur du second élément à comparer
- \$2...\$N (tout type) : paramètres supplémentaires (extraParam)

methodName doit fixer le paramètre suivant : * \$1.result (booléen): vrai si \$1.value < \$1.value2, faux sinon

Si la collection contient des éléments de différents types, ils sont d'abord groupés par type et triés par la suite. Si *attributePath* est un attribut d'objet qui contient des valeurs de types différents, elles sont groupées par type et triées ensuite.

1. null
2. booléens
3. chaînes
4. nombres
5. objets
6. collections
7. dates

Exemple 1

```
var $col; $col2 : Collection
$col:=New collection("Tom";5;"Mary";3;"Henry";1;"Jane";4;"Artie";6;"Chip";2)
$col2:=$col.sort() // $col2=[ "Artie", "Chip", "Henry", "Jane", "Mary", "Tom", 1, 2, 3, 4, 5, 6]
// $col=[ "Artie", "Chip", "Henry", "Jane", "Mary", "Tom", 1, 2, 3, 4, 5, 6]
```

Exemple 2

```
var $col; $col2 : Collection
$col:=New collection(10;20)
$col2:=$col.push(5;3;1;4;6;2).sort() // $col2=[1,2,3,4,5,6,10,20]
```

Exemple 3

```
var $col; $col2; $col3 : Collection
$col:=New collection(33;4;66;1111;222)
$col2:=$col.sort() // tri numérique : [4,33,66,222,1111]
$col3:=$col.sort("numberOrder") // tri alphabétique : [1111,222,33,4,66]
```

```
// méthode projet numberOrder
var $1 : Object
$1.result:=String($1.value)<String($1.value2)
```

.sum()

► Historique

.sum({ *propertyPath* : Text }) : Real

Paramètres	Type		Description
<i>propertyPath</i>	Text	->	Chemin de propriété d'objet à utiliser pour évaluer les valeurs
Résultat	Réel	<-	Somme des valeurs de collection

Description

Seuls les éléments ayant une valeur numérique sont pris en compte pour le calcul (les autres types d'éléments sont ignorés).

Seuls les éléments ayant une valeur numérique sont pris en compte pour le calcul (les autres types d'éléments sont ignorés).

Si la collection contient des objets, passez le paramètre *propertyPath* si vous souhaitez désigner la propriété dont vous voulez connaître la moyenne.

`.sum()` retourne 0 si :

- la collection est vide,
- la collection ne contient pas d'éléments numériques,
- *propertyPath* n'est pas trouvé dans la collection.

Exemple 1

```
var $col : Collection
var $vSum : Real
$col:=New collection(10;20;"Monday";True;2)
$vSum:=$col.sum() //32
```

Exemple 2

```
var $col : Collection
var $vSum : Real
$col:=New collection
$col.push(New object("name";"Smith";"salary";10000))
$col.push(New object("name";"Wesson";"salary";50000))
$col.push(New object("name";"Gross";"salary";10500,5))
$vSum:=$col.sum("salary") //vSum=70500,5
```

.unshift()

► Historique

`.unshift(value : any { ;...valueN : any }) : Collection`

Paramètres	Type		Description
value	Text, Number, Object, Collection, Date	->	Valeur(s) à insérer au début de la collection
Résultat	Réel	<-	Collection contenant des éléments ajoutés

Description

Si plusieurs valeurs sont passées, elles sont insérées toutes en même temps, ce qui signifie qu'elles apparaissent dans la collection résultante dans le même ordre que dans la liste d'arguments.

Cette fonction modifie la collection d'origine.

Si plusieurs valeurs sont passées, elles sont insérées toutes en même temps, ce qui signifie qu'elles apparaissent dans la collection résultante dans le même ordre que dans la liste d'arguments.

Exemple

```
var $c : Collection
$c:=New collection(1;2)
$c.unshift(4) // $c=[4,1,2]
$c.unshift(5) // $c=[5,4,1,2]
$c.unshift(6;7) // $c=[6,7,5,4,1,2]
```

CryptoKey

La classe `CryptoKey` du langage 4D contient une paire de clés de chiffrement asymétrique.

Cette classe est disponible depuis le "class store" de `4D`.

Exemple

L'extrait de code suivant illustre la signature et la vérification d'un message à l'aide d'une nouvelle paire de clés ECDSA, afin de créer, par exemple, un token Web JSON ES256.

```
// Générer une nouvelle paire de clés ECDSA
$key:=4D.CryptoKey.new(New object("type";"ECDSA";"curve";"prime256v1"))

// Obtenir une signature en base64
$message:="hello world"
$signature:=$key.sign($message;New object("hash";"SHA256"))

// Vérifier la signature
$status:=$key.verify($message;$signature;New object("hash";"SHA256"))
ASSERT($status.success)
```

Sommaire

<code>4D.CryptoKey.new(settings : Object) : 4D.CryptoKey</code>	crée un nouvel objet <code>4D.CryptoKey</code> encapsulant une paire de clés de chiffrement
<code>.curve : Texte</code>	nom de la courbe normalisée de la clé
<code>.decrypt(message : Text ; options : Object) : Object</code>	déchiffre le paramètre <i>message</i> à l'aide de la clé privée
<code>.encrypt(message : Text ; options : Object) : Text</code>	crypte le paramètre <i>message</i> à l'aide de la clé publique
<code>.getPrivateKey() : Text</code>	retourne la clé privée de l'objet <code>CryptoKey</code>
<code>.getPublicKey() : Text</code>	retourne la clé publique de l'objet <code>CryptoKey</code>
<code>.sign (message : Text ; options : Text) : Text</code>	signe la représentation utf8 de la chaîne <i>message</i>
<code>.size : Integer</code>	la taille de la clé en octets
<code>.type : Texte</code>	Type de clé : "RSA", "ECDSA" ou "PEM"
<code>.verify(message : Text ; signature : Text ; options : Object) : object</code>	vérifie la signature base64 par rapport à la représentation utf8 du <i>message</i>

4D.CryptoKey.new()

► Historique

4D.CryptoKey.new(*settings* : Object) : 4D.CryptoKey

Paramètres	Type		Description
settings	Object	->	Paramètres pour générer ou charger une paire de clés
result	4D.CryptoKey	<-	Objet contenant une paire de clés de chiffrement

La fonction `4D.CryptoKey.new()` crée un nouvel objet `4D.CryptoKey` encapsulant une paire de clés de chiffrement, en fonction du paramètre d'objet *settings*. Elle permet de générer une nouvelle clé RSA ou ECDSA, ou de charger une paire de clés existante à partir de la définition PEM.

settings

Propriété	Type	Description
<code>curve</code>	Texte	Nom de la courbe ECDSA
<code>pem</code>	Texte	Définition PEM d'une clé de chiffrement à charger
<code>size</code>	entier	Taille de la clé RSA en octets
<code>type</code>	Texte	Type de clé : "RSA", "ECDSA", ou "PEM"

CryptoKey

L'objet `CryptoKey` retourné encapsule une paire de clés de chiffrement. C'est un objet partagé et peut être alors utilisé par de multiples traitements 4D simultanés.

.curve

► Historique

`.curve` : Texte

Défini uniquement pour les clés ECDSA : le nom de la courbe normalisée de la clé. Généralement "prime256v1" pour ES256 (défaut), "secp384r1" pour ES384, "secp521r1" pour ES512.

.decrypt()

► Historique

`.decrypt(message : Text ; options : Object) : Object`

Paramètres	Type		Description
<code>message</code>	Text	->	Chaine message à déchiffrer à l'aide de <code>options.encodingEncrypted</code> et <code>decrypted</code> .
<code>options</code>	Object	->	Options de décodage
Résultat	Object	<-	Statut

La fonction `.decrypt()` déchiffre le paramètre *message* à l'aide de la clé privée. L'algorithme utilisé dépend du type de clé.

La clé doit être une clé RSA, l'algorithme est RSA-OAEP (voir [RFC 3447](#)).

options

Propriété	Type	Description
hash	Texte	Algorithme de hachage à utiliser. Par exemple : "SHA256", "SHA384" ou "SHA512".
encodingEncrypted	Texte	Chiffrement utilisé pour convertir le paramètre <code>message</code> en représentation binaire à déchiffrer. Peut être "Base64" ou "Base64URL". La valeur par défaut est "Base64".
encodingDecrypted	Texte	Encodage utilisé pour convertir le message binaire déchiffré en chaîne de résultat. Peut être "UTF-8", "Base64" ou "Base64URL". La valeur par défaut est "UTF-8".

Résultat

La clé doit être une clé RSA, l'algorithme est RSA-OAEP (voir [RFC 3447](#)).

Propriété	Type	Description
success	boolean	True si le message a été déchiffré avec succès
result	Texte	Message déchiffré et décodé à l'aide de <code>options.encodingDecrypted</code>
errors	collection	Si <code>success</code> est mis sur <code>false</code> , il peut contenir une collection d'erreurs

La fonction renvoie un objet "status" avec la propriété `success` définie sur `true` si le `message` a pu être déchiffré avec succès.

.encrypt()

► Historique

`.encrypt(message : Text ; options : Object) : Text`

Paramètres	Type		Description
message	Text	->	Chaîne message à chiffrer à l'aide de <code>options.encodingDecrypted</code> et <code>encrypted</code> .
options	Object	->	Options de chiffrement
Résultat	Text	<-	Message chiffré et encodé à l'aide de <code>options.encodingEncrypted</code>

La fonction `.encrypt()` crypte le paramètre `message` à l'aide de la clé publique. L'algorithme utilisé dépend du type de clé.

La clé doit être une clé RSA, l'algorithme est RSA-OAEP (voir [RFC 3447](#)).

options

Propriété	Type	Description
hash	Texte	Algorithme de hachage à utiliser. Par exemple : "SHA256", "SHA384" ou "SHA512".
encodingEncrypted	Texte	Chiffrement utilisé pour convertir le message chiffré binaire en chaîne de résultat. Peut être "Base64" ou "Base64URL". La valeur par défaut est "Base64".
encodingDecrypted	Texte	Chiffrement utilisé pour convertir le paramètre <code>message</code> en représentation binaire à chiffrer. Peut être "UTF-8", "Base64" ou "Base64URL". La valeur par défaut est "UTF-8".

Résultat

La clé doit être une clé RSA, l'algorithme est RSA-OAEP (voir [RFC 3447](#)).

.getPrivateKey()

► Historique

`.getPrivateKey() : Text`

Paramètres	Type		Description
Résultat	Text	<-	Clé primaire au format PEM

La fonction `.getPrivateKey()` retourne la clé privée de l'objet `CryptoKey` au format PEM, ou une chaîne vide si aucune n'est disponible.

Résultat

La fonction `.getPrivateKey()` retourne la clé privée de l'objet `CryptoKey` au format PEM, ou une chaîne vide si aucune n'est disponible.

`.getPublicKey()`

► Historique

`.getPublicKey()` : Text

Paramètres	Type		Description
Résultat	Text	<-	Clé publique au format PEM

La fonction `.getPublicKey()` retourne la clé publique de l'objet `CryptoKey` au format PEM, ou une chaîne vide si aucune n'est disponible.

Résultat

La fonction `.getPublicKey()` retourne la clé publique de l'objet `CryptoKey` au format PEM, ou une chaîne vide si aucune n'est disponible. ---## `.pem`

► Historique

`.pem` : Text

Définition PEM d'une clé de chiffrement à charger. Si la clé est une clé privée, la clé publique RSA ou ECDSA en sera déduite.

`.sign()`

► Historique

`.sign (message : Text ; options : Text)` : Text

Paramètres	Type		Description
message	Text	->	Chaîne message à signer
options	Object	->	Options de signature
Résultat	Text	<-	Signature en représentation Base64 ou Base64URL, selon l'option "encoding"

La fonction `.sign()` signe la représentation utf8 de la chaîne `message` à l'aide des clés objet `CryptoKey` et des `options` fournies. Elle retourne sa signature au format base64 ou base64URL, selon la valeur de l'attribut `options.encoding` que vous avez passé.

`.type` : Texte

`options`

Propriété	Type	Description
hash	Texte	Algorithme de hachage à utiliser. Par exemple : "SHA256", "SHA384" ou "SHA512". Lorsqu'elle est utilisée pour produire un JWT, la taille du hachage doit correspondre à la taille de l'algorithme PS@, ES@, RS@ ou PS@
encodingEncrypted	Texte	Chiffrement utilisé pour convertir le message chiffré binaire en chaîne de résultat. Peut être "Base64" ou "Base64URL". La valeur par défaut est "Base64".
pss	boolean	Utilise le Probabilistic Signature Scheme (PSS). Ignoré si la clé n'est pas une clé RSA. Passez <code>true</code> lors de la production d'un JWT pour l'algorithme PS@
encoding	Texte	Représentation à utiliser pour la signature. Valeurs possibles : "Base64" ou "Base64URL". La valeur par défaut est "Base64".

Résultat

`CryptoKey` doit contenir une clé privée valide.

.size

► Historique

`.size` : Integer

Défini uniquement pour les clés RSA : la taille de la clé en octets. Habituellement 2048 (par défaut).

.type

► Historique

`.type` : Texte

Type de clé : "RSA", "ECDSA" ou "PEM"

- "RSA" : paire de clés RSA, utilise `settings.size` pour la taille `.size`.
- "ECDSA" : paire de clés Elliptic Curve Digital Signature Algorithm, utilise `settings.curve` pour la propriété `curve`. A noter que les clés ECDSA ne peuvent pas être utilisées pour le chiffrement, mais uniquement pour la signature.
- "PEM" : Définition de paire de clés au format PEM format, utilise `settings.pem` pour la propriété `.pem`.

.verify()

► Historique

`.verify(message : Text ; signature : Text ; options : Object) : object`

Paramètres	Type		Description
message	Text	->	Chaîne message utilisée pour générer la signature
signature	Text	->	Signature à vérifier, en représentation Base64 ou Base64URL, selon la valeur <code>options.encoding</code>
options	Object	->	Options de signature
Résultat	Object	<-	Statut de la vérification

Si la signature n'a pas pu être vérifiée car elle n'a pas été signée avec le même `message`, la clé ou l'algorithme, l'objet `status` retourné contient une collection d'erreurs dans `status.errors`.

La fonction `.verify()` vérifie la signature base64 par rapport à la représentation utf8 du `message` à l'aide des clés objet `CryptoKey` et des `options` fournies.

options

Propriété	Type	Description
hash	Texte	Algorithme de hachage à utiliser. Par exemple : "SHA256", "SHA384" ou "SHA512". Lorsqu'elle est utilisée pour produire un JWT, la taille du hachage doit correspondre à la taille de l'algorithme PS@, ES@, RS@ ou PS@
pss	boolean	Utilise le Probabilistic Signature Scheme (PSS). Ignoré si la clé n'est pas une clé RSA. Passez <code>true</code> lors de la vérification d'un JWT pour l'algorithme PS@
encoding	Texte	Représentation de la signature fournie. Valeurs possibles : "Base64" ou "Base64URL". La valeur par défaut est "Base64".

Résultat

`CryptoKey` doit contenir une clé publique valide.

La fonction retourne un objet status avec la propriété `success` définie sur `true` si le `message` a pu être déchiffré avec succès (c'est-à-dire si la signature est correspondante).

Propriété	Type	Description
success	boolean	True si la signature correspond au message
errors	collection	Si <code>success</code> est mis sur <code>false</code> , il peut contenir une collection d'erreurs

DataClass

Une [dataclass](#) fournit une interface objet à une table de la base de données. Toutes les dataclasses d'une application 4D sont disponibles en tant que propriété du [datastore](#) `ds`.

Sommaire

[`.attributeName : DataClassAttribute`](#)

objets disponibles directement en tant que propriétés

[`.all \({ settings : Object } \) : 4D.EntitySelection`](#)

interroge le datastore pour trouver toutes les entités de la dataclass et les renvoie en tant qu'"entity selection"

[`.clearRemoteCache\(\)`](#)

vide le cache ORDA d'une dataclass

[`.fromCollection\(objectCol : Collection { ; settings : Object } \) : 4D.EntitySelection`](#)

modifie ou crée des entités dans la dataclass en utilisant la collection d'objets `objectCol` et retourne l'entity selection correspondante

[`.get\(primaryKey : Integer { ; settings : Object } \) : 4D.Entity`](#)

[`.get\(primaryKey : Text { ; settings : Object } \) : 4D.Entity`](#)

interroge la dataclass pour récupérer l'entité correspondant au paramètre `primaryKey`

[`.getCount\(\) : Integer`](#)

retourne le nombre d'entités dans une dataclass

[`.getDataStore\(\) : cs.DataStore`](#)

retourne le datastore de la dataclass

[`. getInfo\(\) : Object`](#)

retourne un objet qui fournit des informations sur la dataclass

[`.getRemoteCache\(\) : Object`](#)

retourne un objet qui contient le cache ORDA pour la dataclass

[`.new\(\) : 4D.Entity`](#)

crée en mémoire et renvoie une nouvelle entité vide pour la dataclass

[`.newSelection\({ keepOrder : Integer } \) : 4D.EntitySelection`](#)

crée en mémoire une entity selection vide, non partageable, liée à la dataclass

[`.query\(queryString : Text { ; ...value : any } { ; querySettings : Object } \) : 4D.EntitySelection`](#)

[`.query\(formula : Object { ; querySettings : Object } \) : 4D.EntitySelection`](#)

recherche les entités répondant aux critères de recherche spécifiés dans `queryString` ou `formula` et (optionnellement) dans `value`toutes les entités de la dataclass

[`.setRemoteCacheSettings\(settings : Object\)`](#)

définit le timeout et la taille maximum du cache ORDA pour la dataclass

[`.attributeName`](#)

.attributeName : DataClassAttribute

Description

Les attributs des dataclasses sont des objets disponibles directement en tant que propriétés de ces classes.

Les objets renvoyés sont du type `DataClassAttribute`. Ces objets ont des propriétés que vous pouvez utiliser et lire pour obtenir des informations sur vos attributs de dataclass.

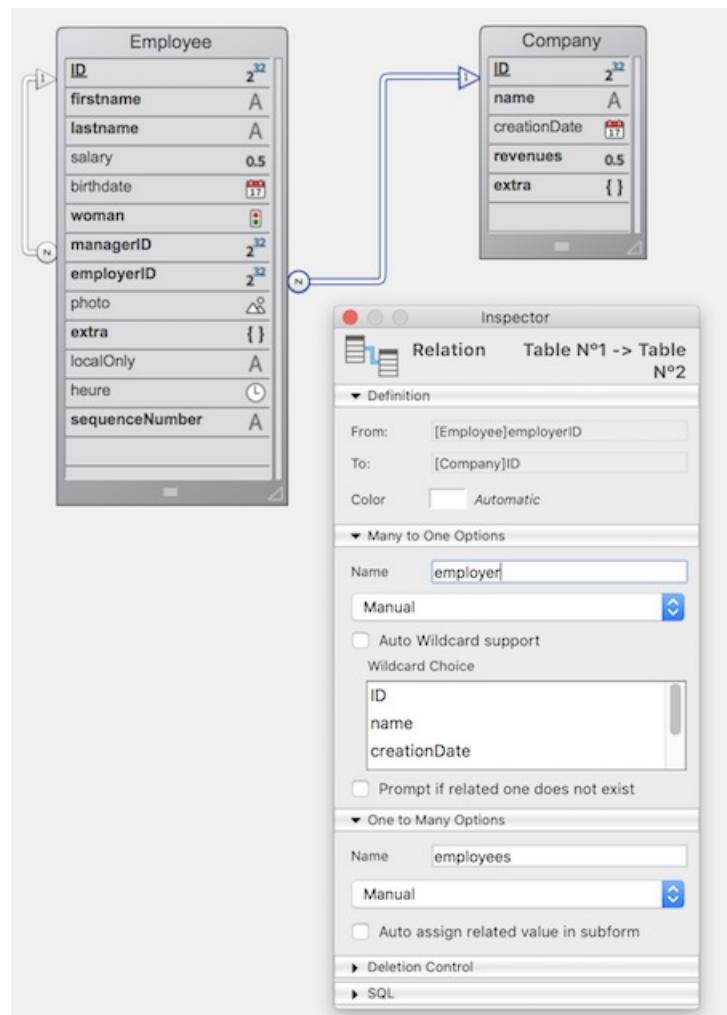
Les objets de l'attribut Dataclass peuvent être modifiés, mais la structure sous-jacente de la base de données ne sera pas altérée.

Exemple 1

```
$salary:=ds.Employee.salary //retourne l'attribut salary dans la dataclass Employee  
$compCity:=ds.Company["city"] //retourne l'attribut city dans la dataclass Company
```

Exemple 2

Considérez la structure suivante d'une base :



```

var $firstnameAtt;$employerAtt;$employeesAtt : Object

$firstnameAtt:=ds.Employee.firstname
//{name:firstname,kind:storage,fieldType:0,type:string,fieldNumber:2,indexed:true,
//keyWordIndexed:false,autoFilled:false,mandatory:false,unique:false}

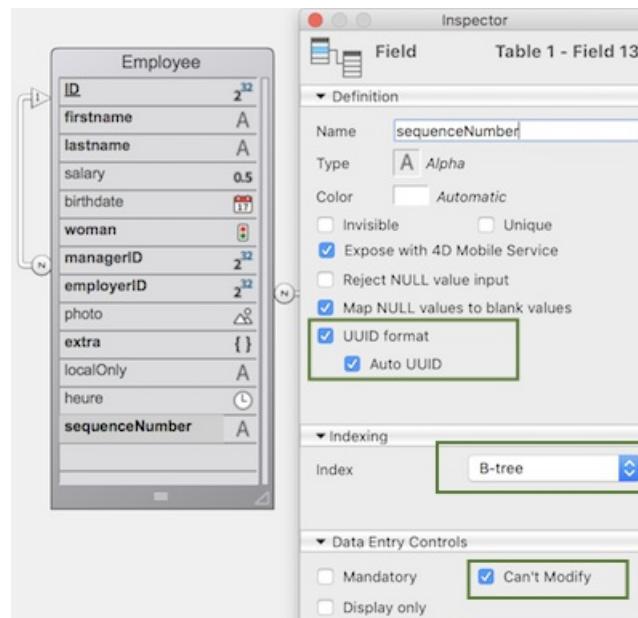
$employerAtt:=ds.Employee.employer
//{name:employer,kind:relatedEntity,relatedDataClass:Company,
//fieldType:38,type:Company,inverseName:employees}
//38=Is object

$employeesAtt:=ds.Company.employees
//{name:employees,kind:relatedEntities,relatedDataClass:Employee,
//fieldType:42,type:EmployeeSelection,inverseName:employer}
//42=Is collection

```

Exemple 3

Considérez les propriétés de table suivantes :



```

var $sequenceNumberAtt : Object
$sequenceNumberAtt=ds.Employee.sequenceNumber
//{name:sequenceNumber,kind:storage,fieldType:0,type:string,fieldNumber:13,
//indexed:true,keyWordIndexed:false,autoFilled:true,mandatory:false,unique:true}

```

.all()

► Historique

.all ({ settings : Object }) : 4D.EntitySelection

Paramètres	Type		Description
settings	Object	->	Option de création : contexte
Résultat	4D.EntitySelection	<-	Références vers toutes les entités de la dataclass

Description

La fonction `.all()` interroge le datastore pour trouver toutes les entités de la dataclass et les renvoie en tant qu'"entity selection".

Les entités sont renvoyées dans l'ordre par défaut, qui est initialement l'ordre dans lequel elles ont été créées. Notez cependant que, si des entités ont été supprimées et que de nouvelles entités ont été ajoutées, l'ordre par défaut ne reflète plus l'ordre de création.

Si aucune entité n'est trouvée, une entity selection vide est renvoyée.

Le mode lazy loading est appliqué.

settings

Dans le paramètre optionnel *settings*, vous pouvez passer un objet contenant des options supplémentaires. La propriété suivante est prise en charge :

Propriété	Type	Description
context	Text	Nom du contexte d'optimisation appliquée à l'entity selection. Ce contexte sera utilisé par le code qui manipule l'entity selection afin de bénéficier de l'optimisation. Cette fonctionnalité est conçue pour le traitement client/serveur ORDA .

Pour obtenir le nombre total d'entités dans une dataclass, il est recommandé d'utiliser la fonction [getCount\(\)](#) qui est plus optimisée que l'expression `ds.myClass.all().length`.

Exemple

```
var $allEmp : cs.EmployeeSelection  
$allEmp:=ds.Employee.all()
```

.clearRemoteCache()

► Historique

.clearRemoteCache() | Paramètres | Type | | Description | | ----- | ---- |::| ----- | | | | Ne requiert aucun paramètre |

Description

La fonction `.clearRemoteCache()` vide le cache ORDA d'une dataclass.

Cette fonction ne réinitialise pas les valeurs de `timeout` et `maxEntries`.

Exemple

```

var $ds : 4D.DataStoreImplementation
var $persons : cs.PersonsSelection
var $p : cs.PersonsEntity
var $cache : Object
var $info : Collection
var $text : Text

$ds:=Open datastore(New object("hostname"; "www.myserver.com"); "myDS")

$persons:=$ds.Persons.all()
$text:=""
For each ($p; $persons)
    $text:=$p.firstname+" lives in "+$p.address.city+ " / "
End for each

$cache:=$ds.Persons.getRemoteCache()

$ds.Persons.clearRemoteCache()
// Cache of the Persons dataclass = {timeout:30;maxEntries:30000;stamp:255;entries:[]}

```

.fromCollection()

► Historique

.fromCollection(*objectCol* : Collection { ; *settings* : Object }) : 4D.EntitySelection

Paramètres	Type		Description
<i>objectCol</i>	Collection	->	Collection d'objets à faire correspondre à des entités
<i>settings</i>	Object	->	Option de création : contexte
Résultat	4D.EntitySelection	<-	Entity selection issue de la collection

Description

La fonction `.fromCollection()` modifie ou crée des entités dans la dataclass en utilisant la collection d'objets *objectCol* et retourne l'entity selection correspondante.

Dans le paramètre *objectCol* passez une collection d'objets destinée à créer ou à modifier des entités de la dataclass. Les noms des propriétés doivent correspondre à ceux des attributs de la dataclass. Si un nom de propriété n'existe pas dans la dataclass, il est ignoré. Si une valeur d'attribut n'est pas définie dans la collection pour une entité créée, l'attribut prend la valeur Null.

La correspondance entre les objets de la collection et les entités est effectuée au niveau des noms d'attributs et de leur type de données. Si une propriété d'objet a le même nom qu'un attribut d'entité mais que leurs types ne sont pas compatibles, l'attribut de l'entité ne reçoit pas de valeur.

Mode création ou modification

Pour chaque objet de *objectCol* :

- Si l'objet contient une propriété booléenne "__NEW" fixée à faux (ou ne contient pas de propriété booléenne "__NEW"), l'entité est modifiée ou créée avec les valeurs correspondantes des propriétés de l'objet. Aucune vérification spécifique n'est effectuée concernant la clé primaire :
 - Si la clé primaire est fournie et existe, l'entité est modifiée. Dans ce cas, la clé primaire peut être fournie telle quelle ou via la propriété "__KEY" (contenant la valeur de la propriété primaire).
 - Si la clé primaire est fournie (telle quelle) et n'existe pas, l'entité est créée
 - Si la clé primaire n'est pas fournie, l'entité est créée et la clé primaire est assignée selon les règles en vigueur de la base de données.
- si l'objet contient une propriété booléenne "__NEW" fixée à vrai, l'entité est créée avec les valeurs correspondantes des propriétés de l'objet. Une vérification est effectuée sur la clé primaire :
 - Si la clé primaire est fournie (telle quelle) et existe, une erreur est générée

- o Si la clé primaire est fournie (telle quelle) et n'existe pas, l'entité est créée
- o Si la clé primaire n'est pas fournie, l'entité est créée et la clé primaire est assignée selon les règles en vigueur de la base de données.

La propriété "__KEY" n'est prise en compte que lorsque la propriété "__NEW" est fixée à faux (ou est omise) et qu'une entité correspondante existe. Dans tous les autres cas, la valeur de la propriété "__KEY" est ignorée, la clé primaire doit être passée "telle quelle".

Related entities

Les objets de *objectCol* peuvent contenir un ou plusieurs objet(s) imbriqué(s) décrivant une ou plusieurs entité(s) relative(s), ce qui peut être utile pour créer ou modifier des relations entre les entités.

Les objets imbriqués décrivant les entités relatives doivent contenir une propriété "__KEY" (contenant la valeur de la clé primaire de l'entité relative) ou la clé primaire de l'entité relative elle-même. L'utilisation de la propriété __KEY permet de ne pas dépendre du nom de l'attribut clé primaire.

Ce mécanisme ne permet pas de créer ou de modifier les entités liées.

Stamp

Si une propriété __STAMP est fournie, une vérification est effectuée sur le stamp (marqueur interne) de l'entité dans le datastore et une erreur est retournée en cas d'invalidité ("Le stamp ne correspond pas à celui de l'enregistrement# XX de la table XXXX"). Pour plus d'informations, voir [Verrouillage d'une entité](#).

settings

Dans le paramètre optionnel *settings*, vous pouvez passer un objet contenant des options supplémentaires. La propriété suivante est prise en charge :

Propriété	Type	Description
context	Text	Nom du contexte d'optimisation appliqué à l'entity selection. Ce contexte sera utilisé par le code qui manipule l'entity selection afin de bénéficier de l'optimisation. Cette fonctionnalité est conçue pour le traitement client/serveur ORDA .

Exemple 1

Nous souhaitons modifier une entité existante. La propriété __NEW n'est pas fixée, la clé primaire de l'employé est passée et existe :

```
var $empsCollection : Collection
var $emp : Object
var $employees : cs.EmployeeSelection

$empsCollection:=New collection
$emp:=New object
$emp.ID:=668 //Clé primaire existante dans la dataclass Employee
$emp.firstName:="Arthur"
$emp.lastName:="Martin"
$emp.employer:=New object("ID";121) //Clé primaire existante dans la dataclass liée Company
// Nous modifions la Company de cet employé en lui assignant une autre clé primaire existante dans la
$empsCollection.push($emp)
$employees:=ds.Employee.fromCollection($empsCollection)
```

Exemple 2

Nous souhaitons modifier une entité existante. La propriété __NEW n'est pas fournie, la clé primaire de l'employé est avec l'attribut __KEY et existe :

```

var $empsCollection : Collection
var $emp : Object
var $employees : cs.EmployeeSelection

$empsCollection:=New collection
$emp:=New object
$emp.__KEY:=1720 //Clé primaire existante dans la dataclass Employee
$emp.firstName:="John"
$emp.lastName:="Boorman"
$emp.employer:=New object("ID";121) //Clé primaire existante dans la dataclass liée Company
// Nous modifions la Company de cet employé en lui assignant une autre clé primaire existante dans la
$empsCollection.push($emp)
$employees:=ds.Employee.fromCollection($empsCollection)

```

Exemple 3

Création simple d'une entité à partir d'une collection :

```

var $empsCollection : Collection
var $emp : Object
var $employees : cs.EmployeeSelection

$empsCollection:=New collection
$emp:=New object
$emp.firstName:="Victor"
$emp.lastName:="Hugo"
$empsCollection.push($emp)
$employees:=ds.Employee.fromCollection($empsCollection)

```

Exemple 4

Nous souhaitons créer une entité. La propriété __NEW est à Vrai, la clé primaire de l'employé n'est pas fournie :

```

var $empsCollection : Collection
var $emp : Object
var $employees : cs.EmployeeSelection

$empsCollection:=New collection
$emp:=New object
$emp.firstName:="Mary"
$emp.lastName:="Smith"
$emp.employer:=New object("__KEY";121) //Clé primaire existante dans la dataclass liée Company
$emp.__NEW:=True
$empsCollection.push($emp)
$employees:=ds.Employee.fromCollection($empsCollection)

```

Exemple 5

Nous souhaitons créer une entité. La propriété __NEW n'est pas passée, la clé primaire de l'employé est fournie et n'existe pas :

```

var $empsCollection : Collection
var $emp : Object
var $employees : cs.EmployeeSelection

$empsCollection:=New collection
$emp:=New object
$emp.ID:=10000 //clé primaire qui n'existe pas
$emp.firstName:="Françoise"
$emp.lastName:="Sagan"
$empsCollection.push($emp)
$employees:=ds.Employee.fromCollection($empsCollection)

```

Exemple 6

Dans cet exemple, la première entité sera bien créée mais la seconde création échouera car les deux entités utilisent la même clé primaire :

```

var $empsCollection : Collection
var $emp; $emp2 : Object
var $employees : cs.EmployeeSelection

$empsCollection:=New collection
$emp:=New object
$emp.ID:=10001 // Clé primaire inexistante
$emp.firstName:="Simone"
$emp.lastName:="Martin"
$emp.__NEW:=True
$empsCollection.push($emp)

$emp2:=New object
$emp2.ID:=10001 // ERREUR clé primaire identique
$emp2.firstName:="Marc"
$emp2.lastName:="Smith"
$emp2.__NEW:=True
$empsCollection.push($emp2)
$employees:=ds.Employee.fromCollection($empsCollection)
//la première entité est créée
//erreur clé dupliquée pour la seconde

```

Voir aussi

[.toCollection\(\)](#)

.get()

► Historique

[.get\(primaryKey : Integer { ; settings : Object } \) : 4D.Entity](#)

[.get\(primaryKey : Text { ; settings : Object } \) : 4D.Entity](#)

Paramètres	Type		Description
primaryKey	Integer OR Text	->	Valeur de la clé primaire de l'entité à récupérer
settings	Object	->	Option de création : contexte
Résultat	4D.Entity	<-	Entité correspondant à la clé primaire indiquée

Description

La fonction `.get()` interroge la dataclass pour récupérer l'entité correspondant au paramètre `primaryKey`.

Dans `primaryKey`, passez la valeur de clé primaire de l'entité à récupérer. Le type de valeur doit correspondre au type de clé primaire définie dans le datastore (entier long ou texte). Vous pouvez également vous assurer que la valeur de la clé primaire est toujours renvoyée en tant que texte en utilisant la fonction `.getKey()` avec le paramètre `dk key as string`.

Si aucune entité avec `primaryKey` n'est trouvée, une entité Null est retournée.

Le chargement différé (lazy loading) est appliqué, ce qui signifie que les données associées sont chargées à partir du disque uniquement lorsque cela est nécessaire.

settings

Dans le paramètre optionnel `settings`, vous pouvez passer un objet contenant des options supplémentaires. La propriété suivante est prise en charge :

Propriété	Type	Description
context	Text	Nom du contexte d'optimisation appliquée à l'entité. Ce contexte sera utilisé par le code qui manipule l'entité afin de bénéficier de l'optimisation. Cette fonctionnalité est conçue pour le traitement client/serveur ORDA .

Exemple 1

```
var $entity : cs.EmployeeEntity
var $entity2 : cs.InvoiceEntity
$entity:=ds.Employee.get(167) // retourne l'entité dont la valeur de clé primaire est 167
$entity2:=ds.Invoice.get("DGGX20030") // retourne l'entité dont la valeur de clé primaire est "DGGX2003
```

Exemple 2

Cet exemple illustre l'utilisation de la propriété `context` :

```
var $e1; $e2; $e3; $e4 : cs.EmployeeEntity
var $settings; $settings2 : Object

$settings:=New object("context";"detail")
$settings2:=New object("context";"summary")

$e1:=ds.Employee.get(1;$settings)
completeAllData($e1) // Dans la méthode completeAllData, une optimisation est lancée et associée au con

$e2:=ds.Employee.get(2;$settings)
completeAllData($e2) // Dans la méthode completeAllData, l'optimisation associée au contexte "detail" e

$e3:=ds.Employee.get(3;$settings2)
completeSummary($e3) //Dans la méthode completeSummary, une optimisation est lancée et associée au cont

$e4:=ds.Employee.get(4;$settings2)
completeSummary($e4) //Dans la méthode completeSummary, l'optimisation associée au contexte "summary" e
```

.getCount()

► Historique

`.getCount() : Integer`

Paramètres	Type		Description
result	Integer	<-	Nombre d'entités dans la dataclass

Description

La fonction `.getCount()` retourne le nombre d'entités dans une dataclass.

Si cette fonction est utilisée dans une transaction, les entités créées durant la transaction sont prises en compte.

Exemple

```
var $ds : 4D.DataStoreImplementation
var $number : Integer

$ds:=Open datastore(New object("hostname"; "www.myserver.com"); "myDS")

$number:=$ds.Persons.getCount()
```

.getDataStore()

► Historique

`.getDataStore()` : cs.DataStore

Paramètres	Type		Description
Résultat	cs.DataStore	<-	Datastore de la dataclass

Description

La fonction `.getDataStore()` retourne le datastore de la dataclass.

Le datastore peut être :

- le datastore principal, retourné par la commande `ds`.
- un datastore distant, ouvert à l'aide de la commande `Open datastore`.

Exemple

La méthode projet `SearchDuplicate` recherche des valeurs dupliquées dans une dataclass.

```
var $pet : cs.CatsEntity
$pet:=ds.Cats.all().first() //lire une entité
SearchDuplicate($pet;"Dogs")

// SearchDuplicate method
// SearchDuplicate(entity_to_search;dataclass_name)

#DECLARE ($pet : Object ; $dataClassName : Text)
var $dataStore; $duplicates : Object

$dataStore:=$pet.getDataClass().getDataStore()
$duplicates:=$dataStore[$dataClassName].query("name=:1";$pet.name)
```

. getInfo()

► Historique

`.getInfo()` : Object

Paramètres	Type		Description
Résultat	Object	<-	Informations sur la dataclass

Description

La fonction `.getInfo()` retourne un objet qui fournit des informations sur la dataclass. Cette fonction est utile pour l'écriture de code générique.

Objet retourné

Propriété	Type	Description
exposed	Booléen	Vrai si la dataclass est exposée en REST
name	Text	Nom de la dataclass
primaryKey	Text	Nom de la clé primaire de la dataclass
tableNumber	Integer	Numéro interne de la table 4D

Exemple 1

```
#DECLARE ($entity : Object)
var $status : Object

computeEmployeeNumber($entity) //Exécuter des actions sur une entité

$status:=$entity.save()
if($status.success)
    ALERT("Enregistrement mis à jour dans la table "+$entity.getDataClass().getInfo().name)
End if
```

Exemple 2

```
var $settings : Object
var $es : cs.ClientsSelection

$settings:=New object
$settings.parameters:=New object("receivedIds";getIds())
$settings.attributes:=New object("pk";ds.Clients.getInfo().primaryKey)
$es:=ds.Clients.query(":pk in :receivedIds";$settings)
```

Exemple 3

```
var $pk : Text
var $dataClassAttribute : Object

$pk:=ds.Employee.getInfo().primaryKey
$dataClassAttribute:=ds.Employee[$pk] // Le cas échéant, l'attribut correspondant à la clé primaire est
```

Voir aussi

[DataClassAttribute.exposed](#)

[.getRemoteCache\(\)](#)

► Historique

.getRemoteCache() : Object

Paramètres	Type		Description
result	Object	<-	Objet décrivant le contenu du cache ORDA pour la dataclass.

Mode avancé : Cette fonction est destinée aux développeurs qui souhaitent personnaliser les fonctionnalités par défaut de ORDA dans le cadre de configurations spécifiques. Dans la plupart des cas, vous n'aurez pas besoin de l'utiliser.

Description

La fonction `.getRemoteCache()` retourne un objet qui contient le cache ORDA pour la dataclass.

Si elle est appelée depuis une application 4D monoposte, la fonction retourne `Null`.

L'objet retourné contient les propriétés suivantes :

Propriété	Type	Description
maxEntries	Integer	Nombre maximum de collections "entries".
stamp	Integer	Marqueur du cache.
timeout	Integer	Durée avant qu'une nouvelle entrée dans le cache soit indiquée comme expirée.
entries	Collection	Contient un objet pour chaque entité dans le cache.

Chaque objet "entrée" de la collection `entries` contient les propriétés suivantes :

Propriété	Type	Description
data	Object	Objet contenant les données de l'entrée.
expired	Booléen	True si l'entrée a expiré.
key	Text	Clé primaire de l'entité.

L'objet `data` de chaque entrée contient les propriétés suivantes :

Propriété	Type	Description
<code>__KEY</code>	Chaine	Clé primaire de l'entité
<code>__STAMP</code>	Longint	Stamp de l'entité dans la base de données
<code>__TIMESTAMP</code>	Chaine	Timestamp de l'entité dans la base de données (le format est YYYY-MM-DDTHH:MM:SS:ms:Z)
<code>dataClassNameAttributeName</code>	Variant	S'il y a des données dans le cache pour un attribut de dataclass, il est retourné dans une propriété du même type que dans la base de données.

Les données concernant les entités liées sont stockées dans le cache de l'objet `data`.

Exemple

Dans l'exemple suivant, `$ds.Persons.all()` charge la première entité avec tous ses attributs. Ensuite, l'optimiseur de requêtes entre en jeu, et seuls `firstname` et `address.city` sont chargés.

Notez que `address.city` est chargé dans le cache de la dataclass `Persons`.

Seule la première entité de la dataclass `Address` est stockée dans le cache. Elle est chargée durant la première itération de la boucle.

```

var $ds : 4D.DataStoreImplementation
var $persons : cs.PersonsSelection
var $p : cs.PersonsEntity
var $cachePersons; $cacheAddress : Object
var $text : Text

$ds:=Open datastore(New object("hostname"; "www.myserver.com"); "myDS")

$persons:=$ds.Persons.all()

$text:=""
For each ($p; $persons)
    $text:=$p.firstname+" lives in "+$p.address.city+ " / "
End for each

$cachePersons:=$ds.Persons.getRemoteCache()
$cacheAddress:=$ds.Adress.getRemoteCache()

```

Voir aussi

[.setRemoteCacheSettings\(\)](#)
[.clearRemoteCache\(\)](#)

.new()

► Historique

.new() : 4D.Entity

Paramètres	Type		Description
Résultat	4D.Entity	<-	Nouvelle entité correspondant à la dataclass

Description

La fonction `.new()` crée en mémoire et renvoie une nouvelle entité vide pour la dataclass.

L'objet entité est créé en mémoire et n'est pas sauvegardé dans la base de données tant que la fonction `.save()` n'est pas appelée. Si l'entité est supprimée avant d'être sauvegardée, elle ne peut pas être récupérée.

4D Server: En client-serveur, si la clé primaire de la table correspondante est auto-incrémentée, elle sera calculée au moment de la sauvegarde de l'entité sur le serveur.

Tous les attributs de l'entité sont initialisés avec la valeur `null`.

Les attributs peuvent être initialisés à des valeurs par défaut si l'option Traduire les NULL en valeurs vides est sélectionnée au niveau de la structure de la base 4D.

Exemple

Cet exemple crée une nouvelle entité dans la dataclass "Log" et enregistre les informations dans l'attribut "info" :

```

var $entity : cs.LogEntity
$entity:=ds.Log.new() //crée une référence
$entity.info:="New entry" //valorise l'attribut info
$entity.save() //sauvegarde l'entité

```

.newSelection()

► Historique

`.newSelection({ keepOrder : Integer }) : 4D.EntitySelection`

Paramètres	Type		Description
keepOrder	Integer	->	<code>dk keep ordered</code> : crée une entity selection triée, <code>dk non ordered</code> : crée une entity selection non triée (défaut si omis)
Résultat	4D.EntitySelection	<-	Nouvelle entity selection vide liée à la dataclass

Description

La fonction `.newSelection()` crée en mémoire une entity selection vide, non partageable, liée à la dataclass.

Pour plus d'informations sur les sélections d'entités non partageables, veuillez vous reporter à [cette section](#).

Si vous voulez créer une entity selection triée, passez le sélecteur `dk keep ordered` dans le paramètre `keepOrder`. Par défaut, si vous omettez ce paramètre ou si vous passez le sélecteur `dk non ordered` la fonction crée une entity selection non triée. Les entity selections non triées sont plus rapides, mais vous ne pouvez pas vous fier aux positions des entités. Pour plus d'informations, voir [Entity selections triées vs Entity selections non-triées](#).

Une fois créée, l'entity selection ne contient aucune entité (`mySelection.length` retourne 0). Cette fonction vous permet de construire progressivement des entity selections en effectuant des appels à la fonction `add()`.

Exemple

```
var $USelection; $OSelection : cs.EmployeeSelection
$USelection:=ds.Employee.newSelection() //crée une entity selection non triée vide
$OSelection:=ds.Employee.newSelection(dk keep ordered) //crée une entity selection triée vide
```

.query()

► Historique

`.query(queryString : Text { ; ...value : any } { ; querySettings : Object }) : 4D.EntitySelection`

`.query(formula : Object { ; querySettings : Object }) : 4D.EntitySelection`

Paramètres	Type		Description
queryString	Text	->	Critères de recherche en texte
formula	Object	->	Critères de recherche en objet formule
value	any	->	Valeur(s) à utiliser comme placeholder(s)
querySettings	Object	->	Options de recherche : parameters, attributes, args, allowFormulas, context, queryPath, queryPlan
Résultat	4D.EntitySelection	<-	Nouvelle entity selection constituée des entités de la dataclass correspondant au(x) critère(s) de recherche fournis dans <code>queryString</code> ou <code>formula</code>

Description

La fonction `.query()` recherche les entités répondant aux critères de recherche spécifiés dans `queryString` ou `formula` et (optionnellement) dans `value` toutes les entités de la dataclass, et retourne un nouvel objet de type `EntitySelection` contenant toutes les entités trouvées. Le mode lazy loading est appliqué.

Si aucune entité correspondante n'est trouvée, une `EntitySelection` vide est retournée.

Paramètre queryString

Le paramètre `queryString` doit respecter la syntaxe suivante :

```
attributePath|formula comparator value  
{logicalOperator attributePath|formula comparator value}  
{order by attributePath {desc | asc}}
```

où :

- **attributePath** : Chemin de l'attribut sur lequel vous souhaitez exécuter la recherche. Ce paramètre peut contenir un nom simple (par exemple "pays") ou un chemin d'attribut valide (par exemple "pays.nom"). Dans le cas d'un chemin d'attribut de type `Collection`, la notation [] est utilisée pour designer toutes les occurrences (par exemple "enfants[].age").

*Vous ne pouvez pas utiliser directement des attributs dont les noms contiennent des caractères spéciaux tels que ".", "[]", ou "=", ">", "#"... , car ils ne seront pas correctement évalués dans la chaîne de recherche. Si vous souhaitez rechercher ces attributs, vous devez utiliser des placeholders, qui permettent d'utiliser un ensemble de caractères plus étendu dans les chemins d'attribut (voir * Utiliser des placeholders ci-dessous).

- **formula** : Une formule valide passée en `Text` ou en `Object`. La formule sera évaluée pour chaque entité traitée et doit retourner une valeur booléenne. Dans la formule, l'entité est disponible via l'objet `This`.
 - `Text` : la chaîne de formule doit être précédée de la déclaration `eval()`, afin que l'analyseur de requêtes évalue l'expression correctement. Par exemple : `"eval(length(This.lastname) >=30)"`
 - `Object`: l'**objet formule** est passé en tant que placeholder (voir ci-dessous). La formule doit avoir été créée à l'aide des commandes `Formula` ou `Formula from string`.

- N'oubliez pas que les formules de 4D prennent uniquement en charge les symboles `&` et `|` comme opérateurs logiques.
- Si la formule n'est pas le seul critère de recherche, le système d'optimisation des requêtes pourra prioriser le traitement d'autres critères (ex : attributs indexés) et ainsi, la formule sera évaluée uniquement pour un sous-ensemble d'entités.

Les formules contenues dans les requêtes peuvent recevoir des paramètres via `$1`. Ce point est détaillé dans la section Paramètres supplémentaires.

- Vous pouvez également passer directement un paramètre `formula` au lieu d'un paramètre `queryString` (recommandé lorsque les formules sont plus complexes). Voir le paragraphe Paramètre formula ci-dessous.
 - Pour des raisons de sécurité, les appels de formule dans les fonctions `query()` peuvent être interdits. Voir la description du paramètre `querySettings`.
- **comparator** : symbole d'opérateur utilisé pour comparer `attributePath` et `value`. Les symboles suivants sont pris en charge :

Comparaison	Symbol(s)	Commentaire
Egal à	=, ==	Retourne les données correspondantes, prend en charge le joker de recherche (@), ne tient pas compte de la casse et est non diacritique.
	==:, IS	Retourne les données correspondantes, considère le @ comme un caractère standard, ne tient pas compte de la casse et est non diacritique
Différent de	#, !=	Prend en charge le joker de recherche (@)
	!=:, IS NOT	Considère le @ comme un caractère standard
Inférieur à	<	
Supérieur à	>	
Inférieur ou égal à	<=	
Supérieur ou égal à	>=	
Inclus parmi	IN	Retourne les données égales à au moins une des valeurs d'une collection ou d'un ensemble de valeurs, prend en charge le joker de recherche (@)
Condition Not appliquée à une assertion	NOT	Les parenthèses sont obligatoires lorsque NOT est utilisé devant une assertion contenant plusieurs opérateurs
Contient mot-clé	%	Les mots-clés peuvent être utilisés avec les attributs de type texte ou image

- value : valeur à comparer à la valeur courante de l'attribut de chaque entité de la sélection ou élément de la collection. Peut être un placeholder (voir Utiliser des placeholders ci-dessous) ou toute expression correspondant à la propriété du type de donnée.

Lorsque vous utilisez une valeur constante, les règles suivantes doivent être respectées :

- Les valeurs constantes de type texte peuvent être passées avec ou sans guillemets (voir Utilisation des guillemets ci-dessous). Pour rechercher une chaîne dans une chaîne (recherche de type "contient"), utilisez le symbole joker (@) dans valeur pour isoler la chaîne à chercher, comme dans cet exemple : "@Smith@". Les mots-clés suivants sont interdits pour des constantes de type texte : true, false.
 - Valeurs constantes de typebooléen: true or false (sensible à la casse).
 - Valeurs constantes de type numérique : les décimales doivent être séparées par un !'
 - Constantes de type date : "YYYY-MM-DD" format
 - Constantes null : en utilisant le mot-clé "null", la recherche trouvera les propriétés ayant la valeur null et undefined.
 - Dans le cas d'une recherche avec un comparateur IN, value doit être une collection, ou des valeurs du même type que les données du chemin d'attribut, fournies entre [] et séparées par des virgules (pour les chaînes, les caractères " " doivent être échappés avec des \).
- logicalOperator : utilisé pour relier des conditions multiples dans la recherche (optionnel). Vous pouvez utiliser un des opérateurs logiques suivants (le nom ou le symbole peut être passé) :

Conjonction	Symbol(s)
AND	&, &&, and
OR	, , or

- order by attributePath : vous pouvez inclure une déclaration order by attributePath dans la requête afin que les données résultantes soient triées selon cette déclaration. Vous pouvez utiliser plusieurs tris par déclaration, en les séparant par des virgules (e.g., order by attributePath1 desc, attributePath2 asc). Par défaut, le tri est par ordre croissant. Passez 'desc' pour définir un tri par ordre décroissant et 'asc' pour définir un tri par ordre croissant.

Si vous utilisez cette déclaration, l'entity selection retournée est triée (pour plus d'informations, veuillez consulter Entity selections triées vs Entity selection non triées).

Utilisation des guillemets

Lorsque vous utilisez des guillemets dans les recherches, vous devez utiliser des apostrophes ' ' à l'intérieur des requêtes et des guillemets " " pour encadrer la recherche complète, sinon une erreur est générée. Par exemple :

```
"employee.name = 'smith' AND employee.firstname = 'john'"
```

Les guillemets simples ('') ne sont pas pris en charge dans les valeurs recherchées car ils casseraient la chaîne de recherche. Par exemple, "comp.name = 'John's pizza' " générera une erreur. Si vous devez rechercher des valeurs contenant des guillemets simples, il est nécessaire d'utiliser des placeholders (voir ci-dessous).

Utilisation des parenthèses

Vous pouvez utiliser des parenthèses dans la recherche afin de prioriser les calculs. Par exemple, vous pouvez organiser une recherche de la manière suivante :

```
"(employee.age >= 30 OR employee.age <= 65) AND (employee.salary <= 10000 OR employee.status = 'Manager'"
```

Utilisation des placeholders

4D vous permet d'utiliser des placeholders pour les arguments *attributePath*, *formula* et *value* dans le paramètre *queryString*. Un placeholder est un paramètre que vous insérez dans des chaînes de recherche et qui est remplacé par une autre valeur au moment où la chaîne de recherche est évaluée. La valeur des placeholders est évaluée une seule fois, au début de la requête ; elle n'est pas évaluée pour chaque élément.

Il existe deux types de placeholders : les placeholders indexés et les placeholders nommés :

	Placeholders indexés	Placeholders nommés
Définition	Vous pouvez combiner tous les types d'arguments dans <i>queryString</i> . Une <i>queryString</i> peut contenir, pour les paramètres <i>attributePath</i> , <i>formula</i> et <i>value</i> :	Les paramètres sont insérés sous la forme <code>:paramName</code> (par exemple <code>:myparam</code>) et leurs valeurs sont fournies dans les objets attributs et/ou <code>parameters</code> dans le paramètre <i>querySettings</i>
Exemple	<code>\$r:=class.query(":1=:2";"city";"Chicago")</code>	<code>\$o.attributes:=New object("att";"city")</code> <code>\$o.parameters:=New object("name";"Chicago")</code> <code>\$r:=class.query(":att=:name";\$o)</code>

Vous pouvez combiner tous les types d'arguments dans *queryString*. Une *queryString* peut contenir, pour les paramètres *attributePath*, *formula* et *value* :

- des valeurs directes (pas de placeholders)
- des placeholders indexés et/ou nommés.

L'utilisation de placeholders dans les recherches est recommandée pour les raisons suivantes :

1. Cela empêche l'injection de code malveillant : si vous utilisez dans la chaîne de recherche des variables dont le contenu provient directement de la saisie de l'utilisateur, celui-ci pourrait modifier les conditions de recherche en saisissant des arguments de recherche supplémentaires. Par exemple, imaginez une chaîne de recherche du type :

```
$vquery:="status = 'public' & name = "+myname //l'utilisateur saisit son nom  
$result:=$col.query($vquery)
```

Cette recherche semble sécurisée puisque les données non publiques sont filtrées. Cependant, si l'utilisateur saisit dans la zone *myname* une chaîne du type *'smith OR status='private'*, la chaîne de recherche sera modifiée à l'étape de l'interprétation et pourra retourner des données privées.

Lorsque vous utilisez des placeholders, le contournement des options de sécurité n'est pas possible :

```
$result:=$col.query("status='public' & name=:1";myname)
```

Dans ce cas, si l'utilisateur saisit *smith OR status='private'* dans la zone *myname*, cela ne sera pas interprété dans la chaîne de recherche, mais uniquement passé en tant que valeur. La recherche d'une personne nommée "smith OR status='private'" échouera simplement.

2. Cela résout les questions liées au formatage des valeurs ou des caractères, notamment lorsque vous gérez des paramètres *attributePath* et *value* qui peuvent contenir des caractères non-alphanumériques tels que ".", "[...]", "[!..."
3. Cela permet l'utilisation de variables ou d'expressions dans les arguments de recherche. Voici quelques exemples :

```
$result:=$col.query("address.city = :1 & name =:2;$city;$myVar+@")
$result2:=$col.query("company.name = :1;"John's Pizzas")
```

Recherche de valeurs null

Lorsque vous recherchez les valeurs null, vous ne pouvez pas utiliser la syntaxe placeholder car le moteur de recherche considère la valeur null comme une valeur de comparaison invalide. Par exemple, si vous exécutez la recherche suivante :

```
$vSingles:=ds.Person.query("spouse = :1;Null) // ne fonctionnera PAS
```

Vous n'obtiendrez pas le résultat souhaité car la valeur null sera évaluée par 4D comme une erreur résultant de l'évaluation du paramètre (pouvant être, par exemple, un attribut provenant d'une autre recherche). Pour ce type de recherches, vous devez utiliser la syntaxe de recherche directe :

```
$vSingles:=ds.Person.query("spouse = null") //syntaxe valide
```

Relier arguments de recherche et attributs de collection

Lorsque vous effectuez une recherche dans des collections au sein d'attributs objet à l'aide de plusieurs arguments de requête reliés par l'opérateur AND, vous souhaiterez éventuellement vous assurer que seules les entités contenant des éléments correspondant à tous les arguments soient retournées, et non les entités où des arguments peuvent être trouvés dans différents éléments. Pour ce faire, vous devez relier les arguments de requête aux éléments de collection, de sorte que seuls les éléments uniques contenant des arguments reliés soient retournés.

Par exemple, avec les deux entités suivantes :

```
Entity 1:
ds.People.name: "martin"
ds.People.places:
{ "locations" : [ {
    "kind":"home",
    "city":"paris"
} ] }

Entity 2:
ds.People.name: "smith"
ds.People.places:
{ "locations" : [ {
    "kind":"home",
    "city":"lyon"
} , {
    "kind":"office",
    "city":"paris"
} ] }
```

Vous souhaitez trouver des personnes dont le type d'emplacement est "home" dans la ville de "paris". Si vous écrivez :

```
ds.People.query("places.locations[].kind= :1 and places.locations[].city= :2";"home";"paris")
```

... la requête retournera "martin" et "smith" car "smith" a un élément "locations" dont "kind" (le type) est "home" et un élément "locations" dont "city" (la ville) est "paris", même si ce sont des éléments différents.

Si vous souhaitez obtenir uniquement les entités dont les arguments correspondants sont dans le même élément de collection, vous devez relier les arguments. Pour lier des arguments de requête :

- Ajoutez une lettre entre le caractère [] dans le premier chemin à lier et répétez la même lettre dans tous les arguments liés. Par exemple : `locations[a].city and locations[a].kind`. Vous pouvez utiliser n'importe quelle lettre de l'alphabet latin (non sensible à la casse).
- Pour ajouter différents critères liés dans la même requête, utilisez une autre lettre. Vous pouvez créer jusqu'à 26 combinaisons de critères dans une seule requête.

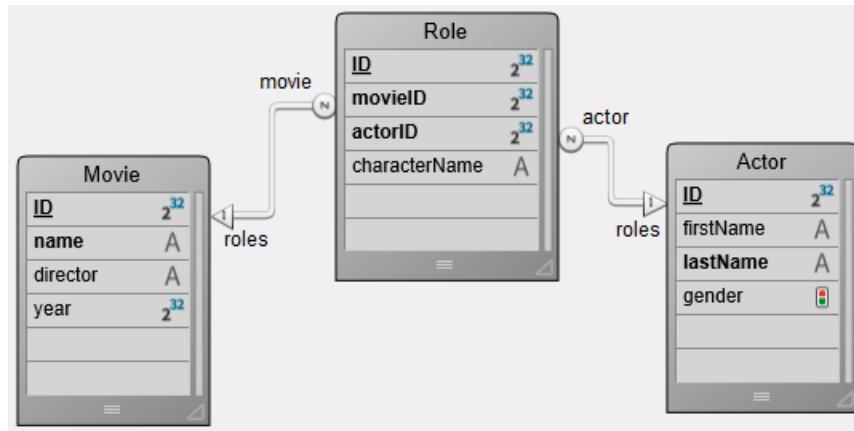
A l'aide des entités ci-dessus, si vous écrivez :

```
ds.People.query("places.locations[a].kind= :1 and places.locations[a].city= :2";"home";"paris")
```

... la requête ne retournera uniquement "martin" car il possède un élément "locations" dont "kind" est "home" et dont "city" est "paris". La requête ne retournera pas "smith" car les valeurs "home" et "paris" ne sont pas contenus dans le même élément de collection.

Recherches dans les relations N vers N

ORDA propose une syntaxe spéciale pour faciliter les recherches dans les relations N vers N. Dans ce contexte, vous pouvez avoir besoin de rechercher des valeurs différentes à l'aide de l'opérateur AND MAIS dans le même attribut. Par exemple, considérez la structure suivante :



Imaginez que vous souhaitez rechercher tous les films dans lesquels un acteur A Et un acteur B ont joué un rôle. Si vous écrivez une recherche simple utilisant l'opérateur AND , cela ne va pas fonctionner :

```
// code invalide
$es:=ds.Movie.query("roles.actor.lastName = :1 AND roles.actor.lastName = :2";"Hanks";"Ryan")
// $es est vide
```

Fondamentalement, le problème vient de la logique interne de la recherche : vous ne pouvez pas rechercher une entité dont un attribut aurait simultanément la valeur "A" et "B".

Pour rendre possible ce type de recherche, ORDA prend en charge une syntaxe spéciale : vous devez juste ajouter un *class index* entre les caractères {} dans tous les attributs relationnels utilisés dans la chaîne de recherche :

```
"relationAttribute.attribute = :1 AND relationAttribute{x}.attribute = :2 [AND relationAttribute{y}.attr
```

{x} indique à ORDA de créer une autre référence pour l'attribut relationnel. Le moteur de requêtes effectuera alors toutes les opérations internes nécessaires. Notez que x peut être n'importe quel nombre à l'exception de 0: {1}, ou {2}, ou {1540}... ORDA a simplement besoin d'une référence unique dans la recherche pour chaque class index.

Dans notre exemple, cela pourrait donner :

```
// code valide
$es:=ds.Movie.query("roles.actor.lastName = :1 AND roles.actor{2}.lastName = :2";"Hanks";"Ryan")
// $es contient des films (You've Got Mail, Sleepless in Seattle, Joe Versus the Volcano)
```

Paramètre formula

Au lieu d'insérer une formule dans le paramètre `queryString` (voir ci-dessus), vous pouvez directement passer un objet formule en tant que critère de recherche booléen. L'utilisation d'un objet formule pour les requêtes est recommandée car vous bénéficiez d'une tokenisation et le code est plus facile à rechercher/lire.

La formule doit avoir été créée à l'aide des commandes `Formula` ou `Formula from string`. Dans ce cas :

- `formula` est évaluée pour chaque entité et doit renvoyer vrai ou faux. Lors de l'exécution de la requête, si le résultat de la formule n'est pas un booléen, il est considéré comme faux.
- dans `formula`, l'entité est disponible via l'objet `This`.
- si l'objet `Formula` est null, l'erreur 1626 ("Attente d'un texte ou d'une formule") est générée, que vous pouvez intercepter à l'aide d'une méthode installée avec `ON ERR CALL`.

Pour des raisons de sécurité, les appels de formule dans les fonctions `query()` peuvent être interdits. Voir la description du paramètre `querySettings`.

Passer des paramètres aux formules

Tout paramètre `formula` appelé par la fonction `query()` peut recevoir des paramètres :

- Les paramètres doivent être passés via la propriété `args` du paramètre `querySettings`.
- La formule reçoit cet objet `args` en tant que paramètre `$1`.

Ce code montre comment les paramètres sont passés aux fonctions :

```
$settings:=New object("args";New object("exclude";"-")) //objet args pour passer des paramètres
$es:=ds.Students.query("eval(checkName($1.exclude))";$settings) //args est reçu dans $1
```

Des exemples supplémentaires sont fournis dans l'exemple 3.

4D Server : En client/serveur, les formules sont exécutées sur le serveur. Dans ce contexte, seul l'objet `querySettings.args` est envoyé aux formules.

Paramètre querySettings

Dans le paramètre `querySettings` vous pouvez passer un objet contenant des options supplémentaires. Les propriétés suivantes sont prises en charge :

Propriété	Type	Description						
parameters	Object	Placeholders nommés pour les valeurs utilisées dans <i>queryString</i> ou <i>formula</i> . Les valeurs sont exprimées sous forme de paires propriété / valeur, où propriété est le nom du placeholder inséré pour une valeur dans <i>queryString</i> ou <i>formula</i> ("placeholder") et où valeur correspond à la valeur à comparer. Vous pouvez combiner, dans une même recherche, des placeholders indexés (valeurs passées directement dans les paramètres <i>value</i>) et les valeurs des placeholders nommés.						
attributes	Object	Placeholders nommés pour les chemins d'attributs utilisés dans <i>queryString</i> ou <i>formula</i> . Les attributs sont exprimés sous forme de paires propriété / valeur, où propriété est le nom du placeholder inséré pour un chemin d'attribut dans <i>queryString</i> or <i>formula</i> ("placeholder"), et où valeur peut être une chaîne ou une collection de chaînes. Chaque valeur est un chemin qui peut désigner soit un attribut scalaire ou relatif de la dataclass soit une propriété d'un champ objet de la dataclass <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>Type</th><th>Description</th></tr> </thead> <tbody> <tr> <td>Chaine</td><td>attributePath exprimé à l'aide de la notation à point, ex : "name" ou "user.address.zipCode"</td></tr> <tr> <td>Collection de chaînes</td><td>Chaque chaîne de la collection représente un niveau d'attributePath, ex : ["name"] ou ["user", "address", "zipCode"]. L'utilisation d'une collection permet de rechercher des attributs dont les noms ne sont pas conformes à la notation à point, ex : ["4Dv17.1", "en/fr"]</td></tr> </tbody> </table>	Type	Description	Chaine	attributePath exprimé à l'aide de la notation à point, ex : "name" ou "user.address.zipCode"	Collection de chaînes	Chaque chaîne de la collection représente un niveau d'attributePath, ex : ["name"] ou ["user", "address", "zipCode"]. L'utilisation d'une collection permet de rechercher des attributs dont les noms ne sont pas conformes à la notation à point, ex : ["4Dv17.1", "en/fr"]
Type	Description							
Chaine	attributePath exprimé à l'aide de la notation à point, ex : "name" ou "user.address.zipCode"							
Collection de chaînes	Chaque chaîne de la collection représente un niveau d'attributePath, ex : ["name"] ou ["user", "address", "zipCode"]. L'utilisation d'une collection permet de rechercher des attributs dont les noms ne sont pas conformes à la notation à point, ex : ["4Dv17.1", "en/fr"]							
		Vous pouvez combiner les valeurs des placeholders indexés (valeurs passées directement dans les paramètres <i>value</i>) et les valeurs des placeholders nommés dans la même recherche.						
args	Object	Paramètre(s) à passer aux formules, le cas échéant. L'objet args sera reçu dans \$1 à l'intérieur des formules et donc ses valeurs seront disponibles via la propriété <i>\$1.property</i> (cf. exemple 3).						
allowFormulas	Booléen	Vrai pour autoriser les appels de formules dans la query (défaut). Passez faux pour interdire l'exécution de formules. Si la <code>query()</code> contient une formule alors que cette propriété est à Faux, une erreur est retournée (1278 - Formule non autorisée).						
context	Text	Nom du contexte d'optimisation automatique appliquée à l'entity selection. Ce contexte sera utilisé par le code qui manipule l'entity selection afin de bénéficier de l'optimisation. Cette fonctionnalité est conçue pour le traitement client/serveur ; pour plus d'informations, veuillez vous reporter à la section Optimisation client/serveur.						
queryPlan	Booléen	Dans l'entity selection résultante, retourne ou ne retourne la description détaillée de la recherche juste avant d'être exécutée, i.e. La propriété retournée est un objet qui inclut chaque recherche et sous-recherche programmée (dans le cas d'une recherche complexe). Cette option est utile durant la phase de développement d'une application. Elle est utilisée conjointement à <i>queryPath</i> . Par défaut, si elle est omise : faux. Note : Cette propriété est prise en charge uniquement par les fonctions <code>entitySelection.query()</code> et <code>dataClass.query()</code> .						
queryPath	Booléen	Dans l'entity selection résultante, retourne ou ne retourne pas la description détaillée de la recherche telle qu'elle est effectuée. La propriété retournée est un objet qui contient le chemin utilisé pour la recherche (généralement identique à celui de <i>queryPlan</i> , mais il peut être différent si le moteur parvient à optimiser la recherche), la durée du traitement et le nombre d'enregistrements trouvés. Cette option est utile durant la phase de développement d'une application. Par défaut, si elle est omise : faux. Note : Cette propriété est prise en charge uniquement par les fonctions <code>entitySelection.query()</code> et <code>dataClass.query()</code> .						

A propos de *queryPlan* et *queryPath*

Les informations enregistrées dans `queryPlan` et `queryPath` incluent le type de recherche (indexée ou séquentielle), chaque sous-recherche nécessaire, ainsi que les opérateurs de conjonction. Les chemins d'accès des requêtes contiennent également le nombre d'entités identifiées et la durée d'exécution de chaque critère de recherche. Les

chemins d'accès des requêtes contiennent également le nombre d'entités identifiées et la durée d'exécution de chaque critère de recherche. Généralement, la description du plan de requête et son chemin d'accès sont identiques mais ils peuvent différer car 4D peut intégrer des optimisations dynamiques lorsqu'une requête est exécutée, afin d'améliorer les performances. Par exemple, le moteur 4D peut convertir dynamiquement une requête indexée en requête séquentielle s'il estime que ce processus est plus rapide. Ce cas particulier peut se produire lorsque le nombre d'entités recherchées est faible.

Par exemple, si vous exécutez la recherche suivante :

```
$sel:=ds.Employee.query("salary < :1 and employer.name = :2 or employer.revenues > :3";\n50000;"Lima West Kilo";10000000;New object("queryPath";True;"queryPlan";True))
```

queryPlan :

```
{Or:[{And:[{item:[index : Employee.salary ] < 50000},  
           {item:Join on Table : Company : Employee.employerID = Company.ID,  
            subquery:[{item:[index : Company.name ] = Lima West Kilo}]}]},  
       {item:Join on Table : Company : Employee.employerID = Company.ID,  
        subquery:[{item:[index : Company.revenues ] > 10000000}]}]}
```

queryPath :

```
{steps:[{description:OR,time:63,recordsfounds:1388132,
    steps:[{description:AND,time:32,recordsfounds:131,
        steps:[{description:[index : Employee.salary ] < 50000,time:16,recordsfounds:728260},{description:Jo
    steps:[{steps:[{description:[index : Company.name ] = Lima West Kilo,time:0,recordsfounds:1}]}]}],{steps:[{steps:[{description:[index : Company.revenues ] > 10000000,time:0,recordsfounds:933}]}]}]}]}
```

Exemple 1

Cette section fournit divers exemples de recherches.

Recherche dans une chaîne :

```
$entitySelection:=ds.Customer.query("firstName = 'S@'")
```

Recherche avec une instruction NOT :

```
$entitySelection:=ds.Employee.query("not(firstName=Kim)")
```

Recherche avec des dates :

```
$entitySelection:=ds.Employee.query("birthDate > :1";"1970-01-01")
$entitySelection:=ds.Employee.query("birthDate <= :1";Current date-10950)
```

Recherche avec des placeholders indexés pour les valeurs :

```
$entitySelection:=ds.Customer.query("(firstName = :1 or firstName = :2) and (lastName = :3 or lastName = :4)", {1:'John', 2:'Doe', 3:'Smith', 4:'Doe'})
```

Recherche avec des placeholders indexés pour les valeurs sur une dataclass liée :

```
$entitySelection:=ds.Employee.query("lastName = :1 and manager.lastName = :2";"M@";"S@")
```

Recherche avec des placeholders indexés avec tri décroissant :

```
$entitySelection:=ds.Student.query("nationality = :1 order by campus.name desc, lastname";"French")
```

Recherche avec des placeholders nommés pour les valeurs :

```
var $querySettings : Object  
var $managedCustomers : cs.CustomerSelection  
$querySettings:=New object  
$querySettings.parameters:=New object("userId";1234;"extraInfo";New object("name";"Smith"))  
$managedCustomers:=ds.Customer.query("salesperson.userId = :userId and name = :extraInfo.name";$querySet
```

Recherche utilisant les placeholders nommés et indexés pour les valeurs :

```
var $querySettings : Object  
var $managedCustomers : cs.CustomerSelection  
$querySettings.parameters:=New object("userId";1234)  
$managedCustomers:=ds.Customer.query("salesperson.userId = :userId and name=:1";"Smith";$querySettings)
```

Recherche avec objets queryPlan et queryPath :

```
$entitySelection:=ds.Employee.query("(firstName = :1 or firstName = :2) and (lastName = :3 or lastName = :4);  
//vous pouvez ensuite obtenir ces propriétés dans la sélection d'entité résultante  
var $queryPlan; $queryPath : Object  
$queryPlan:=$entitySelection.queryPlan  
$queryPath:=$entitySelection.queryPath
```

Recherche avec un chemin d'attribut de type Collection :

```
$entitySelection:=ds.Employee.query("extraInfo.hobbies[].name = :1";"horsebackriding")
```

Recherche avec un chemin d'attribut de type Collection et des attributs reliés :

```
$entitySelection:=ds.Employee.query("extraInfo.hobbies[a].name = :1 and extraInfo.hobbies[a].level=:2";"horsebackriding";1)
```

Recherche avec un chemin d'attribut de type Collection et plusieurs attributs reliés :

```
$entitySelection:=ds.Employee.query("extraInfo.hobbies[a].name = :1 and extraInfo.hobbies[a].level = :2 and extraInfo.hobbies[b].name = :3 and extraInfo.hobbies[b].level = :4";"horsebackriding";2;"Tennis";5)
```

Recherche avec un chemin d'attribut de type Objet :

```
$entitySelection:=ds.Employee.query("extra.eyeColor = :1";"blue")
```

Recherche avec instruction IN :

```
$entitySelection:=ds.Employee.query("firstName in :1";New collection("Kim";"Dixie"))
```

Recherche avec instruction NOT (IN) :

```
$entitySelection:=ds.Employee.query("not (firstName in :1)";New collection("John";"Jane"))
```

Recherche avec des placeholders indexés pour les attributs :

```
var $es : cs.EmployeeSelection  
$es:=ds.Employee.query(":1 = 1234 and :2 = 'Smith'";"salesperson.userId";"name")  
//salesperson est une entité reliée
```

Recherche avec des placeholders indexés pour les attributs et avec des placeholders nommés pour les valeurs :

```
var $es : cs.EmployeeSelection  
var $querySettings : Object  
$querySettings:=New object  
$querySettings.parameters:=New object("customerName";"Smith")  
$es:=ds.Customer.query(":1 = 1234 and :2 = :customerName";"salesperson.userId";"name";$querySettings)  
//salesperson est une entité reliée
```

Recherche avec des placeholders indexés pour les attributs et les valeurs :

```
var $es : cs.EmployeeSelection  
$es:=ds.Clients.query(":1 = 1234 and :2 = :3";"salesperson.userId";"name";"Smith")  
//salesperson est une entité reliée
```

Exemple 2

Cette section illustre les recherches avec des placeholders nommés pour les attributs.

Considérons une dataclass Employee avec 2 entités :

Entité 1 :

```
name: "Marie"  
number: 46  
softwares:{  
"Word 10.2": "Installed",  
"Excel 11.3": "To be upgraded",  
"Powerpoint 12.4": "Not installed"  
}
```

Entité 2 :

```
name: "Sophie"  
number: 47  
softwares:{  
"Word 10.2": "Not installed",  
"Excel 11.3": "To be upgraded",  
"Powerpoint 12.4": "Not installed"  
}
```

Recherche avec des placeholders nommés pour les attributs :

```
var $querySettings : Object
var $es : cs.EmployeeSelection
$querySettings:=New object
$querySettings.attributes:=New object("attName";"name";"attWord";New collection("softwares";"Word 10.2")
$es:=ds.Employee.query(":attName = 'Marie' and :attWord = 'Installed'";$querySettings)
//$es.length=1 (Employee Marie)
```

Recherche avec des placeholders nommés pour les attributs et les valeurs :

```
var $querySettings : Object
var $es : cs.EmployeeSelection
var $name : Text
$querySettings:=New object
//Placeholders pour les valeurs
//Il est demandé à l'utilisateur de saisir un nom
$name:=Request("Veuillez saisir un nom à rechercher :")
If(OK=1)
    $querySettings.parameters:=New object("givenName";$name)
//Placeholders pour les chemins d'attributs
    $querySettings.attributes:=New object("attName";"name")
    $es:=ds.Employee.query(":attName= :givenName";$querySettings)
End if
```

Exemple 3

Ces exemples illustrent les diverses manières d'utiliser des formules avec ou sans paramètres dans vos recherches.

La formule est fournie sous forme de texte avec `eval()` dans le paramètre `queryString` :

```
var $es : cs.StudentsSelection
$es:=ds.Students.query("eval(length(This.lastname) >=30) and nationality='French'")
```

La formule est fournie sous forme d'objet `Formula` via un placeholder :

```
var $es : cs.StudentsSelection
var $formula : Object
$formula:=Formula(Length(This.lastname)>=30)
$es:=ds.Students.query(":1 and nationality='French"';$formula)
```

Seul un objet `Formula` est fourni comme critère de recherche :

```
var $es : cs.StudentsSelection
var $formula : Object
$formula:=Formula(Length(This.lastname)>=30)
$es:=ds.Students.query($formula)
```

Plusieurs formules peuvent être appliquées :

```
var $formula1; $1; $formula2 ;$0 : Object
$formula1:=$1
$formula2:=Formula(Length(This.firstname)>=30)
$0:=ds.Students.query(":1 and :2 and nationality='French'";$formula1;$formula2)
```

Une formule texte dans `queryString` reçoit un paramètre :

```
var $es : cs.StudentsSelection
var $settings : Object
$settings:=New object()
$settings.args:=New object("filter";"-")
$es:=ds.Students.query("eval(checkName($1.filter)) and nationality=:1";"French";$settings)
```

```
//méthode checkName
#DECLARE($exclude : Text) -> $result : Boolean
$result:=(Position($exclude;This.lastname)=0)
```

En utilisant la même méthode `checkName`, un objet `Formula` en placeholder reçoit un paramètre :

```
var $es : cs.StudentsSelection
var $settings; $formula : Object
$formula:=Formula(checkName($1.filter))
$settings:=New object()
$settings.args:=New object("filter";"-")
$es:=ds.Students.query(":1 and nationality=:2";$formula;"French";$settings)
$settings.args.filter:="*" // modifie les paramètres sans mettre à jour l'objet $formula
$es:=ds.Students.query(":1 and nationality=:2";$formula;"French";$settings)
```

Nous voulons interdire les formules, par exemple lorsque les utilisateurs saisissent leurs requêtes :

```
var $es : cs.StudentsSelection
var $settings : Object
var $queryString : Text
$queryString:=Request("Enter your query:")
if(OK=1)
    $settings:=New object("allowFormulas";False)
    $es:=ds.Students.query($queryString;$settings) //Une erreur est générée si $queryString contient une
End if
```

Voir aussi

[.query\(\)](#) pour les entity selections

.setRemoteCacheSettings()

► Historique

`.setRemoteCacheSettings(settings : Object)`

Paramètres	Type		Description
settings	Object	->	Objet définissant le timeout et la taille maximum du cache ORDA pour la dataclass.

Mode avancé : Cette fonction est destinée aux développeurs qui souhaitent personnaliser les fonctionnalités par défaut de ORDA dans le cadre de configurations spécifiques. Dans la plupart des cas, vous n'aurez pas besoin de l'utiliser.

Description

La fonction `.setRemoteCacheSettings()` définit le timeout et la taille maximum du cache ORDA pour la dataclass.

Dans le paramètre *settings*, passez un objet contenant les propriétés suivantes :

Propriété	Type	Description
timeout	Integer	Timeout en secondes.
maxEntries	Integer	Nombre maximum d'entités.

`timeout` définit le timeout du cache ORDA pour la dataclass (30 secondes par défaut). Lorsque le timeout est atteint, les entités de la dataclass dans le cache sont considérées comme expirées. Cela signifie que :

- les données sont toujours présentes
- la prochaine fois que les données seront requises, elles seront demandées au serveur
- 4D supprime automatiquement les données expirées lorsque le nombre maximum d'entités est atteint

Modifier la propriété `timeout` définit un nouveau timeout pour les entités déjà présentes dans le cache. Cela peut être utile lorsque vous travaillez avec des données qui ne changent pas souvent, et pour lesquelles de nouvelles requêtes au serveur ne sont donc pas nécessaires.

`maxEntries` définit le nombre maximum d'entités dans le cache ORDA. Par défaut ce nombre est 30 000.

Le nombre d'entrées minimum est 300, donc la valeur de `maxEntries` doit être égale ou supérieure à 300. Sinon la valeur est ignorée et le nombre d'entrées maximum est fixé à 300.

Si aucune propriété valide n'est passée à `timeout` et `maxEntries`, le cache n'est pas modifié, il conserve ses valeurs précédentes ou par défaut.

Lorsqu'une entité est sauvegardée, elle est mise à jour dans le cache expirera lorsque le timeout sera atteint.

Exemple

```
var $ds : 4D.DataStoreImplementation  
  
$ds:=Open datastore(New object("hostname"; "www.myserver.com"); "myDS")  
  
$ds.Buildings.setRemoteCacheSettings(New object("timeout"; 60; "maxEntries"; 350))
```

Voir aussi

[.clearRemoteCache\(\)](#)
[.getRemoteCache\(\)](#)

DataClassAttribute

Les attributs Dataclass sont disponibles en tant que propriétés de leurs classes respectives. Par exemple :

```
nameAttribute:=ds.Company.name //référence à un attribut de classe  
revenuesAttribute:=ds.Company["revenues"] //méthode alternative
```

Ce code attribue à *nameAttribute* et *revenuesAttribute* des références aux attributs *name* et *revenues* de la classe *Company*. Cette syntaxe ne retourne PAS les valeurs contenues dans l'attribut, mais retourne plutôt des références aux attributs eux-mêmes. Pour gérer les valeurs, vous devez passer par les [Entités](#).

Les objets `DataClassAttribut` ont des propriétés que vous pouvez lire pour obtenir des informations sur les attributs de votre dataclass.

Les objets de l'attribut Dataclass peuvent être modifiés, mais la structure sous-jacente de la base de données ne sera pas altérée.

Sommaire

.autoFilled : Booléen	contient Vrai si la valeur de l'attribut est automatiquement renseignée par 4D
.exposed : Boolean	mise à "true" si l'attribut est "exposed" dans REST
.fieldNumber : Integer	contient le numéro interne du champ 4D de l'attribut
.fieldType : Integer	contient le type 4D de l'attribut dans la base
.indexed : Booléen	contient Vrai s'il existe un index B-tree ou Cluster B-tree dans l'attribut
.inverseName : Text	retourne le nom de l'attribut qui est à l'autre extrémité du lien
.keywordIndexed : Boolean	contient True s'il existe un index de mot-clé sur l'attribut
.kind : Text	retourne la catégorie de l'attribut
.mandatory : Boolean	contient Vrai si une saisie de valeur Null pour l'attribut est rejetée
.name : Text	retourne le nom de l'objet <code>dataClassAttribute</code> sous forme de chaîne
.path : Text	retourne le chemin d'un attribut alias basé sur une relation
.readOnly : Boolean	mise à "true" si l'attribut est en lecture seule
.relatedDataClass : Text	retourne le nom de la dataclass liée à l'attribut
.type : Texte	contient le type conceptuel de la valeur de l'attribut
.unique : Booléen	contient Vrai si la valeur de l'attribut doit être unique

.autoFilled

► Historique

`.autoFilled : Booléen`

Description

La propriété `.autoFilled` contient Vrai si la valeur de l'attribut est automatiquement renseignée par 4D. Cette propriété correspond aux propriétés de champ 4D suivantes :

- "Autoincrement", for les champs de type numérique

- "Auto UUID", pour les champs UUID (de type alpha).

Cette propriété n'est pas retornée si `.kind` = "relatedEntity" ou "relatedEntities".

Pour la programmation générique, vous pouvez utiliser `Bool` (`dataClassAttribute.autoFilled`) pour obtenir une valeur valide (false) même si `.autoFilled` n'est pas retourné.

.exposed

► Historique

`.exposed` : Boolean

Description

La propriété `.exposed` est mise à "true" si l'attribut est "exposed" dans REST.

Voir aussi

[DataClass.getInfo\(\)](#)

.fieldNumber

► Historique

`.fieldNumber` : Integer

Description

La propriété `.fieldNumber` contient le numéro interne du champ 4D de l'attribut.

Cette propriété n'est pas retornée si `.kind` = "relatedEntity" ou "relatedEntities".

Pour la programmation générique, vous pouvez utiliser `Num` (`dataClassAttribute.fieldNumber`) pour obtenir une valeur valide (0) même si `.fieldNumber` n'est pas retourné.

.fieldType

► Historique

`.fieldType` : Integer

Description

La propriété `.fieldType` contient le type 4D de l'attribut dans la base. Elle dépend du `.kind` de l'attribut.

Valeurs possibles :

dataClassAttribute.kind	fieldType
storage	Type de champ 4D correspondant, voir Value type
relatedEntity	38 (Is object)
relatedEntities	42 (Is collection)
calculated	<ul style="list-style-type: none"> scalaire : type de champ 4D correspondant, voir Value type entity: 38 (Is object) entity selection: 42 (Is collection)
alias	<ul style="list-style-type: none"> scalaire : type de champ 4D correspondant, voir Value type entity: 38 (Is object) entity selection: 42 (Is collection)

Voir aussi

[.type](#)

.indexed

► Historique

.indexed : Booléen

Description

La propriété `.indexed` contient Vrai s'il existe un index B-tree ou Cluster B-tree dans l'attribut.

Cette propriété n'est pas retournée si `.kind = "relatedEntity"` ou `"relatedEntities"`.

Pour la programmation générique, vous pouvez utiliser `Bool (dataClassAttribute.indexed)` pour obtenir une valeur valide (`false`) même si `.indexed` n'est pas retourné.

.inverseName

► Historique

.inverseName : Text

Description

La propriété `.inverseName` retourne le nom de l'attribut qui est à l'autre extrémité du lien.

Cette propriété n'est pas retournée si `.kind = "storage"`. Elle doit être du type `"relatedEntities"` ou `"relatedEntities"`.

Pour la programmation générique, vous pouvez utiliser `String (dataClassAttribute.inverseName)` pour obtenir une valeur valide ("") même si `.inverseName` n'est pas retourné.

.keywordIndexed

► Historique

.keywordIndexed : Boolean

Description

La propriété `.keywordIndexed` contient True s'il existe un index de mot-clé sur l'attribut.

Cette propriété n'est pas retornée si `.kind` = "relatedEntity" ou "relatedEntities".

Pour la programmation générique, vous pouvez utiliser Bool (dataClassAttribute.keywordIndexed) pour obtenir une valeur valide (false) même si `.keywordIndexed` n'est pas retourné.

.kind

► Historique

`.kind` : Text

Description

La propriété `.kind` retourne la catégorie de l'attribut. La valeur retournée peut être l'une des chaînes suivantes :

- "storage" : attribut de stockage (ou scalaire), c'est-à-dire un attribut stockant une valeur, et non une référence à un autre attribut
- "calculated" : attribut calculé, c'est-à-dire défini par [la fonction get](#)
- "alias": attribut basé sur [un autre attribut](#)
- "relatedEntity" : attribut relationnel N -> (référence vers une entité)
- "relatedEntities" : attribut relationnel 1 -> N (référence vers une entity selection)

Exemple

Considérons les tables et relations suivantes :

```
var $attKind : Text
$attKind:=ds.Employee.lastname.kind // $attKind="storage"
$attKind:=ds.Employee.manager.kind // $attKind="relatedEntity"
$attKind:=ds.Employee.directReports.kind // $attKind="relatedEntities"
```

.mandatory

► Historique

`.mandatory` : Boolean

Description

La propriété `.mandatory` contient Vrai si une saisie de valeur Null pour l'attribut est rejetée.

Cette propriété n'est pas retornée si `.kind` = "relatedEntity" ou "relatedEntities".

Pour la programmation générique, vous pouvez utiliser Bool (dataClassAttribute.mandatory) pour obtenir une valeur valide (false) même si `.mandatory` n'est pas retourné. Attention : Cette propriété correspond à la propriété du champ "Reject NULL value input" au niveau de la base de données 4D. Elle n'est pas liée à la propriété "Mandatory" existante qui est une option de contrôle de la saisie de données pour une table.

.name

► Historique

`.name` : Text

Description

La propriété `.name` retourne le nom de l'objet `dataClassAttribute` sous forme de chaîne.

Exemple

```
var $attName : Text  
$attName:=ds.Employee.lastname.name //attName="lastname"
```

.path

► Historique

.path : Text

Description

La propriété `.path` retourne le chemin d'un attribut alias basé sur une relation.

Exemple

```
var $path : Text  
$path:=ds.Teacher.students.path //path="courses.student"
```

.readOnly

► Historique

.readOnly : Boolean

Description

La propriété `.readOnly` est mise à "true" si l'attribut est en lecture seule.

Par exemple, les champs calculés sans fonction `set` sont en lecture seule.

.relatedDataClass

► Historique

.relatedDataClass : Text

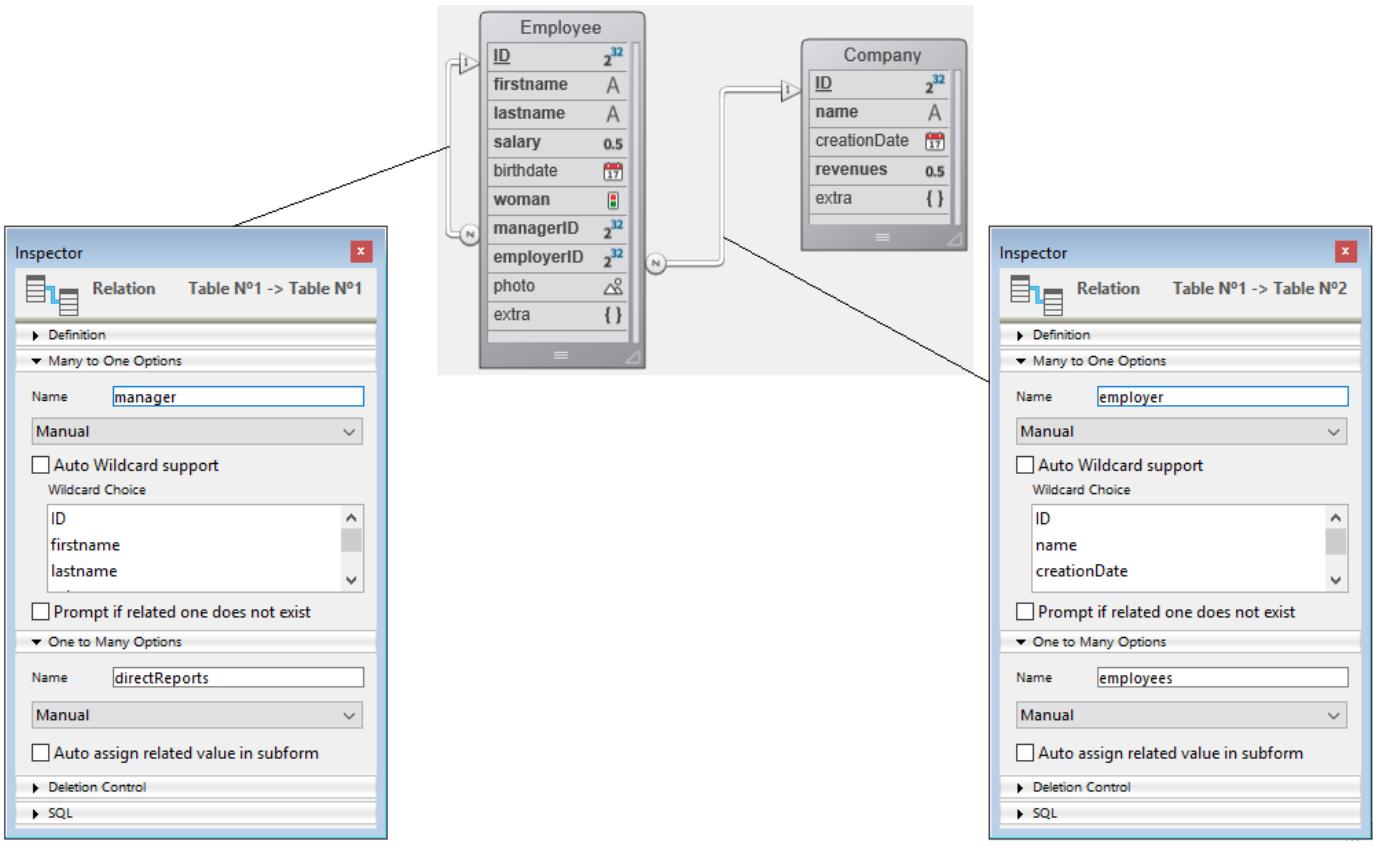
Description

Cette propriété n'est disponible qu'avec les attributs de la propriété "relatedEntity" ou "relatedEntities" `.kind`.

La propriété `.relatedDataClass` retourne le nom de la dataclass liée à l'attribut.

Exemple

Considérons les tableaux et relations suivants :



```

var $relClass1; $relClassN : Text
$relClass1:=ds.Employee.employer.relatedDataClass // $relClass1="Company"
$relClassN:=ds.Employee.directReports.relatedDataClass // $relClassN="Employee"

```

.type

► Historique

.type : Texte

Description

La propriété `.type` contient le type conceptuel de la valeur de l'attribut, utile pour de la programmation générique.

Le type de valeur conceptuelle dépend de l'attribut `.kind`.

Valeurs possibles :

dataClassAttribute.kind	type	Commentaire
storage	"blob", "bool", "date", "image", "number", "object", ou "string"	"nombre" est retourné pour tous les types numériques, y compris la durée. "string" est retourné pour les types de champs uuid, alpha et text. Les attributs "blob" sont des objets blob , ils sont gérés à l'aide de la classe Blob .
relatedEntity	related dataClass name	Ex : "Companies"
relatedEntities	related dataClass name + "Selection" suffix	Ex : "EmployeeSelection"
calculated	<ul style="list-style-type: none"> ● storage: type ("blob", "number", etc.) ● entity: dataClass name ● entity selection: dataClass name + "Selection" 	

Voir aussi

[.fieldType](#)

.unique

► Historique

.unique : Booléen

Description

La propriété `.unique` contient Vrai si la valeur de l'attribut doit être unique. Cette propriété correspond à la propriété de champ 4D "Unique".

Cette propriété n'est pas retournée si `.kind` = "relatedEntity" ou "relatedEntities".

Pour la programmation générique, vous pouvez utiliser `Bool (dataClassAttribute.unique)` pour obtenir une valeur valide (`false`) même si `.unique` n'est pas retourné.

DataStore

Un [Datastore](#) correspond à l'objet d'interface fourni par ORDA pour référencer et accéder à une base de données. Les objets `Datastore` sont retournés par les commandes suivantes :

- [ds](#) : un raccourci vers le datastore principal
- [Open datastore](#) : pour ouvrir n'importe quel datastore distant

Sommaire

<code>.cancelTransaction()</code>	annule la transaction
<code>.clearAllRemoteContexts()</code>	efface tous les attributs pour tous les contextes actifs dans le datastore
<code>.dataclassName : 4D.DataClass</code>	contient la description de la dataclass
<code>.encryptionStatus(): Object</code>	retourne un objet qui fournit le statut de chiffrement du fichier de données courant
<code>.getAllRemoteContexts() : Collection</code>	retourne une collection d'objets contenant des informations sur tous les contextes d'optimisation actifs dans le datastore
<code>.getInfo(): Object</code>	retourne un objet qui fournit des informations sur le datastore
<code>.getRemoteContextInfo(contextName : Text) : Object</code>	retourne un objet qui contient des informations sur le contexte d'optimisation <code>contextName</code> dans le datastore.
<code>.getRequestLog() : Collection</code>	retourne les requêtes ORDA enregistrées en mémoire sur le poste client
<code>.makeSelectionsAlterable()</code>	définit toutes les nouvelles sélections d'entités comme altérables par défaut dans tous les datastores de l'application
<code>.provideDataKey(curPassPhrase : Text) : Object</code> <code>.provideDataKey(curDataKey : Object) : Object</code>	permet de fournir une clé de chiffrement des données pour le fichier de données courant du datastore et détecte si la clé correspond aux données chiffrées
<code>.setAdminProtection(status : Boolean)</code>	permet de désactiver tout accès aux données sur le port web admin , y compris via le Data Explorer dans les sessions <code>WebAdmin</code>
<code>.setRemoteContextInfo(contextName : Text ; dataClassName : Text ; attributes : Text {; contextType : Text { ; pageLength : Integer}})</code> <code>.setRemoteContextInfo(contextName : Text ; dataClassName : Text; attributesColl : Collection {; contextType : Text { ; pageLength : Integer }})</code> <code>.setRemoteContextInfo(contextName : Text ; dataClassObject : 4D.DataClass ; attributes : Text {; contextType : Text { ; pageLength : Integer }})</code> <code>.setRemoteContextInfo(contextName : Text ; dataClassObject : 4D.DataClass ; attributesColl : Collection {;</code>	

`contextType : Text { ; pageLength : Integer } }`

relie les attributs de dataclass spécifiés au contexte d'optimisation `contextName`

`.startRequestLog()`

`.startRequestLog(file : 4D.File)`

`.startRequestLog(reqNum : Integer)`

lance l'enregistrement des requêtes ORDA sur le poste client

`.startTransaction()`

démarre une transaction dans le process courant sur la base de données du datastore

`.stopRequestLog()`

stoppe tout enregistrement des requêtes ORDA sur le poste client

`.validateTransaction()`

valide la transaction

ds

► Historique

`ds { (localID : Text) } : cs.DataStore`

Paramètres	Type		Description
localID	Text	->	Identifiant local du datastore distant
Résultat	cs.DataStore	<-	Nouvelle référence de datastore

Description

La commande `ds` retourne une nouvelle référence vers le datastore correspondant à la base de données 4D courante ou à la base désignée par `localID`.

Si vous omettez le paramètre `localID` (ou si vous passez une chaîne vide ""), la commande renvoie une référence au datastore correspondant à la base de données 4D locale (ou à la base 4D Server en cas d'ouverture d'une base de données distante sur 4D Server). Le datastore est ouvert automatiquement et est disponible directement via `ds`.

Vous pouvez également obtenir une référence sur un datastore distant ouvert en passant son identifiant local dans le paramètre `localID`. Le datastore doit avoir été préalablement ouvert avec la commande `Open datastore` par la base de données courante (hôte ou composant). L'identifiant local est défini lors de l'utilisation de cette commande.

La portée de l'identifiant local est la base de données dans laquelle le datastore a été ouvert.

Si aucun datastore nommé `localID` n'est trouvé, la commande renvoie Null.

Les objets disponibles dans le `cs.Datastore` sont créés à partir de la base de données cible en fonction des [règles générales](#) de correspondance d'ORDA.

Exemple 1

Utilisation du datastore principal de la base 4D :

```
$result:=ds.Employee.query("firstName = :1";"S@")
```

Exemple 2

```

var $connectTo; $firstFrench; $firstForeign : Object
var $frenchStudents; $foreignStudents : cs.DataStore

$connectTo:=New object("type";"4D Server";"hostname";"192.168.18.11:8044")
$frenchStudents:=Open datastore($connectTo;"french")

$connectTo.hostname:="192.168.18.11:8050"
$foreignStudents:=Open datastore($connectTo;"foreign")
//...
//...
$firstFrench:=getFirst("french";"Students")
$firstForeign:=getFirst("foreign";"Students")

```

```

//méthode getFirst
//getFirst(localID;dataclass) -> entity
#DECLARE( $localId : Text; $dataClassName : Text ) -> $entity : 4D.Entity

$0:=ds($localId)[${dataClassName}].all().first()

```

Open datastore

► Historique

`Open datastore(connectionInfo : Object ; localID : Text) : cs.DataStore`

Paramètres	Type		Description
connectionInfo	Object	->	Propriétés de connexion utilisées pour joindre le datastore distant
localID	Text	->	Identifiant à affecter au datastore ouvert sur l'application locale (obligatoire)
Résultat	cs.DataStore	<-	Objet datastore

Description

La commande `Open datastore` connecte l'application à la base de données 4D identifiée par le paramètre `connectionInfo` et retourne un objet `cs.DataStore` associé à l'alias local `localID`.

La base de données `connectionInfo` 4D doit être disponible en tant que datastore distant, c'est-à-dire :

- son serveur Web doit être lancé avec http et/ou https activé,
- son option [Activer le service REST](#) doit être cochée,
- au moins une licence client est disponible.

Si aucune base de données correspondante n'est trouvée, `Open datastore` retourne Null.

`localID` est un alias local de la session ouverte sur le datastore distant. Si `localID` existe déjà dans l'application, il est utilisé. Sinon, une nouvelle session `localID` est créée lors de l'utilisation de l'objet datastore.

Les objets disponibles dans le `cs.Datastore` sont créés à partir de la base de données cible en fonction des [règles générales](#) de correspondance d'ORDA.

Une fois la session ouverte, les instructions suivantes deviennent équivalentes et renvoient une référence sur le même objet datastore :

```

$myds:=Open datastore(connectionInfo;"myLocalId")
$myds2:=ds("myLocalId")
//$myds et $myds2 sont équivalents

```

Passez dans *connectionInfo* un objet décrivant le datastore distant auquel vous souhaitez vous connecter. Il peut contenir les propriétés suivantes (toutes les propriétés sont optionnelles, à l'exception de *hostname*) :

Propriété	Type	Description
hostname	Text	Nom ou adresse IP de la base de données distante + ":" + numéro de port (le numéro de port est obligatoire)
user	Text	Nom d'utilisateur
password	Text	Mot de passe de l'utilisateur
idleTimeout	Longint	Délai d'inactivité de la session (exprimé en minutes), au terme duquel la session est automatiquement fermée par 4D. Si cette propriété est omise, la valeur par défaut est 60 (1h). La valeur ne peut pas être < 60 (si une valeur inférieure est passée, le timeout est fixé à 60). Pour plus d'informations, voir Fermeture des sessions.
tls	Boolean	Utilisez une connexion sécurisée(*). Si cette propriété est omise, "false" par défaut. L'utilisation d'une connexion sécurisée est recommandée dans la mesure du possible.
type	Text	Doit être "4D Server"

(*) Si *tls* est vrai, le protocole HTTPS est utilisé si :

- HTTPS est activé sur le datastore distant
- Le port donné correspond au port HTTPS configuré dans les propriétés
- Un certificat valide et une clé de chiffrement privée sont installés dans la base. Sinon, l'erreur "1610 - Une requête vers l'hôte: "{xxx}" a échoué" est générée

Exemple 1

Connexion à un datastore distant sans utilisateur/mot de passe :

```
var $connectTo : Object
var $remoteDS : cs.DataStore
$connectTo:=New object("type";"4D Server";"hostname";"192.168.18.11:8044")
$remoteDS:=Open datastore($connectTo;"students")
ALERT("This remote datastore contains "+String($remoteDS.Students.all().length)+" students")
```

Exemple 2

Connexion à un datastore distant avec utilisateur/mot de passe/timeout/tls :

```
var $connectTo : Object
var $remoteDS : cs.DataStore
$connectTo:=New object("type";"4D Server";"hostname";\"192.168.18.11:4443";\
"user";"marie";"password";$pwd;"idleTimeout";70;"tls";True)
$remoteDS:=Open datastore($connectTo;"students")
ALERT("This remote datastore contains "+String($remoteDS.Students.all().length)+" students")
```

Exemple 3

Travailler avec plusieurs datastores distants :

```

var $connectTo : Object
var $frenchStudents; $foreignStudents : cs.DataStore
$connectTo:=New object("hostname";"192.168.18.11:8044")
$frenchStudents:=Open datastore($connectTo;"french")
$connectTo.hostname:="192.168.18.11:8050"
$foreignStudents:=Open datastore($connectTo;"foreign")
ALERT("They are "+String($frenchStudents.Students.all().length)+" French students")
ALERT("They are "+String($foreignStudents.Students.all().length)+" foreign students")

```

Gestion des erreurs

En cas d'erreur, la commande retourne Null. Si le datastore distant ne peut pas être joint (adresse incorrecte, web serveur non lancé, http et https non activés, etc.), l'erreur 1610 "Une requête vers l'hôte: {xxx} a échoué" est générée. Vous pouvez intercepter cette erreur avec une méthode installée par `ON ERR CALL`.

`.dataclassName`

► Historique

`.dataclassName` : 4D.DataClass

Description

Chaque dataclass d'un datastore est disponible en tant que propriété de l'objet `DataStore`. L'objet retourné contient la description de la dataclass.

Exemple

```

var $emp : cs.Employee
var $sel : cs.EmployeeSelection
$emp:=ds.Employee // $emp contient la dataclass Employee
$sel:=$emp.all() // obtient une entity selection de tous les employés

// vous pouvez également saisir directement :
$sel:=ds.Employee.all()

```

`.cancelTransaction()`

► Historique

`.cancelTransaction()` | Paramètres | Type | | Description | | ----- | ---- |::| ----- | | | | Ne requiert aucun paramètre |

Description

La fonction `.cancelTransaction()` annule la transaction ouverte par la fonction `.startTransaction()` au niveau correspondant dans le process courant du datastore spécifié.

La fonction `.cancelTransaction()` annule toutes les modifications apportées aux données durant la transaction.

Vous pouvez imbriquer plusieurs transactions (sous-transactions). Si la transaction principale est annulée, toutes ses sous-transactions sont également annulées, même si elles ont été validées individuellement à l'aide de la fonction `.validateTransaction()`.

Exemple

Voir l'exemple de la fonction `.startTransaction()`.

`.clearAllRemoteContexts()`

► Historique

.clearAllRemoteContexts() | Paramètres | Type | Description | | ----- | ---- | ::| ----- | | | | |
Ne requiert aucun paramètre |

Description

La fonction `.clearAllRemoteContexts()` efface tous les attributs pour tous les contextes actifs dans le datastore.

Cette fonction est utile principalement dans le contexte du débogage. Gardez à l'esprit que lorsque vous ouvrez le débogueur, il envoie des requêtes au serveur et récupère tous les attributs de la dataclass pour les afficher. Cela peut surcharger vos contextes avec des données inutiles.

Si cela se produit, vous pouvez utiliser `.clearAllRemoteContexts()` pour réinitialiser vos contextes.

Voir aussi

[.getRemoteContextInfo\(\)](#)
[.getAllRemoteContexts\(\)](#)
[.setRemoteContextInfo\(\)](#)

.encryptionStatus()

► Historique

`.encryptionStatus(): Object`

Paramètres	Type		Description
Résultat	Object	<-	Informations sur le chiffrement du datastore courant et de chaque table

Description

La fonction `.encryptionStatus()` retourne un objet qui fournit le statut de chiffrement du fichier de données courant (i.e., le fichier de données du datastore `ds`). Le statut de chiffrement pour chaque table est également fourni.

Utilisez la commande `Data file encryption status` pour déterminer l'état de cryptage de tout autre fichier de données.

Valeur renournée

L'objet retourné contient les propriétés suivantes :

Propriété		Type	Description
isEncrypted		Boolean	Vrai si le fichier de données est chiffré
keyProvided		Boolean	Vrai si la clé de chiffrement correspondant au fichier de données chiffré est fournie(*) .
tables		Object	Objet contenant autant de propriétés que de tables chiffrables ou chiffrées.
	<i>tableName</i>	Object	Table chiffrable ou chiffrée
	name	Text	Nom de la table
	num	Number	Numéro de la table
	isEncryptable	Boolean	Vrai si la table est déclarée chiffrable dans le fichier de structure
	isEncrypted	Boolean	Vrai si les enregistrements de la table sont chiffrés dans le fichier de données

(*) La clé de chiffrement peut être fournie :

- via la fonction `.provideDataKey()`,
- à la racine d'un appareil connecté avant l'ouverture du datastore,
- via la commande `Discover data key`.

Exemple

Vous souhaitez connaitre le nombre de tables chiffrées dans le fichier de données courant :

```
var $status : Object

$status:=dataStore.encryptionStatus()

If($status.isEncrypted) //the database is encrypted
    C_LONGINT($vcount)
    C_TEXT($tabName)
    For each($tabName;$status.tables)
        If($status.tables[$tabName].isEncrypted)
            $vcount:=$vcount+1
        End if
    End for each
    ALERT(String($vcount)+" table(s) chiffrée(s) dans ce datastore.")
Else
    ALERT("This database is not encrypted.")
End if
```

.getAllRemoteContexts()

► Historique

`.getAllRemoteContexts()` : Collection

Paramètres	Type		Description
Résultat	Object	<-	Collection d'objets contextes d'optimisation

Mode avancé : Cette fonction est destinée aux développeurs qui souhaitent personnaliser les fonctionnalités par défaut de ORDA dans le cadre de configurations spécifiques. Dans la plupart des cas, vous n'aurez pas besoin de l'utiliser.

Description

La fonction `.getAllRemoteContexts()` fonction retourne une collection d'objets contenant des informations sur tous les contextes d'optimisation actifs dans le datastore.

Pour plus d'informations sur la création des contextes, voir [Optimisation client/serveur](#).

Chaque objet de la collection renvoyée contient les propriétés listées dans la section [.getRemoteContextInfo\(\)](#).

Exemple

Le code suivant définit deux contextes et les récupère à l'aide de `.getAllRemoteContexts()` :

```

var $ds : 4D.DataStoreImplementation
var $persons : cs.PersonsSelection
var $addresses : cs.AddressSelection
var $p : cs.PersonsEntity
var $a : cs.AddressEntity
var $contextA; $contextB : Object
var $info : Collection
var $text : Text

// Accès datastore distant
$ds:=Open datastore(New object("hostname"; "www.myserver.com"); "myDS")

// définition contexte A
$contextA:=New object("context"; "contextA")
$persons:=$ds.Persons.all($contextA)
$text:=""
For each ($p; $persons)
    $text:=$p.firstname+" lives in "+$p.address.city+"
End for each

// définition contexte B
$contextB:=New object("context"; "contextB")
$addresses:=$ds.Address.all($contextB)
$text:=""
For each ($a; $addresses)
    $text:=$a.zipCode
End for each

// récupérer tous les contextes distants (dans ce cas, contextA et contextB)
$info:=$ds.getAllRemoteContexts()
// $info = [{name:"contextB"; dataclass:"Address"; main:"zipCode"}, {name:"contextA"; dataclass:"Persons"; main:"firstname,address.city"}]

```

Cet exemple est à but de démonstration, il n'est pas destiné à une implémentation réelle.

Voir aussi

- [.getRemoteContextInfo\(\)](#)
- [.setRemoteContextInfo\(\)](#)
- [.clearAllRemoteContexts\(\)](#)

.getInfo()

► Historique

[.getInfo\(\): Object](#)

Paramètres	Type		Description
Résultat	Object	<-	Propriétés du datastore

Description

La fonction `.getInfo()` fonction retourne un objet qui fournit des informations sur le datastore. Cette fonction est utile pour l'écriture de code générique.

Objet retourné

Propriété	Type	Description															
type	string	<ul style="list-style-type: none"> "4D" : datastore principal, disponible via ds "4D Server" : datastore distant ouvert avec Open datastore 															
networked	boolean	<ul style="list-style-type: none"> Vrai : le datastore est accessible via une connexion réseau. Faux : le datastore n'est pas accessible via une connexion réseau (base locale) 															
localID	text	Identifiant du datastore sur la machine. Correspond à la chaîne localID donnée avec la commande <code>Open datastore</code> . Chaîne vide ("") pour le datastore principal.															
connection	object	Objet décrivant la connexion au datastore distant (non retourné pour le datastore principal). Propriétés disponibles : <table border="1"> <thead> <tr> <th>Propriété</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>hostname</td> <td>text</td> <td>Adresse IP ou nom du datastore distant + ":" + numéro de port</td> </tr> <tr> <td>tls</td> <td>boolean</td> <td>Vrai si une connexion sécurisée est utilisée avec le datastore distant</td> </tr> <tr> <td>idleTimeout</td> <td>number</td> <td>Délai d'inactivité autorisé de la session (en minutes)</td> </tr> <tr> <td>user</td> <td>text</td> <td>Utilisateur authentifié sur le datastore distant</td> </tr> </tbody> </table>	Propriété	Type	Description	hostname	text	Adresse IP ou nom du datastore distant + ":" + numéro de port	tls	boolean	Vrai si une connexion sécurisée est utilisée avec le datastore distant	idleTimeout	number	Délai d'inactivité autorisé de la session (en minutes)	user	text	Utilisateur authentifié sur le datastore distant
Propriété	Type	Description															
hostname	text	Adresse IP ou nom du datastore distant + ":" + numéro de port															
tls	boolean	Vrai si une connexion sécurisée est utilisée avec le datastore distant															
idleTimeout	number	Délai d'inactivité autorisé de la session (en minutes)															
user	text	Utilisateur authentifié sur le datastore distant															

- Si la fonction `. getInfo()` est exécutée sur un 4D Server ou un 4D monoposte, `networked` est Faux.
- Si la fonction `. getInfo()` est exécutée sur un 4D distant, `networked` est Vrai

Exemple 1

```
var $info : Object

$info:=ds.getInfo() //Exécuté sur 4D Server ou 4D
//>{"type":"4D","networked":false,"localID":""}

$info:=ds.getInfo() // Exécuté sur 4D distant
//>{"type":"4D","networked":true,"localID":""}
```

Exemple 2

Sur un datastore distant :

```
var $remoteDS : cs.DataStore
var $info; $connectTo : Object

$connectTo:=New object("hostname";"111.222.33.44:8044";"user";"marie";"password";"aaaa")
$remoteDS:=Open datastore($connectTo;"students")
$info:=$remoteDS.getInfo()

//{"type":"4D Server",
//"localID":"students",
//"networked":true,
//"connection":{hostname:"111.222.33.44:8044", "tls":false, "idleTimeout":2880, "user":"marie"}}
```

.getRemoteContextInfo()

► Historique

`.getRemoteContextInfo(contextName : Text) : Object`

Paramètres	Type		Description
contextName	Text	->	Nom du contexte
Résultat	Object	<-	Description du contexte

Mode avancé : Cette fonction est destinée aux développeurs qui souhaitent personnaliser les fonctionnalités par défaut de ORDA dans le cadre de configurations spécifiques. Dans la plupart des cas, vous n'aurez pas besoin de l'utiliser.

Description

La fonction `.getRemoteContextInfo()` retourne un objet qui contient des informations sur le contexte d'optimisation `contextName` dans le datastore..

Pour plus d'informations sur la création des contextes, voir [Optimisation client/serveur](#).

Objet retourné

L'objet retourné contient les propriétés suivantes :

Propriété	Type	Description
name	Text	Nom du contexte
main	Text	Attribut(s) associé(s) au contexte (les noms d'attributs sont séparés par des virgules)
dataclass	Text	Nom de la dataclass
currentItem (optionnel)	Text	Attributs du mode page si le contexte est lié à une list box. Retourn <code>Null</code> ou un élément de texte vide si le contexte n'est pas utilisé pour une list box, ou s'il n'y a pas de contexte pour l'élément courant (currentItem)

Comme les contextes se comportent comme des filtres d'attributs, si `main` est retourné vide, cela signifie qu'aucun filtre n'est appliqué et que le serveur donc retourne tous les attributs de la dataclass.

Exemple

Voir l'exemple de la section [.setRemoteContextInfo\(\)](#).

Voir aussi

[.setRemoteContextInfo\(\)](#)
[.getAllRemoteContexts\(\)](#)
[.clearAllRemoteContexts\(\)](#)

.getRequestLog()

► Historique

`.getRequestLog()` : Collection

Paramètres	Type		Description
Résultat	Collection	<-	Collection d'objets décrivant les requêtes

Description

La fonction `.getRequestLog()` retourne les requêtes ORDA enregistrées en mémoire sur le poste client.

L'enregistrement des requêtes ORDA doit avoir été préalablement activé à l'aide de la fonction [.startRequestLog\(\)](#) .

Cette fonction doit être appelée sur un client 4D distant, sinon elle retourne une collection vide. Elle est conçue à des fins de débogage dans les configurations client/serveur.

Valeur retournée

Collection d'objets requête empilés. La requête la plus récente porte l'indice 0.

Pour plus de détails sur le format d'enregistrement des requêtes ORDA, veuillez consulter la section [ORDA client requests](#).

Exemple

Voir l'exemple 2 de [.startRequestLog\(\)](#).

.isAdminProtected()

► Historique

.isAdminProtected() : Boolean

Paramètres	Type	Description
Résultat	Booléen	<- Vrai si l'accès au Data Explorer est désactivé, Faux s'il est activé (défaut)

Description

La fonction `.isAdminProtected()` retourne `Vrai` si l'accès au [Data Explorer](#) est désactivé pour la session courante.

Par défaut, l'accès au Data Explorer est autorisé pour les sessions `webAdmin`, mais il peut être désactivé pour empêcher que les administrateurs puissent accéder aux données (cf. fonction [.setAdminProtection\(\)](#)).

Voir aussi

[.setAdminProtection\(\)](#)

.makeSelectionsAlterable()

► Historique

.makeSelectionsAlterable() | Paramètres | Type | | Description | | ----- | ---- | ::| ----- | | | | |
Ne requiert aucun paramètre |

Description

La fonction `.makeSelectionsAlterable()` définit toutes les nouvelles sélections d'entités comme altérables par défaut dans tous les datastores de l'application (y compris les [datastores distants](#)). Elle est destinée à être appelée une fois, par exemple dans la méthode base `On Startup`.

Lorsque cette méthode n'est pas appelée, les nouvelles sélections d'entités peuvent être partageables, selon la nature de leur "parent" ou la façon dont elles sont créées (voir la section [Entity selections partageables et non partageables](#)).

Cette fonction ne modifie pas les sélections d'entités créées par `.copy()` ou `OB Copy` lorsque l'option explicite `ck shared` est utilisée.

Compatibilité : Cette fonction doit être utilisée uniquement dans des projets convertis à partir de versions de 4D antérieures à 4D v18 R5 et contenant des appels `.add()`. Dans ce contexte, l'utilisation de `.makeSelectionsAlterable()` peut faire gagner du temps en restaurant instantanément le précédent comportement 4D dans les projets existants. En revanche, l'utilisation de cette méthode dans les nouveaux projets créés dans 4D v18 R5 et les versions plus récentes n'est pas recommandée, car elle empêche le partage des entity selections, ce qui offre de meilleures performances et une plus grande évolutivité.

.provideDataKey()

► Historique

```
.provideDataKey( curPassPhrase : Text ) : Object
.provideDataKey( curDataKey : Object ) : Object
```

Paramètres	Type		Description
curPassPhrase	Text	->	Phrase secrète courante
curDataKey	Object	->	Clé de chiffrement des données courante
Résultat	Object	<-	Résultat de la mise en correspondance de la clé de chiffrement

Description

La fonction `.provideDataKey()` permet de fournir une clé de chiffrement des données pour le fichier de données courant du datastore et détecte si la clé correspond aux données chiffrées. Cette fonction peut être utilisée à l'ouverture d'une base chiffrée, ou à l'exécution de n'importe quelle opération de chiffrement qui nécessite la clé de chiffrement, telle que le re-chiffrement du fichier de données.

- La fonction `.provideDataKey()` doit être appelée dans une base de données chiffrée. Si elle est appelée dans une base de données non chiffrée, l'erreur 2003 (la clé de cryptage ne correspond pas aux données.) est retournée. Utilisez la commande `Data file encryption status` pour déterminer si la base de données est chiffrée.
- La fonction `.provideDataKey()` ne peut pas être appelée depuis un 4D distant ou un datastore distant chiffré.

Si vous utilisez le paramètre `curPassPhrase`, passez la chaîne utilisée pour générer la clé de chiffrement des données. Lorsque vous utilisez ce paramètre, une clé de chiffrement est générée.

Si vous utilisez le paramètre `curDataKey`, passez un objet (avec la propriété `encodedKey`) contenant la clé de chiffrement des données. Cette clé peut avoir été générée à l'aide de la commande `New data key`.

Si une clé de chiffrement des données valide est fournie, elle est ajoutée à la `keyChain` dans la mémoire et le mode chiffrement est activé :

- toutes les modifications de données apportées dans les tables chiffrables sont chiffrées sur le disque (fichiers .4DD, .journal, .4Dindx).
- toutes les données chargées à partir de tables chiffrables sont déchiffrées dans la mémoire

Résultat

Le résultat de la commande est décrit dans l'objet retourné :

Propriété		Type	Description
success		Booléen	Vrai si la clé de chiffrement fournie correspond aux données chiffrées, sinon Faux
			Les propriétés ci-dessous sont retournées uniquement si success est à Faux
status		Nombre	Code d'erreur (4 si la clé de chiffrement fournie est erronée)
statusText		Text	Message d'erreur
errors		Collection	Pile d'erreurs. La première erreur possède l'indice le plus élevé.
	[].componentSignature	Text	Nom du composant interne
	[].errCode	Nombre	Numéro de l'erreur
	[].message	Text	Message d'erreur

Si aucun paramètre `curPassphrase` ou `curDataKey` n'est fourni, `.provideDataKey()` retourne null (aucune erreur n'est générée).

Exemple

```
var $keyStatus : Object
var $passphrase : Text

$passphrase:=Request("Enter the passphrase")
If(OK=1)
    $keyStatus:=ds.provideDataKey($passphrase)
    If($keyStatus.success)
        ALERT("You have provided a valid encryption key")
    Else
        ALERT("You have provided an invalid encryption key, you will not be able to work with encrypted d
End if
End if
```

.setAdminProtection()

► Historique

.setAdminProtection(*status* : Boolean)

Paramètres	Type		Description
status	Booléen	- >	Vrai pour désactiver l'accès au Data Explorer sur le port <code>webAdmin</code> , Faux (défaut) pour permettre l'accès

Description

La fonction `.setAdminProtection()` permet de désactiver tout accès aux données sur le [port web admin](#), y compris via le [Data Explorer](#) dans les sessions [WebAdmin](#).

Par défaut lorsque la fonction n'est pas appelée, l'accès aux données est possible via le Data Explorer sur le port d'administration web pour une session avec le privilège [WebAdmin](#). Dans certaines configurations, par exemple lorsque le serveur d'application est hébergé sur la machine d'un prestataire de service, vous pouvez souhaiter que l'administrateur ne puisse pas visualiser vos données, même s'il peut accéder à la configuration du serveur, y compris au paramétrage de l'[access key](#).

Dans ce cas, vous pouvez appeler cette fonction pour désactiver l'accès aux données depuis le Data Explorer sur le port web admin de la machine, même pour les sessions utilisateurs ayant le privilège [WebAdmin](#). Lorsque cette fonction est exécutée, le fichier de données est immédiatement protégé et le statut est sauvegardé sur disque : le fichier de données sera protégé même si l'application est redémarrée.

Exemple

Vous créez une méthode projet `protectDataFile` à appeler par exemple avant le déploiement :

```
ds.setAdminProtection(True) //Désactive l'accès aux données de l'Explorateur de données
```

Voir aussi

[.isAdminProtected\(\)](#)

.setRemoteContextInfo()

► Historique

.setRemoteContextInfo(*contextName* : Text ; *dataClassName* : Text ; *attributes* : Text {; *contextType* : Text { ; *pageLength* : Integer}})
.setRemoteContextInfo(*contextName* : Text ; *dataClassName* : Text; *attributesColl* : Collection {; *contextType* : Text {

```

; pageLength : Integer } } )
.setRemoteContextInfo( contextName : Text ; dataClassObject : 4D.DataClass ; attributes : Text {; contextType : Text {
; pageLength : Integer } } )
.setRemoteContextInfo( contextName : Text ; dataClassObject : 4D.DataClass ; attributesColl : Collection {;
contextType : Text { ; pageLength : Integer } } )

```

Paramètres	Type		Description
contextName	Text	->	Nom du contexte
dataClassName	Text	->	Nom de la dataclass
dataClassObject	4D.DataClass	->	Objet dataclass (e.g datastore.Employee)
attributes	Text	->	Liste d'attributs séparés par des virgules
attributesColl	Collection	->	Collection de noms d'attributs (text)
contextType	Text	->	Si passé, "main" ou "currentItem"
pageLength	Integer	->	Taille de page de l'entity selection associée au contexte (80 par défaut)

Mode avancé : Cette fonction est destinée aux développeurs qui souhaitent personnaliser les fonctionnalités par défaut de ORDA dans le cadre de configurations spécifiques. Dans la plupart des cas, vous n'aurez pas besoin de l'utiliser.

Description

La fonction `.setRemoteContextInfo()` relie les attributs de dataclass spécifiés au contexte d'optimisation `contextName`. Si un contexte d'optimisation existe déjà pour les attributs spécifiés, la commande le remplace.

Lorsque vous passez un contexte aux fonctions de classe ORDA, l'optimisation des requêtes REST est déclenchée immédiatement :

- la première entité n'est pas chargée intégralement, à la différence du mode automatique
- des pages de 80 entités (ou de `pageLength` entités) sont immédiatement demandées au serveur avec uniquement les attributs du contexte

Pour plus d'informations sur la création des contextes, voir [Optimisation client/serveur](#)

Dans `contextName`, passez le nom du contexte d'optimisation à lier aux attributs de la dataclass.

Pour désigner la dataclass qui doit recevoir le contexte, vous pouvez passer un `dataClassName` ou un `dataClassObject`.

Pour désigner les attributs à lier au contexte, passez soit une liste d'attributs séparés par des virgules dans `attributes` (Text), soit une collection de noms d'attributs dans `attributesColl` (collection de textes).

Si `attributes` est un texte vide, ou si `attributesColl` est une collection vide, tous les attributs scalaires de la dataclass sont intégrés au contexte d'optimisation. Si vous passez un attribut qui n'existe pas dans la dataclass, la fonction l'ignore et une erreur est générée.

Vous pouvez passer un `contextType` pour spécifier si le contexte est standard ou s'il s'agit du contexte de l'élément courant de l'entity selection affichée dans une list box :

- Si sa valeur est "main" (défaut), `contextName` désigne un contexte standard.
- Si sa valeur est " currentItem", les attributs passés sont intégrés dans le contexte de l'élément courant. Voir [Listbox basée sur une sélection d'entités](#).

Dans `pageLength`, spécifiez le nombre d'entités de dataclass à demander au serveur.

Vous pouvez passer une `pageLength` pour un attribut relationnel qui est une entity selection (1-vers-N). La syntaxe est `relationAttributeName:pageLength` (e.g employees:20).

Exemple 1

```

var $ds : 4D.DataStoreImplementation
var $persons : cs.PersonsSelection
var $p : cs.PersonsEntity
var $contextA : Object
var $info : Object
var $text : Text

// Ouvrir datastore distant
$ds:=Open datastore(New object("hostname"; "www.myserver.com"); "myDS")

// définir le contexte
$contextA:=New object("context"; "contextA")
$ds.setRemoteContextInfo("contextA"; $ds.Persons; "firstname, lastname")

// envoi de requêtes au serveur dans une boucle
$persons:=$ds.Persons.all($contextA)
$text:=""
For each ($p; $persons)
    $text:=$p.firstname + " " + $p.lastname
End for each

// vérifier le contenu du contexte
$info:=$ds.getRemoteContextInfo("contextA")
// $info = {name:"contextA";dataclass:"Persons";main:"firstname, lastname"}

```

Cet exemple est à but de démonstration, il n'est pas destiné à une implémentation réelle.

Exemple 2

Le code suivant demande des pages de 30 entités de la dataclass `Address` au serveur. Les entités retournées contiennent uniquement l'attribut `zipCode`.

Pour chaque entité `Address`, 20 entités `Persons` sont retournées, et elles contiennent uniquement les attributs `lastname` et `firstname` :

```

var $ds : 4D.DataStoreImplementation

$ds:=Open datastore(New object("hostname"; "www.myserver.com"); "myDS")

$ds.setRemoteContextInfo("contextA"; $ds.Address; "zipCode, persons:20,\n
persons.lastname, persons.firstname"; "main"; 30)

```

Example 3 - Listbox

```

// Au chargement du formulaire
Case of
  : (Form event code=On Load)

    Form.ds:=Open datastore(New object("hostname"; "www.myserver.com"); "myDS")

    // définir les attributs du contexte de page
    Form.ds.setRemoteContextInfo("LB"; Form.ds.Persons; "age, gender,\n
      children"; "currentItem")

    Form.settings:=New object("context"; "LB")
    Form.persons:=Form.ds.Persons.all(Form.settings)
    // Form.persons est affiché dans une list box

End case

// lorsque vous récupérez les attributs dans le contexte de l'élément courant :
Form.currentItemLearntAttributes:=Form.selectedPerson.getRemoteContextAttributes()
// Form.currentItemLearntAttributes = "age, gender, children"

```

Voir aussi

[.getRemoteContextInfo\(\)](#)
[.getAllRemoteContexts\(\)](#)
[.clearAllRemoteContexts\(\)](#)

.startRequestLog()

► Historique

[.startRequestLog\(\)](#)
[.startRequestLog\(file : 4D.File \)](#)
[.startRequestLog\(reqNum : Integer \)](#)

Paramètres	Type		Description
file	4D.File	->	Objet File
reqNum	Integer	->	Nombre de requêtes à conserver en mémoire

Description

La fonction `.startRequestLog()` fonction lance l'enregistrement des requêtes ORDA sur le poste client.

Cette fonction doit être appelée sur un 4D distant, sinon elle ne fait rien. Elle est conçue à des fins de débogage dans les configurations client/serveur.

L'enregistrement des requêtes ORDA peut être effectué dans un fichier ou dans la mémoire, en fonction du type de paramètre :

- Si vous avez passé un objet `file` créé à l'aide de la commande `File`, les données de l'enregistrement sont écrites dans ce fichier sous forme de collection d'objets (format JSON). Chaque objet représente une requête. Si le fichier n'existe pas encore, il est créé. Sinon, s'il existe déjà, les nouvelles données d'enregistrement y sont ajoutées. Si la fonction `.startRequestLog()` est appelée avec un fichier alors qu'un enregistrement des requêtes est déjà en cours en mémoire, l'enregistrement en mémoire est stoppé et vidé.

Un caractère `]` doit être ajouté manuellement à la fin du fichier pour effectuer une validation JSON

- Si vous avez passé un numéro `reqNum`, l'enregistrement en mémoire est vidé (le cas échéant) et un nouvel enregistrement est lancé. Il gardera en mémoire les requêtes jusqu'à atteindre le nombre `reqNum`, auquel cas les entrées précédentes sont vidées (pile FIFO). Si la fonction `.startRequestLog()` est appelée avec un `reqNum` alors qu'un enregistrement des requêtes dans un fichier est déjà en cours, l'enregistrement dans le fichier est stoppé.

- Si vous n'avez passé aucun paramètre, l'enregistrement est lancé dans la mémoire. Si `.startRequestLog()` a été préalablement appelée avec un `reqNum` (avant un `.stopRequestLog()`), les données enregistrées sont empilées dans la mémoire jusqu'au prochain vidage ou appel de `.stopRequestLog()`.

Pour plus de détails sur le format d'enregistrement des requêtes ORDA, veuillez consulter la section [ORDA client requests](#).

Exemple 1

Vous souhaitez enregistrer des requêtes ORDA clientes dans un fichier et utiliser le numéro de séquence de l'enregistrement :

```
var $file : 4D.File
var $e : cs.PersonsEntity

$file:=File("/LOGS/ORDAResults.txt") //dossier logs

SET DATABASE PARAMETER(Client Log Recording;1) //pour déclencher le numéro de séquence log global
ds.startRequestLog($file)
$e:=ds.Persons.get(30001) //envoyer une requête
ds.stopRequestLog()
SET DATABASE PARAMETER(Client Log Recording;0)
```

Exemple 2

Vous souhaitez enregistrer des requêtes ORDA clientes dans la mémoire :

```
var $es : cs.PersonsSelection
var $log : Collection

ds.startRequestLog(3) //garde 3 requêtes dans la mémoire

$es:=ds.Persons.query("name=:1";"Marie")
$es:=ds.Persons.query("name IN :1";New collection("Marie"))
$es:=ds.Persons.query("name=:1";"So@")

$log:=ds.getRequestLog()
ALERT("The longest request lasted: "+String($log.max("duration"))+" ms")
```

.startTransaction()

► Historique

`.startTransaction()` | Paramètres | Type | Description | ----- | ---- | | ----- | | | | Ne requiert aucun paramètre |

Description

La fonction `.startTransaction()` démarre une transaction dans le process courant sur la base de données du datastore. Toute modification apportée aux entités du datastore dans le process de la transaction est temporairement stockée jusqu'à ce que la transaction soit validée ou annulée.

Si cette méthode est appelée sur le datastore principal (c'est-à-dire le datastore retourné par la commande `ds`), la transaction est appliquée à toutes les opérations effectuées sur le datastore principal et sur la base de données sous-jacente, incluant donc le langage ORDA et le langage classique.

Vous pouvez imbriquer plusieurs transactions (sous-transactions). Chaque transaction ou sous-transaction doit être annulée ou validée. A noter que si la transaction principale est annulée, toutes ses sous-transactions le sont également, même si elles avaient été validées individuellement à l'aide de la fonction `.validateTransaction()`.

Exemple

```
var $connect; $status : Object
var $person : cs.PersonsEntity
var $ds : cs.DataStore
var $choice : Text
var $error : Boolean

Case of
  :($choice=="local")
    $ds:=ds
  :($choice=="remote")
    $connect:=New object("hostname";"111.222.3.4:8044")
    $ds:=Open datastore($connect;"myRemoteDS")
End case

$ds.startTransaction()
$person:=$ds.Persons.query("lastname=:1";"Peters").first()

If($person#Null)
  $person.lastname:="Smith"
  $status:=$person.save()
End if
...
...
If($error)
  $ds.cancelTransaction()
Else
  $ds.validateTransaction()
End if
```

.stopRequestLog()

► Historique

.stopRequestLog()

Paramètres	Type	Description	-----	----	-----			Ne requiert aucun paramètre

Description

La fonction `.stopRequestLog()` stoppe tout enregistrement des requêtes ORDA sur le poste client (dans un fichier ou dans la mémoire). Elle est particulièrement utile en cas d'enregistrement dans un fichier, étant donné qu'elle ferme le document ouvert sur le disque.

Cette fonction doit être appelée sur un 4D distant, sinon elle ne fait rien. Elle est conçue à des fins de débogage dans les configurations client/serveur.

Exemple

Voir les exemples de [.startRequestLog\(\)](#).

.validateTransaction()

► Historique

.validateTransaction()

Paramètres	Type	Description	-----	----	-----			Ne requiert aucun paramètre

Description

La fonction `.validateTransaction()` valide la transaction démarrée avec `.startTransaction()` au niveau correspondant dans le datastore.

La fonction sauvegarde les modifications apportées aux données sur le datastore durant la transaction.

Vous pouvez imbriquer plusieurs transactions (sous-transactions). Si la transaction principale est annulée, toutes ses sous-transactions sont également annulées, même si elles ont été validées individuellement à l'aide de cette fonction.

Exemple

Voir l'exemple de `.startTransaction()`.

Email

La création, l'envoi et la réception d'emails dans 4D s'effectuent par l'intermédiaire des objets `Email`.

Des objets `Email` sont créés lorsque vous recevez des emails via une fonction de classe *transporter* :

- IMAP - fonctions `.getMail()` et `.getMails()` pour récupérer des emails depuis un serveur IMAP
- POP3 - fonction `.getMail()` pour récupérer un email depuis un serveur POP3.

Vous pouvez également créer un nouvel objet `Email` vide en appelant la commande 4D `New object` puis le remplir avec des propriétés d'objet `Email`.

Vous envoyez des objets `Email` à l'aide de la fonction SMTP `.send()`.

Les commandes `MAIL Convert from MIME` et `MAIL Convert to MIME` peuvent être utilisées pour convertir des objets `Email` depuis et vers des contenus MIME.

Objet Email

Les objets `Email` exposent les propriétés suivantes :

Le format des objets `Email` de 4D suit la [spécification JMAP](#).

`.attachments : Collection`

collection d'objets `4D.MailAttachment`

`.bcc : Text`

`.bcc : Object`

`.bcc : Collection`

copie carbone invisible (cci) des [adresse\(s\)](#) des destinataires cachés de l'email

`.bodyStructure : Object`

`EmailBodyPart`, i.e. la structure MIME complète du corps du message (optionnel)

`.bodyValues : Object`

`EmailBodyValue` contenant un objet pour chaque `<partID>` de `bodyStructure` (facultatif)

`.cc : Text`

`.cc : Object`

`.cc : Collection`

les [adresse\(s\) email supplémentaire\(s\)](#) des destinataires en Copie Carbone (CC) de l'email

`.comments : Text`

en-tête de commentaires supplémentaire

`.from : Text`

`.from : Object`

`.from : Collection`

[adresse\(s\)](#) d'envoi d'origine de l'email

`.headers : Collection`

collection d'objets `EmailHeader`, dans l'ordre dans lequel ils apparaissent dans le message

`.htmlBody : Text`

représentation HTML de l'email (l'encodage par défaut est UTF-8) (facultatif, SMTP uniquement)

.id : Text

ID unique du serveur IMAP

.inReplyTo : Text

identifiant(s) du ou des messages originaux auquel/auxquels le message courant est un message de réponse

.keywords : Object

un ensemble de mots-clés sous forme d'objet, où chaque nom de propriété est un mot-clé et où chaque valeur est mise à Vrai

.messageId : Texte

en-tête d'identificateur de message ("message-id")

.receivedAt : Texte

l'horodatage de l'arrivée de l'e-mail sur le serveur IMAP au format ISO 8601 UTC (ex : 2020-09-13T16:11:53Z)

.references : Collection

collection de tous les identifiants de messages de la chaîne de réponses précédente

.replyTo : Texte

.replyTo : Objet

.replyTo : Collection

adresse(s) des destinataires de réponse à l'email

.sendAt : Texte

horodatage de l'email au format ISO 8601 UTC

.sender : Texte

.sender : Objet

.sender : Collection

adresse(s) source(s) de l'email

.size : Integer

taille (exprimée en octets) de l'objet Email retourné par le serveur IMAP

.subject : Texte

description du sujet

.textBody : Texte

représentation en texte brut de l'email (l'encodage par défaut est UTF-8) (facultatif, SMTP uniquement)

.to : Texte

.to : Objet

.to : Collection

le ou les adresse(s) des destinataires principaux de l'email

Adresses Email

Toutes les propriétés qui contiennent des adresses email (`from` , `cc` , `bcc` , `to` , `sender` , `replyTo`) acceptent des valeurs de type texte, objet ou collection.

Text

- une adresse uniquement : "somebody@domain.com"
- un nom+adresse : "Somebody somebody@domain.com"

- combinaison de plusieurs adresses : "Somebody somebody@domain.com,me@home.org"

Object

Un objet avec deux propriétés :

Propriété	Type	Description
name	Text	Nom à afficher (peut être null)
email	Text	Adresse email

Collection

Une collection d'objets adresse.

Traitement du body

Les propriétés `textBody` et `htmlBody` sont utilisées uniquement par la fonction `SMTP.send()` pour permettre d'envoyer des emails simples. Lorsque les deux propriété sont remplies, le type MIME content-type multipart/alternative est utilisé. Le client email doit alors reconnaître la partie multipart/alternative et afficher la partie texte ou html nécessaire.

`bodyStructure` et `bodyValues` sont utilisés par le `SMTP` lorsque l'`objet Email` est construit depuis un document MIME, e.g. généré par la commande `MAIL Convert from MIME`. Dans ce cas, les deux propriétés `bodyStructure` et `bodyValues` doivent être passées ensemble, et il est déconseillé d'utiliser `textBody` et `htmlBody`.

Exemple d'objets `bodyStructure` et `bodyValues`

```
"bodyStructure": {
  "type": "multipart/mixed",
  "subParts": [
    {
      "partId": "p0001",
      "type": "text/plain"
    },
    {
      "partId": "p0002",
      "type": "text/html"
    }
  ],
  "bodyValues": {
    "p0001": {
      "value": "I have the most brilliant plan. Let me tell you all about it."
    },
    "p0002": {
      "value": "<!DOCTYPE html><html><head><title></title><style type=\"text/css\">div{font-size:16px}</st
    }
  }
}
```

.attachments

`.attachments` : Collection

Description

La propriété `.attachments` contient une collection d'objets `4D.MailAttachment`.

Les objets Attachment (pièce jointe) sont créés à l'aide de la commande `MAIL New attachment`. Les objets Attachment ont des propriétés et fonctions spécifiques.

.bcc

.bcc : Text
.bcc : Object
.bcc : Collection

Description

La propriété `.bcc` contient la copie carbone invisible (cci) des [adresse\(s\)](#) des destinataires cachés de l'email.

.bodyStructure

.bodyStructure : Object

Description

La propriété `.bodyValues` stocke l'objet *EmailBodyValue* contenant un objet pour chaque `<partID>` de `bodyStructure` (facultatif). Voir section [Traitement du body](#).

L'objet `.bodyStructure` contient les propriété suivantes :

Propriété	Type	Valeur
partID	Text	Identifie la partie de manière unique dans l'email
type	Text	(obligatoire) Valeur du champ d'en-tête Content-Type de la partie
charset	Text	Valeur du paramètre charset du champ d'en-tête Content-Type
encoding	Text	Si <code>isEncodingProblem=true</code> , la valeur de Content-Transfer-Encoding est ajoutée (par défaut indéfini)
disposition	Text	Valeur du champ d'en-tête Content-Disposition de la partie
language	Collection de textes	Liste de balises de langage, telles que définies dans la RFC3282 , dans le champ d'en-tête Content-Language de la partie, le cas échéant.
location	Text	URI, tel que défini dans la RFC2557 , dans le champ d'en-tête Content-Location de la partie, le cas échéant.
subParts	Collection d'objets	Parties du corps de chaque enfant (collection d'objets <i>EmailBodyPart</i>)
headers	Collection d'objets	Liste de tous les champs d'en-tête de la partie, dans leur ordre d'apparition de l'email (collection d'objets <i>EmailHeader</i> voir propriété <code>headers</code>)

.bodyValues

.bodyValues : Object

Description

La propriété `.htmlBody` contient la représentation HTML de l'email (l'encodage par défaut est UTF-8) (facultatif, SMTP uniquement). Voir section [Traitement du body](#).

L'objet `.bodyValues` contient les propriété suivantes :

Propriété	Type	Valeur
<code>partID.value</code>	text	Valeur de la partie body
<code>partID.isEncodingProblem</code>	boolean	Vrai si des sections malformées sont identifiées durant le décodage du charset, si le charset est inconnu, ou si le content transfer-encoding est inconnu. Faux par défaut

.CC

.cc : Text
.cc : Object
.cc : Collection

Description

La propriété `.cc` contient les [adresse\(s\) email supplémentaire\(s\)](#) des destinataires en Copie Carbone (CC) de l'email.

.comments

.comments : Text

Description

La propriété `.comments` contient un en-tête de commentaires supplémentaire.

Les commentaires n'apparaissent que dans la zone d'en-tête du message (le body du message reste inchangé).

Pour les exigences propres au formatage, veuillez consulter la [RFC#5322](#).

.from

.from : Text
.from : Object
.from : Collection

Description

La propriété `.from` contient les [adresse\(s\)](#) d'envoi d'origine de l'email.

Chaque email envoyé comporte à la fois les adresses du `sender` et du `from` :

- le domaine `sender` correspond à ce que le serveur de réception d'email obtient à l'ouverture de la session,
- l'adresse `from` correspond à ce que le(s) destinataire(s) visualise(nt).

Pour mieux livrer l'email, il est recommandé d'utiliser les mêmes adresses pour `from` et `sender`.

.headers

.headers : Collection

Description

La propriété `.headers` contient une collection d'objets `EmailHeader`, dans l'ordre dans lequel ils apparaissent dans le message. Cette propriété permet aux utilisateurs d'ajouter des en-têtes extended (enregistrés) ou des en-têtes user-defined (non enregistrés, commençant par "X").

Si une propriété d'objet `EmailHeader` définit un en-tête tel que "from" ou "cc" qui est déjà défini comme propriété au niveau du mail, la propriété `EmailHeader` est ignorée.

Chaque objet de la collection de headers peut contenir les propriétés suivantes :

Propriété	Type	Valeur
<code>[] .name</code>	Texte	(obligatoire) Nom du champ en-tête, tel que défini dans la RFC#5322 . S'il est null ou indéfini, le champ en-tête n'est pas ajouté à l'en-tête MIME.
<code>[] .value</code>	Texte	Valeur du champ d'en-tête telle que définie dans la RFC#5322

.htmlBody

.htmlBody : Text

Description

La propriété `.textBody` contient la représentation en texte brut de l'email (l'encodage par défaut est UTF-8) (facultatif, SMTP uniquement). Voir section [Traitement du body](#).

.id

.id : Text

Description

[IMAP transporter](#) uniquement.

La propriété `.id` contient l' ID unique du serveur IMAP.

.inReplyTo

.inReplyTo : Text

Description

La propriété `.inReplyTo` contient le ou les identifiant(s) du ou des messages originaux auquel/auxquels le message courant est un message de réponse.

Pour les exigences propres au formatage, veuillez consulter la [RFC#5322](#).

.keywords

.keywords : Object

Description

La propriété `.keywords` contient un ensemble de mots-clés sous forme d'objet, où chaque nom de propriété est un mot-clé et où chaque valeur est mise à Vrai.

Cette propriété est l'en-tête "keywords" (voir la [RFC#4021](#)).

Propriété	Type	Valeur
<code>. <keyword></code>	boolean	Mot-clé à définir à définir (la valeur doit être mise à vrai)

Mots-clés réservés :

- `$draft` - Indique qu'un message est un brouillon
- `$seen` - Indique qu'un message a été lu
- `$flagged` - Indique qu'un message nécessite une attention particulière (ex : Urgent)
- `$answered` - Indique qu'un message a reçu une réponse
- `$deleted` - Indique un message à supprimer

Exemple

```
$mail.keywords["$flagged"]:=True  
$mail.keywords["4d"]:=True
```

.messageId

.messageId : Texte

Description

La propriété `.messageId` contient un en-tête d'identificateur de message ("message-id").

Cet en-tête est généralement "desChiffresOuDesLettres@nomdededomaine", par exemple "abcdef.123456@4d.com". Cet identifiant unique est notamment utilisé sur les forums ou les listes de diffusion publiques. En général, les serveurs de messagerie ajoutent automatiquement cet en-tête aux messages qu'ils envoient.

.receivedAt

.receivedAt : Texte

Description

[IMAP transporter](#) uniquement.

La propriété `.receivedAt` contient l'horodatage de l'arrivée de l'e-mail sur le serveur IMAP au format ISO 8601 UTC (ex : 2020-09-13T16:11:53Z).

.references

.references : Collection

Description

La propriété `.references` contient la collection de tous les identifiants de messages de la chaîne de réponses précédente.

Pour les exigences propres au formatage, veuillez consulter la [RFC#5322](#).

.replyTo

.replyTo : Texte

.replyTo : Objet

.replyTo : Collection

Description

La propriété `.replyTo` contient les [adresse\(s\)](#) des destinataires de réponse à l'email.

.sendAt

.sendAt : Texte

Description

La propriété `.sendAt` contient l' horodatage de l'email au format ISO 8601 UTC.

.sender

.sender : Texte

.sender : Objet

.sender : Collection

Description

La propriété `.sender` contient les [adresse\(s\) source\(s\)](#) de l'email.

Chaque email envoyé comporte à la fois les adresses du sender et du [from](#) :

- le domaine sender correspond à ce que le serveur de réception d'email obtient à l'ouverture de la session,
- l'adresse from correspond à ce que le(s) destinataire(s) visualise(nt).

Pour mieux livrer l'email, il est recommandé d'utiliser les mêmes adresses pour from et sender.

`.size`

`.size` : Integer

Description

[IMAP transporter](#) uniquement.

La propriété `.size` contient la taille (exprimée en octets) de l'objet Email retourné par le serveur IMAP.

`.subject`

`.subject` : Texte

Description

La propriété `.subject` contient la description du sujet.

`.textBody`

`.textBody` : Texte

Description

La propriété `.bodyStructure` contient l'objet [*EmailBodyPart*](#), i.e. la structure MIME complète du corps du message (optionnel). Voir section [Traitement du body](#).

`.to`

`.to` : Texte

`.to` : Objet

`.to` : Collection

Description

La propriété `.to` contient le ou les [adresse\(s\)](#) des destinataires principaux de l'email.

MAIL Convert from MIME

► Historique

MAIL Convert from MIME(*mime* : Blob) : Object

MAIL Convert from MIME(*mime* : Text) : Object

Paramètres	Type		Description
<i>mime</i>	Blob, Text	->	Email en MIME
Résultat	Object	<-	Objet email

Description

La commande `MAIL Convert from MIME` convertit un document MIME en un objet email valide.

Le format des objets Email de 4D suit la [spécification JMAP](#).

Passez dans *mime* un document MIME valide à convertir. Il peut être fourni par tout type de serveur ou d'application de messagerie. Vous pouvez passer un BLOB ou un texte dans le paramètre *mime*. Si le MIME provient d'un fichier, il est recommandé d'utiliser un paramètre BLOB pour éviter les problèmes liés aux conversions de charset et de retours à la ligne.

Objet retourné

Objet email.

Exemple 1

Vous souhaitez charger un template mail enregistré au format MIME dans un document texte et l'envoyer par email :

```
var $mime: Blob
var $mail;$server;$transporter;$status: Object

$mime:=File("/PACKAGE/Mails/templateMail.txt").getContent()

$mail:=MAIL Convert from MIME($mime)
$mail.to:="smith@mail.com"
$mail.subject:="Hello world"

$server:=New object
$server.host:="smtp.gmail.com"
$server.port:=465
$server.user:="test@gmail.com"
$server.password:="XXXX"

$transporter:=SMTP New transporter($server)
$status:=$transporter.send($mail)
```

Exemple 2

Dans cet exemple, vous envoyez directement un document 4D Write Pro contenant des images :

```

var $mime: Blob
var $email;$server;$transporter;$status: Object

// Export Mime du document 4D Write Pro
WP EXPORT VARIABLE(WParea;$mime;wk mime html)

// convertir la variable Mime de 4D Write Pro en objet email
$email:=MAIL Convert from MIME($mime)

// Remplir les en-têtes de l'objet email
$email.subject:="4D Write Pro HTML body"
$email.from:="YourEmail@gmail.com"
$email.to:="RecipientEmail@mail.com"

$server:=New object
$server.host:="smtp.gmail.com"
$server.port:=465
$server.user:="YourEmail@gmail.com"
$server.password:="XXXX"

$transporter:=SMTP New transporter($server)
$status:=$transporter.send($email)

```

MAIL Convert to MIME

► Historique

MAIL Convert to MIME(*mail* : Object { ; *options* : Object }) : Text

Paramètres	Type		Description
mail	Object	->	Objet email
options	Object	->	Options d'encodage et de charset du mail
Résultat	Text	<-	Objet email converti en MIME

Description

La commande `MAIL Convert to MIME` convertit un objet email en un texte MIME. Cette commande est appelée en interne par `SMTP_transporter.send()` pour formater l'objet email avant de l'envoyer. Elle peut être utilisée pour analyser le format MIME de l'objet.

Dans *mail*, passez les éléments du contenu et de la structure de l'email à convertir. Cela inclut des informations telles que les adresses e-mail (expéditeur et destinataire(s)), le contenu de l'e-mail lui-même et son type d'affichage.

Le format des objets Email de 4D suit la [spécification JMAP](#).

Dans *options*, vous pouvez configurer l'encodage et le charset du mail. Les propriétés suivantes sont disponibles :

Propriété	Type	Description															
headerCharset	Text	<p>Charset et encodage utilisés pour les parties de mail suivantes : le sujet, les noms de fichiers joints et le nom du mail. Valeurs possibles :</p> <table border="1"> <thead> <tr> <th>Constante</th> <th>Valeur</th> <th>Commentaire</th> </tr> </thead> <tbody> <tr> <td>mail mode ISO2022JP</td> <td>US-ASCII_ISO-2022-JP_UTF8_QP</td> <td> <ul style="list-style-type: none"> <i>headerCharset</i> : US-ASCII si possible, japonais (ISO-2022-JP) & Quoted-printable si possible, sinon UTF-8 & Quoted-printable <i>bodyCharset</i> : US-ASCII si possible, japonais (ISO-2022-JP) et 7 bits si possible, sinon UTF-8 et Quoted-printable </td> </tr> <tr> <td>mail mode ISO88591</td> <td>ISO-8859-1</td> <td> <ul style="list-style-type: none"> <i>headerCharset</i>: ISO-8859-1 & Quoted-printable <i>bodyCharset</i>: ISO-8859-1 & 8-bit </td> </tr> <tr> <td>mail mode UTF8</td> <td>US-ASCII_UTF8_QP</td> <td><i>headerCharset</i> & <i>bodyCharset</i> : US-ASCII si possible, sinon UTF-8 & Quoted-printable (valeur par défaut)</td> </tr> <tr> <td>mail mode UTF8 in base64</td> <td>US-ASCII_UTF8_B64</td> <td><i>headerCharset</i> &<i>bodyCharset</i> : US-ASCII si possible, sinon UTF-8 & base64</td> </tr> </tbody> </table>	Constante	Valeur	Commentaire	mail mode ISO2022JP	US-ASCII_ISO-2022-JP_UTF8_QP	<ul style="list-style-type: none"> <i>headerCharset</i> : US-ASCII si possible, japonais (ISO-2022-JP) & Quoted-printable si possible, sinon UTF-8 & Quoted-printable <i>bodyCharset</i> : US-ASCII si possible, japonais (ISO-2022-JP) et 7 bits si possible, sinon UTF-8 et Quoted-printable 	mail mode ISO88591	ISO-8859-1	<ul style="list-style-type: none"> <i>headerCharset</i>: ISO-8859-1 & Quoted-printable <i>bodyCharset</i>: ISO-8859-1 & 8-bit 	mail mode UTF8	US-ASCII_UTF8_QP	<i>headerCharset</i> & <i>bodyCharset</i> : US-ASCII si possible, sinon UTF-8 & Quoted-printable (valeur par défaut)	mail mode UTF8 in base64	US-ASCII_UTF8_B64	<i>headerCharset</i> & <i>bodyCharset</i> : US-ASCII si possible, sinon UTF-8 & base64
Constante	Valeur	Commentaire															
mail mode ISO2022JP	US-ASCII_ISO-2022-JP_UTF8_QP	<ul style="list-style-type: none"> <i>headerCharset</i> : US-ASCII si possible, japonais (ISO-2022-JP) & Quoted-printable si possible, sinon UTF-8 & Quoted-printable <i>bodyCharset</i> : US-ASCII si possible, japonais (ISO-2022-JP) et 7 bits si possible, sinon UTF-8 et Quoted-printable 															
mail mode ISO88591	ISO-8859-1	<ul style="list-style-type: none"> <i>headerCharset</i>: ISO-8859-1 & Quoted-printable <i>bodyCharset</i>: ISO-8859-1 & 8-bit 															
mail mode UTF8	US-ASCII_UTF8_QP	<i>headerCharset</i> & <i>bodyCharset</i> : US-ASCII si possible, sinon UTF-8 & Quoted-printable (valeur par défaut)															
mail mode UTF8 in base64	US-ASCII_UTF8_B64	<i>headerCharset</i> & <i>bodyCharset</i> : US-ASCII si possible, sinon UTF-8 & base64															
bodyCharset	Text	Charset et encodage utilisés pour le contenu html et le texte du body du mail. Valeurs possibles : Identiques à celles de headerCharset (voir ci-dessus)															

Si le paramètre *options* est omis, la configuration mail mode UTF8 est utilisée pour les parties en-tête et corps.

Exemple

```

var $mail: Object
var $mime: Text
$mail:=New object

// Création d'un mail
$mail.from:="tsales@massmarket.com"
$mail.subject:="Terrific Sale! This week only!"
$mail.textBody:="Text format email"
$mail.htmlBody:="<html><body>HTML format email</body></html>"
$mail.to:=New collection
$mail.to.push(New object ("email";"noreply@4d.com"))
$mail.to.push(New object ("email";"test@4d.com"))

// transform the mail object in MIME
$mime:=MAIL Convert to MIME($mail)

// Contents of $mime:
// MIME-Version: 1.0
// Date: Thu, 11 Oct 2018 15:42:25 GMT
// Message-ID: <7CA5D25B2B5E0047A36F2E8CB30362E2>
// Sender: tsales@massmarket.com
// From: tsales@massmarket.com
// To: noreply@4d.com
// To: test@4d.com
// Content-Type: multipart/alternative; boundary="E0AE5773D5E95245BBBBD80DD0687E218"
// Subject: Terrific Sale! This week only!
//
// --E0AE5773D5E95245BBBBD80DD0687E218
// Content-Type: text/plain; charset="UTF-8"
// Content-Transfer-Encoding: quoted-printable
//
// Text format email
// --E0AE5773D5E95245BBBBD80DD0687E218
// Content-Type: text/html; charset="UTF-8"
// Content-Transfer-Encoding: quoted-printable
//
// <html><body>HTML format email</body></html>
// --E0AE5773D5E95245BBBBD80DD0687E218--

```

Entity

Une [entity](#) est une instance d'une [Dataclass](#), tel un enregistrement de la table correspondant à la dataclass contenue dans son datastore associé. Elle contient les mêmes attributs que la dataclass ainsi que les valeurs des données et des propriétés et fonctions spécifiques.

Sommaire

<code>.attributeName : any</code>	stocke la valeur de l'attribut pour l'entité
<code>.clone() : 4D.Entity</code>	crée en mémoire une nouvelle entité référençant le même enregistrement que l'entité d'origine
<code>.diff(entityToCompare : 4D.Entity { ; attributesToCompare : Collection }) : Collection</code>	compare le contenu de deux entités et retourne leurs différences
<code>.drop({ mode : Integer }) : Object</code>	supprime les données de l'entité de la table associée à la dataclass
<code>.first(): 4D.Entity</code>	retourne une référence vers l'entité en première position dans l'entity selection à laquelle appartient l'entité
<code>.fromObject(filler : Object)</code>	remplit les attributs de l'entité avec le contenu de <i>filler</i>
<code>.getDataClass() : 4D.DataClass</code>	retourne la dataclass de l'entité
<code>.getKey({ mode : Integer }) : Text</code> <code>.getKey({ mode : Integer }) : Integer</code>	retourne la valeur de la clé primaire de l'entité
<code>.getRemoteContextAttributes() : Text</code>	retourne des informations relatives au contexte d'optimisation utilisé par l'entity
<code>.getSelection(): 4D.EntitySelection</code>	retourne l'entity selection à laquelle appartient l'entité
<code>.getStamp() : Integer</code>	retourne la valeur courante du stamp de l'entité
<code>.indexOf({ entitySelection : 4D.EntitySelection }) : Integer</code>	retourne la position de l'entité dans une entity selection
<code>.isNew() : Boolean</code>	renvoie Vrai si l'entité à laquelle elle est appliquée vient d'être créée et n'a pas encore été enregistrée dans le datastore
<code>.last() : 4D.Entity</code>	renvoie une référence vers l'entité en dernière position dans l'entity selection à laquelle l'entité appartient
<code>.lock({ mode : Integer }) : Object</code>	pose un verrou pessimiste sur l'enregistrement référencé par l'entité

`.next() : 4D.Entity`

retourne une référence sur l'entité suivante dans l'entity selection à laquelle appartient l'entité

`.previous() : 4D.Entity`

retourne une référence sur l'entité précédente dans l'entity selection à laquelle appartient l'entité

`.reload() : Object`

recharge en mémoire le contenu de l'entité

`.save({ mode : Integer }) : Object`

sauvegarde les modifications effectuées sur l'entité

`.toObject() : Object`

`.toObject(filterString : Text { ; options : Integer }) : Object`

`.toObject(filterCol : Collection { ; options : Integer }) : Object`

retourne un objet construit à partir de l'entité

`.touched() : Boolean`

indique si un attribut de l'entité a été modifié ou non depuis que l'entité a été chargée en mémoire ou sauvegardée

`.touchedAttributes() : Collection`

retourne les noms des attributs qui ont été modifiés depuis que l'entité a été chargée en mémoire

`.unlock() : Object`

supprime le verrou pessimiste posé sur l'enregistrement correspondant à l'entité

.attributeName

► Historique

`.attributeName` : any

Description

Tout attribut de dataclass est disponible en tant que propriété des entités de la dataclass, et qui stocke la valeur de l'attribut pour l'entité.

Les attributs de dataclass peuvent également être obtenus en utilisant la syntaxe alternative avec [].

Le type de valeur de l'attribut dépend du type (`kind`) d'attribut (relation ou stockage) :

- Si le type de `attributeName` est storage : `.attributeName` retourne une valeur du même type que `attributeName`.
- Si le type de `attributeName` est relatedEntity : `.attributeName` retourne une entité reliée. Les valeurs de l'entité liée sont directement disponibles par le biais des propriétés en cascade, par exemple "myEntity.employer.employees[0].lastname".
- Si le type de `attributeName` est relatedEntities : `.attributeName` retourne une nouvelle entity selection d'entités liées. Les doublons sont supprimés (une entity selection non ordonnée est retournée).

Exemple

```
var $myEntity : cs.EmployeeEntity
$myEntity:=ds.Employee.new() //Créer une nouvelle entity
$myEntity.name:="Dupont" //assigner 'Dupont' à l'attribut 'name'
$myEntity.firstname:="John" //assigner 'John' à l'attribut 'firstname'
$myEntity.save() //sauvegarder l'entity
```

.clone()

► Historique

.clone() : 4D.Entity

Paramètres	Type		Description
Résultat	4D.Entity	<-	Nouvelle entité référençant l'enregistrement

Description

La fonction `.clone()` crée en mémoire une nouvelle entité référençant le même enregistrement que l'entité d'origine. Cette fonction vous permet de mettre à jour des entités séparément.

A noter que toute modification apportée aux entités n'est stockée dans l'enregistrement référencé qu'au moment où la fonction `.save()` est exécutée.

Cette fonction ne peut être utilisée qu'avec des entités déjà enregistrées dans la base de données. Elle ne peut pas être appelée sur une entité nouvellement créée (pour laquelle `.isNew()` retourne True).

Exemple

```
var $emp; $empCloned : cs.EmployeeEntity
$emp:=ds.Employee.get(672)
$empCloned:=$emp.clone()

$emp.lastName:="Smith" //Les mises à jour effectuées sur $emp ne le sont pas sur $empCloned
```

.diff()

► Historique

.diff(*entityToCompare* : 4D.Entity { ; *attributesToCompare* : Collection }) : Collection

Paramètres	Type		Description
<i>entityToCompare</i>	4D.Entity	->	Entité à comparer à l'entité d'origine
<i>attributesToCompare</i>	Collection	->	Noms des attributs à comparer
Résultat	Collection	<-	Différences entre les entités

Description

La fonction `.diff()` compare le contenu de deux entités et retourne leurs différences.

Dans le paramètre *entityToCompare*, passez l'entité à comparer à l'entité d'origine.

Dans le paramètre *attributesToCompare*, vous pouvez désigner les attributs spécifiques à comparer. Si le paramètre est passé, la comparaison est effectuée uniquement sur les attributs spécifiés. S'il est omis, toutes les différences entre les entités sont retournées.

Les différences sont retournées sous forme de collection d'objets dont les propriétés sont :

Nom de propriété	Type	Description
attributeName	String	Nom de l'attribut
value	Dépend du type d'attribut	Valeur de l'attribut dans l'entité d'origine
otherValue	Dépend du type d'attribut	Valeur de l'attribut dans <i>entityToCompare</i>

Seuls les attributs dont les valeurs diffèrent sont inclus dans la collection. Si aucune différence n'est trouvée, `.diff()` retourne une collection vide.

La fonction s'applique aux attributs dont le `kind` est `storage` ou `relatedEntity`. Dans le cas où une entité liée a été mise à jour (c'est-à-dire la clé étrangère), le nom de l'entité liée et de sa clé primaire sont retournés comme propriétés `attributeName` (`value` et `otherValue` sont vides pour l'entité liée).

Si l'une des entités comparées vaut `Null`, une erreur est retournée.

Exemple 1

```
var $diff1; $diff2 : Collection
employee:=ds.Employee.query("ID=1001").first()
$clone:=employee.clone()
employee.firstName:="MARIE"
employee.lastName:="SOPHIE"
employee.salary:=500
$diff1:=$clone.diff(employee) // Toutes les différences sont retournées
$diff2:=$clone.diff(employee;New collection"firstName";"lastName"))
// Seules les différences relevées sur firstName et lastName sont retournées
```

\$diff1:

```
[  
  {  
    "attributeName": "firstName",  
    "value": "Natasha",  
    "otherValue": "MARIE"  
  },  
  {  
    "attributeName": "lastName",  
    "value": "Locke",  
    "otherValue": "SOPHIE"  
  },  
  {  
    "attributeName": "salary",  
    "value": 66600,  
    "otherValue": 500  
  }  
]
```

\$diff2:

```
[  
  {  
    "attributeName": "firstName",  
    "value": "Natasha",  
    "otherValue": "MARIE"  
  },  
  {  
    "attributeName": "lastName",  
    "value": "Locke",  
    "otherValue": "SOPHIE"  
  }  
]
```

Exemple 2

```

var vCompareResult1; vCompareResult2; vCompareResult3; $attributesToInspect : Collection
vCompareResult1:=New collection
vCompareResult2:=New collection
vCompareResult3:=New collection
$attributesToInspect:=New collection

$e1:=ds.Employee.get(636)
$e2:=ds.Employee.get(636)

$e1.firstName:=$e1.firstName+" update"
$e1.lastName:=$e1.lastName+" update"

$c:=ds.Company.get(117)
$e1.employer:=$c
$e2.salary:=100

$attributesToInspect.push("firstName")
$attributesToInspect.push("lastName")

vCompareResult1:=$e1.diff($e2)
vCompareResult2:=$e1.diff($e2;$attributesToInspect)
vCompareResult3:=$e1.diff($e2;$e1.touchedAttributes())

```

vCompareResult1 (toutes les différences sont retournées) :

```
[
{
  "attributeName": "firstName",
  "value": "Karla update",
  "otherValue": "Karla"
},
{
  "attributeName": "lastName",
  "value": "Marrero update",
  "otherValue": "Marrero"
},
{
  "attributeName": "salary",
  "value": 33500,
  "otherValue": 100
},
{
  "attributeName": "employerID",
  "value": 117,
  "otherValue": 118
},
{
  "attributeName": "employer",
  "value": "[object Entity]">// Entity 117 de Company
  "otherValue": "[object Entity]"// Entity 118 de Company
}
]
```

vCompareResult2 (seules les différences sur \$attributesToInspect sont retournées)

```
[
  {
    "attributeName": "firstName",
    "value": "Karla update",
    "otherValue": "Karla"
  },
  {
    "attributeName": "lastName",
    "value": "Marrero update",
    "otherValue": "Marrero"
  }
]
```

vCompareResult3 (seules les différences sur les attributs touchés de \$e1 sont retournés)

```
[
  {
    "attributeName": "firstName",
    "value": "Karla update",
    "otherValue": "Karla"
  },
  {
    "attributeName": "lastName",
    "value": "Marrero update",
    "otherValue": "Marrero"
  },
  {
    "attributeName": "employerID",
    "value": 117,
    "otherValue": 118
  },
  {
    "attributeName": "employer",
    "value": "[object Entity]",// Entity 117 de Company
    "otherValue": "[object Entity]"// Entity 118 de Company
  }
]
```

.drop()

► Historique

.drop({mode : Integer}) : Object

Paramètres	Type		Description
mode	Integer	->	dk force drop if stamp changed : Force la suppression même si le marqueur interne a changé
Résultat	Object	<-	Résultat de l'opération de suppression

Description

La fonction `.drop()` supprime les données de l'entité de la table associée à la dataclass. A noter que l'entité reste en mémoire.

Dans une application multiprocess ou multi-utilisateurs, la fonction `.drop()` est exécutée en mode "[verrouillage optimiste](#)" dans lequel un marqueur de verrouillage interne est automatiquement incrémenté à chaque sauvegarde de l'enregistrement.

Par défaut, lorsque le paramètre `mode` est omis, la méthode retourne une erreur (cf. ci-dessous) si l'entité a été

modifiée (i.e. le marqueur interne a changé) entre-temps par un autre process ou utilisateur.

Sinon, vous pouvez passer l'option `dk force drop if stamp changed` dans le paramètre *mode* : dans ce cas, l'entité est supprimée même si la valeur du marqueur interne est différente (si la clé primaire est identique).

Résultat

L'objet retourné par `.drop()` contient les propriétés suivantes :

Propriété		Type	Description
success		boolean	vrai si l'action de suppression a été effectuée avec succès, sinon Faux.
			<i>Disponible uniquement en cas d'erreur :</i>
status(*)		number	Code d'erreur, voir ci-dessous
statusText(*)		text	Description de l'erreur, voir ci-dessous
			<i>Disponible uniquement en cas d'erreur de verrouillage pessimiste :</i>
LockKindText		text	"Locked by record"
lockInfo		object	Information sur l'origine du verrouillage
	task_id	number	Id du process
	user_name	text	Nom d'utilisateur de la session sur la machine
	user4d_alias	text	Alias utilisateur si défini avec <code>SET USER ALIAS</code> , sinon le nom d'utilisateur dans le répertoire de la base 4D
	host_name	text	Nom de la machine
	task_name	text	Nom du process
	client_version	text	
			<i>Disponible uniquement en cas d'erreur critique (clé primaire dupliquée, disque plein..) :</i>
errors		collection of objects	
	message	text	Message d'erreur
	component signature	text	signature du composant interne (e.g. "dmbg" pour le composant de base de données)
	errCode	number	Code d'erreur

(*) Les valeurs suivantes peuvent être retournées dans les propriétés *status* et *statusText* de l'objet *Résultat* en cas d'erreur :

Constante	Valeur	Commentaire
dk status entity does not exist anymore	5	L'entité n'existe plus dans les données. Cette erreur peut se produire dans les cas suivants : <ul style="list-style-type: none"> l'entité a été supprimée (le stamp est modifié et l'espace mémoire est libéré) l'entité a été supprimée et remplacée par une autre avec une clé primaire différente (le stamp est modifié et une nouvelle entité occupe l'espace mémoire). Avec drop(), cette erreur peut être retournée lorsque l'option dk force drop if stamp changed est utilisée. Avec lock(), cette erreur peut être retournée lorsque l'option dk reload if stamp changed est utilisée statusText associé : "Entity does not exist anymore"
dk status locked	3	L'entité est verrouillée par un verrou pessimiste. statusText associé : "Already locked"
dk status serious error	4	Une erreur critique peut être une erreur de bas niveau de la base de données (ex. clé dupliquée), une erreur matérielle, etc. statusText associé : "Other error"
dk status stamp has changed	2	La valeur du marqueur interne (stamp) de l'entité ne correspond pas à celle de l'entité stockée dans les données (verrouillage optimiste). <ul style="list-style-type: none"> avec .save() : erreur uniquement si l'option dk auto merge n'est pas utilisée avec .drop() : erreur uniquement si l'option dk force drop if stamp changed n'est pas utilisée avec .lock() : erreur uniquement si l'option dk reload if stamp changed n'est pas utilisée statusText associé : "Stamp has changed"

Exemple 1

Exemple sans option dk force drop if stamp changed :

```

var $employees : cs.EmployeeSelection
var $employee : cs.EmployeeEntity
var $status : Object
$employees:=ds.Employee.query("lastName=:1";"Smith")
$employee:=$employees.first()
$status:=$employee.drop()
Case of
    :($status.success)
        ALERT("You have dropped "+$employee.firstName+" "+$employee.lastName) //L'entité supprimée reste
    :($status.status=dk status stamp has changed)
        ALERT($status.statusText)
End case

```

Exemple 2

Même exemple avec l'option dk force drop if stamp changed :

```

var $employees : cs.EmployeeSelection
var $employee : cs.EmployeeEntity
var $status : Object
$employees:=ds.Employee.query("lastName=:1";"Smith")
$employee:=$employees.first()
$status:=$employee.drop(dk force drop if stamp changed)
Case of
    :($status.success)
        ALERT("You have dropped "+$employee.firstName+" "+$employee.lastName) //L'entité supprimée reste
    :($status.status=dk status entity does not exist anymore)
        ALERT($status.statusText)
End case

```

.first()

► Historique

.first(): 4D.Entity

Paramètres	Type		Description
Résultat	4D.Entity	<-	Référence à la première entité de l'entity selection (Null si non trouvée)

Description

La fonction `.first()` retourne une référence vers l'entité en première position dans l'entity selection à laquelle appartient l'entité.

Si l'entité n'appartient à aucune entity selection (i.e. `.getSelection()` retourne Null), la fonction renvoie une valeur Null.

Exemple

```

var $employees : cs.EmployeeSelection
var $employee; $firstEmployee : cs.EmployeeEntity
$employees:=ds.Employee.query("lastName = :1";"H@") //Cette entity selection contient 3 entités
$employee:=$employees[2]
$firstEmployee:=$employee.first() // $firstEmployee est la première entité de l'entity selection $employ

```

.fromObject()

► Historique

.fromObject(*filler* : Object)

Paramètres	Type		Description
<i>filler</i>	Object	->	Objet avec lequel remplir l'entité

Description

La fonction `.fromObject()` remplit les attributs de l'entité avec le contenu de *filler*.

Cette fonction modifie l'entity d'origine.

La correspondance entre l'objet et l'entité est établie à partir des noms de propriétés/d'attributs :

- Si une propriété de l'objet n'existe pas dans la dataclass, elle est ignorée.
- Les types de données doivent être équivalents. S'il existe une différence de type entre l'objet et la dataclass, 4D

essaie de convertir les données lorsque cela est possible (voir [Conversion des types de données](#)), sinon l'attribut demeure intact.

- La clé primaire peut être donnée telle quelle ou avec une propriété "__KEY" (remplie avec la valeur de la clé primaire). Si elle n'existe pas déjà dans la dataclass, l'entité est créée avec la valeur donnée lorsque `.save()` est appelé. Si la clé primaire n'est pas fournie, l'entité est créée et la valeur de la clé primaire est affectée en fonction des règles de la base de données. L'auto-incrémentation n'est calculée que si la clé primaire est nulle.

`filler` peut contenir une related entity dans les conditions suivantes :

- `filler` contient lui-même la clé étrangère, ou
- `filler` contient une propriété de type objet qui a le même nom que l'entité relative, contenant une seule propriété nommée "__KEY".
- si l'entité relative n'existe pas, elle est ignorée.

Exemple

Avec l'objet \$o suivant :

```
{  
    "firstName": "Mary",  
    "lastName": "Smith",  
    "salary": 36500,  
    "birthDate": "1958-10-27T00:00:00.000Z",  
    "woman": true,  
    "managerID": 411, // relatedEntity fournie avec clé primaire  
    "employerID": 20 // relatedEntity fournie avec clé primaire  
}
```

Le code suivant créera une entité avec les entités relatives manager et employeur.

```
var $o : Object  
var $entity : cs.EmpEntity  
$entity:=ds.Emp.new()  
$entity.fromObject($o)  
$entity.save()
```

Vous pouvez également utiliser une entité relative fournie sous forme d'objet :

```
{  
    "firstName": "Marie",  
    "lastName": "Lechat",  
    "salary": 68400,  
    "birthDate": "1971-09-03T00:00:00.000Z",  
    "woman": false,  
    "employer": { // relatedEntity fournie sous forme d'objet  
        "__KEY": "21"  
    },  
    "manager": { // relatedEntity fournie sous forme d'objet  
        "__KEY": "411"  
    }  
}
```

`.getDataClass()`

► Historique

`.getDataClass()` : 4D.DataClass

Paramètres	Type		Description
Résultat	4D.DataClass	<-	Dataclass à laquelle appartient l'entité

Description

La fonction `.getDataClass()` retourne la dataclass de l'entité. Cette fonction est utile pour l'écriture du code générique.

Exemple

Le code générique suivant duplique toute entité :

```
//méthode duplicate_entity
//duplicate_entity($entity)

#DECLARE($entity : 4D.Entity)
var $entityNew : 4D.Entity
var $status : Object

$entityNew:=$entity.getDataClass().new() //crée une nouvelle entité dans la dataclass parente
$entityNew.fromObject($entity.toObject()) //lire tous les attributs
$entityNew[$entity.getDataClass().getInfo().primaryKey]:=Null //réinitialise la clé primaire
$status:=$entityNew.save() //sauvegarde l'entité dupliquée
```

.getKey()

► Historique

`.getKey({ mode : Integer }) : Text`
`.getKey({ mode : Integer }) : Integer`

Paramètres	Type		Description
mode	Integer	->	<code>dk key as string</code> : retourner la clé primaire en texte, quel que soit son type d'origine
Résultat	Text	<-	Valeur de la clé primaire texte de l'entité
Résultat	Integer	<-	Valeur de la clé primaire numérique de l'entité

Description

La fonction `.getKey()` retourne la valeur de la clé primaire de l'entité.

Les clés primaires peuvent être des nombres (integer) ou des textes. Vous pouvez "forcer" la méthode à retourner la valeur de clé primaire sous forme de chaîne, quel que soit son type d'origine, en passant l'option `dk key as string` dans le paramètre `mode`.

Exemple

```
var $employees : cs.EmployeeSelection
var $employee : cs.EmployeeEntity
$employees:=ds.Employee.query("lastName=:1";"Smith")
$employee:=$employees[0]
ALERT("The primary key is "+$employee.getKey(dk key as string))
```

.getRemoteContextAttributes()

► Historique

.getRemoteContextAttributes() : Text

Paramètres	Type		Description
result	Text	<-	Attributs de contexte associés à l'entity, séparés par une virgule

Mode avancé : Cette fonction est destinée aux développeurs qui souhaitent personnaliser les fonctionnalités par défaut de ORDA dans le cadre de configurations spécifiques. Dans la plupart des cas, vous n'aurez pas besoin de l'utiliser.

Description

La fonction `.getRemoteContextAttributes()` retourne des informations relatives au contexte d'optimisation utilisé par l'entity .

S'il n'existe pas de [contexte d'optimisation](#) pour l'entity, la fonction retourne un texte vide.

Exemple

```
var $ds : 4D.DataStoreImplementation
var $address : cs.AddressEntity
var $p : cs.PersonsEntity
var $contextA : Object
var $info : Text
var $text : Text

$ds:=Open datastore(New object("hostname"; "www.myserver.com"); "myDS")

$contextA:=New object("context"; "contextA")

$address:=$ds.Address.get(1; $contextA)
$text:=""
For each ($p; $address.persons)
    $text:=$p.firstname+" "+$p.lastname
End for each

$info:=$address.getRemoteContextAttributes()

//$info = "persons,persons.lastname,persons.firstname"
```

Voir aussi

[EntitySelection.getRemoteContextAttributes\(\)](#)
[.clearAllRemoteContexts\(\)](#)
[.getRemoteContextInfo\(\)](#)
[.getAllRemoteContexts\(\)](#)
[.setRemoteContextInfo\(\)](#)

.getSelection()

► Historique

[.getSelection\(\): 4D.EntitySelection](#)

Paramètres	Type		Description
Résultat	4D.EntitySelection	<-	Entity selection à laquelle appartient l'entité (Null si non trouvée)

Description

La fonction `.getSelection()` retourne l'entity selection à laquelle appartient l'entité.

Si l'entité n'appartient pas à une entity selection, la fonction renvoie Null.

Exemple

```
var $emp : cs.EmployeeEntity
var $employees; $employees2 : cs.EmployeeSelection
$emp:=ds.Employee.get(672) // Cette entité n'appartient à aucune entity selection
$employees:=$emp.getSelection() // $employees contient Null

$employees2:=ds.Employee.query("lastName=:1";"Smith") //Cette entity selection contient 6 entités
$emp:=$employees2[0] // Cette entité appartient à une entity selection

ALERT("L'entity selection contient "+String($emp.getSelection().length)+" entités")
```

.getStamp()

► Historique

.getStamp() : Integer

Paramètres	Type		Description
Résultat	Integer	<-	Valeur du "stamp" de l'entité (0 si l'entité vient d'être créée)

Description

La fonction `.getStamp()` retourne la valeur courante du stamp de l'entité.

Le stamp (marqueur interne) d'une entité est automatiquement incrémenté par 4D à chaque fois qu'une entité est enregistrée sur disque. Il permet de gérer les accès et modifications concurrent(e)s sur les mêmes entités (cf. [Entity locking](#)).

Pour une nouvelle entité (jamais enregistrée), la fonction retourne 0. Pour savoir si une entité vient d'être créée, il est cependant recommandé d'utiliser [.isNew\(\)](#).

Exemple

```
var $entity : cs.EmployeeEntity
var $stamp : Integer

$entity:=ds.Employee.new()
$entity.lastname:="Smith"
$entity.save()
$stamp:=$entity.getStamp() // $stamp=1

$entity.lastname:="Wesson"
$entity.save()
$stamp:=$entity.getStamp() // $stamp=2
```

.indexOf()

► Historique

.indexOf({ entitySelection : 4D.EntitySelection }) : Integer

Paramètres	Type		Description
entitySelection	4D.EntitySelection	->	Entity selection dans laquelle obtenir la position de l'entité
Résultat	Integer	<-	Position de l'entité dans l'entity selection

Description

La fonction `.index0f()` retourne la position de l'entité dans une entity selection.

Par défaut, si le paramètre `entitySelection` est omis, la fonction retourne la position de l'entité dans sa propre entity selection (si elle existe). Sinon, elle renvoie la position de l'entité dans l'`entitySelection` spécifiée.

La valeur résultante est comprise entre 0 et la longueur de l'entity selection -1.

- Si l'entité n'a pas d'entity selection ou n'appartient pas à `entitySelection`, la fonction retourne -1.
- Si `entitySelection` est Null ou n'appartient pas à la même dataclass que l'entité, une erreur est générée.

Exemple

```
var $employees : cs.EmployeeSelection
var $employee : cs.EmployeeEntity
$employees:=ds.Employee.query("lastName = :1;"H@") //Cette entity selection contient 3 entités
$employee:=$employees[1] //Cette entité appartient à l'entity selection
ALERT("La position de cette entité dans son entity selection est "+String($employee.index0f())) //1

$employee:=ds.Employee.get(725) //Cette entité n'appartient pas à l'entity selection
ALERT("La position de cette entité est "+String($employee.index0f())) // -1
```

.isNew()

► Historique

`.isNew()` : Boolean

Paramètres	Type		Description
Résultat	Booléen	<-	Vrai si l'entité vient juste d'être créée et n'a pas encore été enregistrée. Sinon, Faux.

Description

La fonction `.isNew()` renvoie Vrai si l'entité à laquelle elle est appliquée vient d'être créée et n'a pas encore été enregistrée dans le datastore. Sinon, elle renvoie Faux.

Exemple

```
var $emp : cs.EmployeeEntity

$emp:=ds.Employee.new()

If($emp.isNew())
    ALERT("Ceci est une nouvelle entité")
End if
```

.last()

► Historique

`.last()` : 4D.Entity

Paramètres	Type		Description
Résultat	4D.Entity	<-	Référence à la dernière entité de l'entity selection (Null si non trouvée)

Description

La fonction `.last()` renvoie une référence vers l'entité en dernière position dans l'entity selection à laquelle l'entité appartient.

Si l'entité n'appartient à aucune entity selection (i.e. `.getSelection()` retourne Null), la fonction renvoie une valeur Null.

Exemple

```
var $employees : cs.EmployeeSelection
var $employee; $lastEmployee : cs.EmployeeEntity
$employees:=ds.Employee.query("lastName = :1";"H@") //Cette entity selection contient 3 entités $employ
$lastEmployee:=$employee.last() // $lastEmployee est la dernière entité de l'entity selection $employees
```

.lock()

► Historique

`.lock({ mode : Integer }) : Object`

Paramètres	Type		Description
mode	Integer	->	<code>dk reload if stamp changed</code> : Recharger avant de verrouiller si le marqueur interne a changé
Résultat	Object	<-	Résultat de l'opération lock

Description

La fonction `.lock()` pose un verrou pessimiste sur l'enregistrement référencé par l'entité. Le [verrou est posé](#) pour l'enregistrement et toutes les références de l'entité dans le process courant.

Pour les autres process, cet enregistrement apparaîtra verrouillé (la propriété `result.success` contiendra Faux s'ils tentent de verrouiller la même entité à l'aide de cette fonction). Seules les fonctions exécutées dans la session à l'origine du verrouillage auront la possibilité de modifier et de sauvegarder les attributs de l'entité. L'entité peut être chargée en lecture seulement dans les autres sessions, mais elles ne pourront ni saisir ni sauvegarder des valeurs.

Un enregistrement verrouillé peut être déverrouillé :

- lorsque la fonction `unlock()` est appelée sur une entité correspondante dans le même process
- automatiquement, lorsqu'elle n'est plus référencée par aucune entité en mémoire. Par exemple, si le verrou n'est posé que sur une référence locale d'une entité, celle-ci est déverrouillée à la fin de la fonction. Tant qu'il existe des références à l'entité en mémoire, l'enregistrement reste verrouillé.

Pour plus d'informations, veuillez consulter la section [Verrouillage d'une entité](#).

Par défaut, lorsque le paramètre `mode` est omis, la méthode retourne une erreur (cf. ci-dessous) si l'entité a été modifiée (i.e. le marqueur interne a changé) entre-temps par un autre process ou utilisateur.

Sinon, vous pouvez passer l'option `dk reload if stamp changed` dans le paramètre `mode` : dans ce cas, aucune erreur n'est générée et l'entité est simplement rechargée si le stamp a changé (si l'entité existe toujours et si la clé primaire est toujours la même).

Résultat

L'objet retourné par `.lock()` contient les propriétés suivantes :

Propriété		Type	Description
success		boolean	vrai si l'action de verrouillage a été effectuée avec succès (ou si l'entité est déjà verrouillée dans le process courant), sinon faux.
			<i>Disponible uniquement si l'option <code>dk reload if stamp changed</code> est utilisée :</i>
wasReloaded		boolean	vrai si l'entité a été correctement rechargée, sinon faux.
			<i>Disponible uniquement en cas d'erreur :</i>
status(*)		number	Code d'erreur, voir ci-dessous
statusText(*)		text	Description de l'erreur, voir ci-dessous
			<i>Disponible uniquement en cas d'erreur de verrouillage pessimiste :</i>
lockKindText		text	"Locked by record" si verrouillage par un process 4D, "Locked by session" si verrouillage par une session REST
lockInfo		object	Information sur l'origine du verrouillage. Les propriétés retournées dépendent de l'origine du verrouillage (process 4D ou session REST).
			<i>Disponible uniquement pour un verrouillage par process 4D:</i>
	task_id	number	ID du process
	user_name	text	Nom d'utilisateur de la session sur la machine
	user4d_alias	Texte	Nom ou alias de l'utilisateur 4D
	user4d_id	number	Identifiant utilisateur dans le répertoire de la base 4D
	host_name	text	Nom de la machine
	task_name	text	Nom du process
	client_version	Texte	Version du client
			<i>Disponible uniquement pour un verrouillage par session REST :</i>
	host	text	URL ayant verrouillé l'entité (ex : " www.myserver.com ")
	IPAddr	text	Adresse IP d'origine du verrouillage (ex. 127.0.0.1")
	userAgent	Texte	userAgent de l'origine du verrouillage (ex : "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.110 Safari/537.36")
			<i>Disponible uniquement en cas d'erreur critique (clé primaire dupliquée, disque plein...) :</i>
errors		collection d'objets	
	message	Texte	Message d'erreur
	component signature	Texte	signature du composant interne (e.g. "dmbg" pour le composant de base de données)
	errCode	number	Code d'erreur

(*) Les valeurs suivantes peuvent être retournées dans les propriétés *status* et *statusText* de l'objet *Résultat* en cas d'erreur :

Constante	Valeur	Commentaire
dk status entity does not exist anymore	5	<p>L'entité n'existe plus dans les données. Cette erreur peut se produire dans les cas suivants :</p> <ul style="list-style-type: none"> • l'entité a été supprimée (le stamp est modifié et l'espace mémoire est libéré) • l'entité a été supprimée et remplacée par une autre avec une clé primaire différente (le stamp est modifié et une nouvelle entité occupe l'espace mémoire). Avec <code>.drop()</code>, cette erreur peut être retournée lorsque l'option <code>dk force drop if stamp changed</code> est utilisée. Avec <code>.lock()</code>, cette erreur peut être retournée lorsque l'option <code>dk reload if stamp changed</code> est utilisée. <p>statusText associé : "Entity does not exist anymore"</p>
dk status locked	3	<p>L'entité est verrouillée par un verrou pessimiste.</p> <p>statusText associé : "Already locked"</p>
dk status serious error	4	<p>Une erreur critique peut être une erreur de bas niveau de la base de données (ex. clé dupliquée), une erreur matérielle, etc.</p> <p>statusText associé : "Other error"</p>
dk status stamp has changed	2	<p>La valeur du marqueur interne (stamp) de l'entité ne correspond pas à celle de l'entité stockée dans les données (verrouillage optimiste).</p> <ul style="list-style-type: none"> • avec <code>.save()</code> : erreur uniquement si l'option <code>dk auto merge</code> n'est pas utilisée • avec <code>.drop()</code> : erreur uniquement si l'option <code>dk force drop if stamp changed</code> n'est pas utilisée • avec <code>.lock()</code> : erreur uniquement si l'option <code>dk reload if stamp changed</code> n'est pas utilisée <p>statusText associé : "Stamp has changed"</p>

Exemple 1

Exemple avec erreur :

```

var $employee : cs.EmployeeEntity
var $status : Object
$employee:=ds.Employee.get(716)
$status:=$employee.lock()
Case of
    :($status.success)
        ALERT("You have locked "+$employee.firstName+" "+$employee.lastName)
    :($status.status=dk status stamp has changed)
        ALERT($status.statusText)
End case

```

Exemple 2

Exemple avec option `dk reload if stamp changed` :

```

var $employee : cs.EmployeeEntity
var $status : Object
$employee:=ds.Employee.get(717)
$status:=$employee.lock(dk reload if stamp changed)
Case of
    :($status.success)
        ALERT("You have locked "+$employee.firstName+" "+$employee.lastName)
    :($status.status=dk status entity does not exist anymore)
        ALERT($status.statusText)
End case

```

.next()

► Historique

.next() : 4D.Entity

Paramètres	Type		Description
Résultat	4D.Entity	<-	Référence à l'entité suivante dans l'entity selection (Null si non trouvée)

Description

La fonction `.next()` retourne une référence sur l'entité suivante dans l'entity selection à laquelle appartient l'entité.

Si l'entité n'appartient à aucune entity selection existante (i.e. `.getSelection()` retourne Null), la fonction renvoie une valeur Null.

S'il n'y a pas d'entité suivante valide dans l'entity selection (i.e. vous êtes sur la dernière entité de la sélection), la fonction retourne Null. Si l'entité suivante a été supprimée, la fonction renvoie l'entité valide suivante (et finalement Null).

Exemple

```

var $employees : cs.EmployeeSelection
var $employee; $nextEmployee : cs.EmployeeEntity
$employees:=ds.Employee.query("lastName = :1";"H@") //Cette entity selection contient 3 entités
$employee:=$employees[0]
$nextEmployee:=$employee.next() // $nextEmployee est la deuxième entité de l'entity selection $employees

```

.previous()

► Historique

.previous() : 4D.Entity

Paramètres	Type		Description
Résultat	4D.Entity	<-	Référence à l'entité précédente dans l'entity selection (Null si non trouvée)

Description

La fonction `.previous()` retourne une référence sur l'entité précédente dans l'entity selection à laquelle appartient l'entité.

Si l'entité n'appartient à aucune entity selection existante (i.e. `.getSelection()` retourne Null), la fonction renvoie une valeur Null.

Si l'entité n'appartient à aucune entity selection (i.e. `.getSelection()` retourne Null), la fonction renvoie une valeur Null.

Exemple

```
var $employees : cs.EmployeeSelection
var $employee; $previousEmployee : cs.EmployeeEntity
$employees:=ds.Employee.query("lastName = :1";"HQ") //Cette entity selection contient 3 entités
$employee:=$employees[1]
$previousEmployee:=$employee.previous() // $previousEmployee est la première entité de l'entity selection
```

.reload()

► Historique

.reload() : Object

Paramètres	Type		Description
Résultat	Object	<-	Objet statut

Description

La fonction `.reload()` recharge en mémoire le contenu de l'entité à partir des informations stockées dans la table associée à la dataclass dans le datastore. Le rechargement est effectué uniquement si l'entité existe toujours avec la même clé primaire.

Résultat

L'objet retourné par `.reload()` contient les propriétés suivantes :

Propriété	Type	Description
success	boolean	Vrai si le rechargement a été effectué avec succès, sinon Faux. <i>Disponible uniquement en cas d'erreur :</i>
status(*)	number	Code d'erreur, voir ci-dessous
statusText(*)	text	Description de l'erreur, voir ci-dessous

(*) Les valeurs suivantes peuvent être retournées dans les propriétés `status` et `statusText` de l'objet `Résultat` en cas d'erreur :

Constante	Valeur	Commentaire
dk status entity does not exist anymore	5	L'entité n'existe plus dans les données. Cette erreur peut se produire dans les cas suivants : <ul style="list-style-type: none">• l'entité a été supprimée (le stamp est modifié et l'espace mémoire est libéré)• l'entité a été supprimée et remplacée par une autre avec une clé primaire différente (le stamp est modifié et une nouvelle entité occupe l'espace mémoire). Avec <code>.drop()</code>, cette erreur peut être retournée lorsque l'option <code>dk force drop if stamp changed</code> est utilisée. Avec <code>.lock()</code>, cette erreur peut être retournée lorsque l'option <code>dk reload if stamp changed</code> est utilisée. <i>statusText associé : "Entity does not exist anymore"</i>
dk status serious error	4	Une erreur critique peut être une erreur de bas niveau de la base de données (ex. clé dupliquée), une erreur matérielle, etc. <i>statusText associé : "Other error"</i>

Exemple

```

var $employee : cs.EmployeeEntity
var $employees : cs.EmployeeSelection
var $result : Object

$employees:=ds.Employee.query("lastName=:1";"Hollis")
$employee:=$employees[0]
$employee.firstName:="Mary"
$result:=$employee.reload()

Case of
    :($result.success)
        ALERT("Rechargement effectué")
    :($result.status=dk status entity does not exist anymore)
        ALERT("L'entité a été supprimée")
End case

```

.save()

► Historique

.save({ mode : Integer }) : Object

Paramètres	Type		Description
mode	Integer	->	dk auto merge : Active le mode "automatic merge"
Résultat	Object	<-	Résultat de la sauvegarde

Description

La fonction `.save()` sauvegarde les modifications effectuées sur l'entité dans la table de sa dataclass. Vous devez appeler cette fonction après toute création ou modification d'entité si vous souhaitez sauvegarder les changements.

La sauvegarde est effectuée si et seulement si au moins un attribut de l'entité a été "touché" (cf. les fonctions `.touched()` et `.touchedAttributes()`). Sinon, la fonction ne fait rien (le trigger n'est pas appelé).

Dans une application multi-utilisateur ou multi-process, la fonction `.save()` est exécutée avec le mécanisme du "verrouillage optimiste", dans lequel un compteur interne (stamp) est automatiquement incrémenté à chaque sauvegarde de l'enregistrement.

Par défaut, si le paramètre `mode` est omis, la fonction retournera systématiquement une erreur (voir ci-dessous) lorsque la même entité a été modifiée entre-temps par un autre process ou utilisateur, quel(s) que soi(en)t l(es) attribut(s) modifié(s).

Sinon, vous pouvez passer l'option `dk auto merge` dans le paramètre `mode` afin d'activer le mode "automatic merge". Dans ce mode, une modification simultanée effectuée par un autre process/utilisateur sur la même entité mais sur un attribut différent ne génère pas d'erreur. Les données effectivement stockées dans l'enregistrement résultent alors de la combinaison (le "merge") des modifications non-concurrentes (si des modifications ont été effectuées sur le même attribut, la sauvegarde échoue et une erreur est retournée, même en mode "automatic merge").

Le mode de fusion automatique n'est pas disponible pour les attributs de type Image, Objet et Texte lorsqu'ils sont stockés en dehors de l'enregistrement. Des modifications simultanées de ces attributs entraîneront une erreur "dk status stamp has changed".

Résultat

L'objet retourné par `.save()` contient les propriétés suivantes :

Propriété		Type	Description
success		boolean	Vrai si la sauvegarde a été effectuée avec succès, sinon faux.
			<i>Disponible uniquement si l'option <code>dk auto merge</code> a été utilisée :</i>
autoMerged		boolean	Vrai si un "auto merge" a été effectué, sinon faux.
			<i>Disponible uniquement en cas d'erreur :</i>
status		number	Code d'erreur, voir ci-dessous
statusText		text	Description de l'erreur, voir ci-dessous
			<i>Disponible uniquement en cas d'erreur en verrouillage pessimiste :</i>
lockKindText		text	"Locked by record"
lockInfo		object	Information sur l'origine du verrouillage
	task_id	number	Id du process
	user_name	text	Nom d'utilisateur de la session sur la machine
	user4d_alias	text	Alias utilisateur si défini avec <code>SET USER ALIAS</code> , sinon le nom d'utilisateur dans le répertoire de la base 4D
	host_name	text	Nom de la machine
	task_name	text	Nom du process
	client_version	text	
			<i>Disponible uniquement en cas d'erreur critique (clé primaire dupliquée, disque plein...) :</i>
errors		collection of objects	
	message	text	Message d'erreur
	componentSignature	text	Signature du composant interne (e.g. "dmbg" pour le composant de base de données)
	errCode	number	Code d'erreur

status et statusText

Les valeurs suivantes peuvent être retournées dans les propriétés `status` et `statusText` de l'objet Résultat en cas d'erreur :

Constante	Valeur	Commentaire
dk status automerge failed	6	(Uniquement si l'option <code>dk auto merge</code> est utilisée) Echec du mécanisme de merge automatique lors de la sauvegarde de l'entité. statusText associé : "Auto merge failed"
dk status entity does not exist anymore	5	L'entité n'existe plus dans les données. Cette erreur peut se produire dans les cas suivants : <ul style="list-style-type: none"> l'entité a été supprimée (le stamp est modifié et l'espace mémoire est libéré) l'entité a été supprimée et remplacée par une autre avec une clé primaire différente (le stamp est modifié et une nouvelle entité occupe l'espace mémoire). Avec <code>.drop()</code>, cette erreur peut être retournée lorsque l'option <code>dk force drop if stamp changed</code> est utilisée. Avec <code>.lock()</code>, cette erreur peut être retournée lorsque l'option <code>dk reload if stamp changed</code> est utilisée. statusText associé: "Entity does not exist anymore"
dk status locked	3	L'entité est verrouillée par un verrou pessimiste. statusText associé : "Already locked"
dk status serious error	4	Une erreur critique peut être une erreur de bas niveau de la base de données (ex. clé dupliquée), une erreur matérielle, etc. statusText associé : "Other error"
dk status stamp has changed	2	La valeur du marqueur interne (stamp) de l'entité ne correspond pas à celle de l'entité stockée dans les données (verrouillage optimiste). <ul style="list-style-type: none"> avec <code>.save()</code> : erreur uniquement si l'option <code>dk auto merge</code> n'est pas utilisée avec <code>.drop()</code> : erreur uniquement si l'option <code>dk force drop if stamp changed</code> n'est pas utilisée avec <code>.lock()</code> : erreur uniquement si l'option <code>dk reload if stamp changed</code> n'est pas utilisée statusText associé: "Stamp has changed"

Exemple 1

Création d'une entité :

```

var $status : Object
var $employee : cs.EmployeeEntity
$employee:=ds.Employee.new()
$employee.firstName:="Mary"
$employee.lastName:="Smith"
$status:=$employee.save()
If($status.success)
    ALERT("Employé créé")
End if

```

Exemple 2

Mise à jour d'une entité sans option `dk auto merge` :

```

var $status : Object
var $employee : cs.EmployeeEntity
var $employees : cs.EmployeeSelection
$employees:=ds.Employee.query("lastName=:1";"Smith")
$employee:=$employees.first()
$employee.lastName:="Mac Arthur"
$status:=$employee.save()
Case of
    :($status.success)
        ALERT("Employé mis à jour")
    :($status.status=dk status stamp has changed)
        ALERT($status.statusText)
End case

```

Exemple 3

Mise à jour d'une entité avec option `dk auto merge` :

```

var $status : Object
var $employee : cs.EmployeeEntity
var $employees : cs.EmployeeSelection

$employees:=ds.Employee.query("lastName=:1";"Smith")
$employee:=$employees.first()
$employee.lastName:="Mac Arthur"
$status:=$employee.save(dk auto merge)
Case of
    :($status.success)
        ALERT("Employé mis à jour")
    :($status.status=dk status automerge failed)
        ALERT($status.statusText)
End case

```

.toObject()

► Historique

`.toObject() : Object`
`.toObject(filterString : Text { ; options : Integer}) : Object`
`.toObject(filterCol : Collection { ; options : Integer }) : Object`

Paramètres	Type		Description
filterString	Text	->	Attribut(s) à extraire (chaînes séparées par des virgules)
filterCol	Collection	->	Collection d'attribut(s) à extraire
options	Integer	->	<code>dk with primary key</code> : ajouter la propriété _KEY ; <code>dk with stamp</code> : ajouter la propriété _STAMP
Résultat	Object	<-	Objet généré à partir de l'entité

Description

La fonction `.toObject()` retourne un objet construit à partir de l'entité. Les noms des propriétés de l'objet correspondent aux noms des attributs de l'entité.

Si aucun filtre n'est passé ou si le paramètre `filterString` contient une chaîne vide ou `"*"`, l'objet retourné contiendra :

- tous les attributs storage de l'entité
- attributs dont le `kind` est `relatedEntity` : vous obtenez une propriété avec le même nom que l'entité liée (nom du

lien N vers 1). L'attribut est extrait sous forme simple.

- attributs dont le `kind` est `relatedEntities` : non retournés.

Dans le premier paramètre, indiquez le ou les attribut(s) à extraire. Vous pouvez passer :

- `filterString` : une chaîne avec les chemins des propriétés séparés par des virgules : "propertyPath1, propertyPath2, ...", ou
- `filterCol` : une collection de chaînes contenant des chemins de propriétés : ["propertyPath1","propertyPath2";...]

Si un filtre contient des attributs dont le `kind` est `relatedEntity` :

- `propertyPath = "relatedEntity"` -> l'attribut est extrait sous forme simple : un objet avec la propriété `__KEY` (clé primaire).
- `propertyPath = "relatedEntity.*"` -> toutes les propriétés sont extraites
- `propertyPath = "relatedEntity.propertyName1; relatedEntity.propertyName2; ..."` -> seules ces propriétés sont extraites

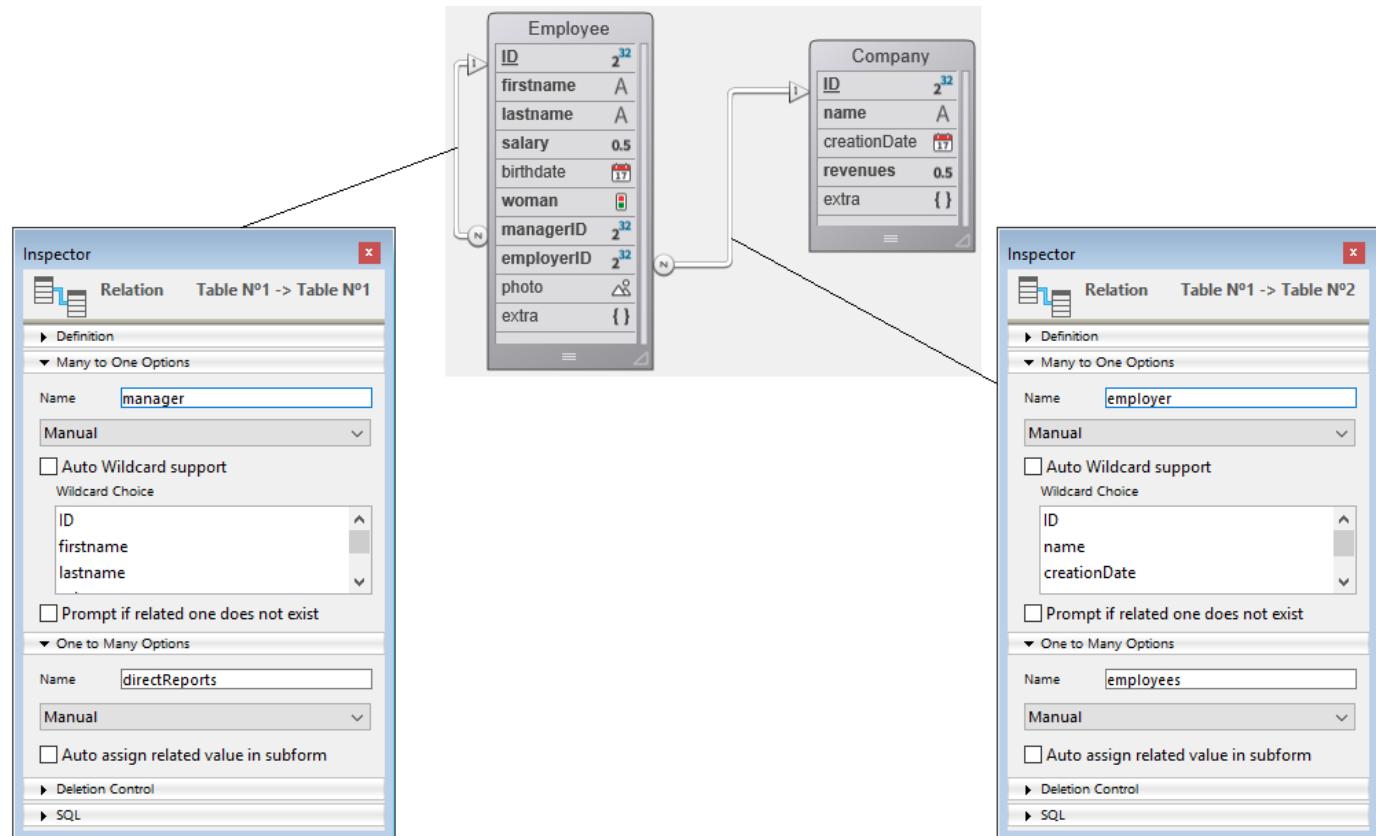
Si un filtre contient de attributs dont le `kind` est `relatedEntities` :

- `propertyPath = "relatedEntities.*"` -> toutes les propriétés sont extraites
- `propertyPath = "relatedEntities.propertyName1; relatedEntities.propertyName2; ..."` -> seules ces propriétés sont extraites

Dans le paramètre `options`, vous pouvez passer les sélecteurs `dk with primary key` et/ou `dk with stamp` afin d'ajouter les clés primaires et/ou les stamps dans les objets extraits.

Exemple 1

La structure suivante sera utilisée pour les exemples de cette section :



Sans paramètre filtre :

```
employeeObject:=employeeSelected.toObject()
```

Retourne :

```
{
    "ID": 413,
    "firstName": "Greg",
    "lastName": "Wahl",
    "salary": 0,
    "birthDate": "1963-02-01T00:00:00.000Z",
    "woman": false,
    "managerID": 412,
    "employerID": 20,
    "photo": "[object Picture]",
    "extra": null,
    "employer": { // relatedEntity sous forme simple
        "__KEY": 20
    },
    "manager": {
        "__KEY": 412
    }
}
```

Exemple 2

Extraction de la clé primaire et du stamp :

```
employeeObject:=employeeSelected.toObject("";dk with primary key+dk with stamp)
```

Retourne :

```
{
    "__KEY": 413,
    "__STAMP": 1,
    "ID": 413,
    "firstName": "Greg",
    "lastName": "Wahl",
    "salary": 0,
    "birthDate": "1963-02-01T00:00:00.000Z",
    "woman": false,
    "managerID": 412,
    "employerID": 20,
    "photo": "[object Picture]",
    "extra": null,
    "employer": {
        "__KEY": 20
    },
    "manager": {
        "__KEY": 412
    }
}
```

Exemple 3

Extraction complète des attributs des `relatedEntities` :

```
employeeObject:=employeeSelected.toObject("directReports.*")
```

```
{
  "directReports": [
    {
      "ID": 418,
      "firstName": "Lorena",
      "lastName": "Boothe",
      "salary": 44800,
      "birthDate": "1970-10-02T00:00:00.000Z",
      "woman": true,
      "managerID": 413,
      "employerID": 20,
      "photo": "[object Picture]",
      "extra": null,
      "employer": {
        "__KEY": 20
      },
      "manager": {
        "__KEY": 413
      }
    },
    {
      "ID": 419,
      "firstName": "Drew",
      "lastName": "Caudill",
      "salary": 41000,
      "birthDate": "2030-01-12T00:00:00.000Z",
      "woman": false,
      "managerID": 413,
      "employerID": 20,
      "photo": "[object Picture]",
      "extra": null,
      "employer": {
        "__KEY": 20
      },
      "manager": {
        "__KEY": 413
      }
    },
    {
      "ID": 420,
      "firstName": "Nathan",
      "lastName": "Gomes",
      "salary": 46300,
      "birthDate": "2010-05-29T00:00:00.000Z",
      "woman": false,
      "managerID": 413,
      "employerID": 20,
      "photo": "[object Picture]",
      "extra": null,
      "employer": {
        "__KEY": 20
      },
      "manager": {
        "__KEY": 413
      }
    }
  ]
}
}
```

Exemple 4

Extraction de quelques attributs des `relatedEntities` :

```
employeeObject:=employeeSelected.toObject("firstName, directReports.lastName")
```

Retourne :

```
{  
    "firstName": "Greg",  
    "directReports": [  
        {  
            "lastName": "Boothe"  
        },  
        {  
            "lastName": "Caudill"  
        },  
        {  
            "lastName": "Gomes"  
        }  
    ]  
}
```

Exemple 5

Extraction d'une `relatedEntity` sous forme simple :

```
$coll:=New collection("firstName";"employer")  
employeeObject:=employeeSelected.toObject($coll)
```

Retourne :

```
{  
    "firstName": "Greg",  
    "employer": {  
        "__KEY": 20  
    }  
}
```

Exemple 6

Extraction de tous les attributs d'une `relatedEntity` :

```
employeeObject:=employeeSelected.toObject("employer.*")
```

Retourne :

```
{  
    "employer": {  
        "ID": 20,  
        "name": "India Astral Secretary",  
        "creationDate": "1984-08-25T00:00:00.000Z",  
        "revenues": 12000000,  
        "extra": null  
    }  
}
```

Exemple 7

Extraction de quelques attributs d'une `relatedEntity` :

```
$col:=New collection
$col.push("employer.name")
$col.push("employer.revenues")
employeeObject:=employeeSelected.toObject($col)
```

Retourne :

```
{
    "employer": {
        "name": "India Astral Secretary",
        "revenues": 12000000
    }
}
```

.touched()

► Historique

`.touched()` : Boolean

Paramètres	Type		Description
Résultat	Booléen	<-	Vrai si au moins un attribut de l'entité a été modifié et non encore sauvegardé, sinon Faux

Description

La fonction `.touched()` indique si un attribut de l'entité a été modifié ou non depuis que l'entité a été chargée en mémoire ou sauvegardée.

Si un attribut a été modifié ou calculé, la fonction retourne Vrai, sinon elle retourne Faux. Vous pouvez utiliser cette fonction pour savoir s'il est nécessaire de sauvegarder l'entité.

Cette fonction retourne Faux pour une entité qui vient d'être créée (avec `.new()`). A noter cependant que si vous utilisez une fonction pour calculer un attribut de l'entité, la fonction `.touched()` retournera Vrai. Par exemple, si vous appelez `.getKey()` pour calculer la clé primaire, `.touched()` retourne alors Vrai.

Exemple

Cet exemple vérifie s'il est nécessaire de sauvegarder l'entité :

```
var $emp : cs.EmployeeEntity
$emp:=ds.Employee.get(672)
$emp.firstName:=$emp.firstName //Même réassigné avec sa propre valeur, l'attribut est considéré "touche
If($emp.touched()) //si au moins l'un des attributs a été modifié
    $emp.save()
End if // sinon, inutile de sauvegarder l'entité
```

.touchedAttributes()

► Historique

`.touchedAttributes()` : Collection

Paramètres	Type		Description
Résultat	Collection	<-	Noms des attributs touchés ou collection vide

Description

La fonction `.touchedAttributes()` retourne les noms des attributs qui ont été modifiés depuis que l'entité a été chargée en mémoire.

Cette fonction est applicable aux attributs dont le `kind` est `storage` ou `relatedEntity`.

Dans le cas d'un attribut relationnel ayant été "touché" (i.e., la clé étrangère), le nom de l'entité liée et celui de sa clé primaire sont retournés.

Si aucun attribut de l'entité n'a été touché, la fonction retourne une collection vide.

Exemple 1

```
var $touchedAttributes : Collection
var $emp : cs.EmployeeEntity

$touchedAttributes:=New collection
$emp:=ds.Employee.get(725)
$emp.firstName:=$emp.firstName //Même modifié avec la même valeur, l'attribut est considéré comme touché
$emp.lastName:="Martin"
$touchedAttributes:=$emp.touchedAttributes()
//$touchedAttributes: ["firstName", "lastName"]
```

Exemple 2

```
var $touchedAttributes : Collection
var $emp : cs.EmployeeEntity
var $company : cs.CompanyEntity

$touchedAttributes:=New collection

$emp:=ds.Employee.get(672)
$emp.firstName:=$emp.firstName
$emp.lastName:="Martin"

$company:=ds.Company.get(121)
$emp.employer:=$company

$touchedAttributes:=$emp.touchedAttributes()

//collection $touchedAttributes: ["firstName", "lastName", "employer", "employerID"]
```

Dans ce cas :

- `firstName` et `lastName` ont un type `storage`
- `employer` a un type `relatedEntity`
- `employerID` est la clé étrangère de l'entité reliée `employer`

.unlock()

► Historique

`.unlock() : Object`

Paramètres	Type		Description
Résultat	Object	<-	Objet statut

Description

La fonction `.unlock()` supprime le verrou pessimiste posé sur l'enregistrement correspondant à l'entité dans sa table du datastore.

Pour plus d'informations, veuillez consulter la section [Verrouillage d'une entité](#).

Un enregistrement est automatiquement déverrouillé lorsqu'il n'est plus référencé par aucune entité dans le process qui l'a verrouillé (par exemple : si le verrou est posé sur uniquement sur une référence locale d'une entité, l'entité et donc l'enregistrement sont déverrouillés lorsque le process se termine).

Lorsqu'un enregistrement est verrouillé, il doit être déverrouillé depuis le process qui l'a verrouillé et via la référence d'entité sur laquelle le verrou a été posé. Par exemple :

```
$e1:=ds.Emp.all()[0]
$e2:=ds.Emp.all()[0]
$res:=$e1.lock() // $res.success=true
$res:=$e2.unlock() // $res.success=false
$res:=$e1.unlock() // $res.success=true
```

Résultat

L'objet retourné par `.unlock()` contient la propriété suivante :

Propriété	Type	Description
success	Boolean	Vrai si l'action unlock a été exécutée avec succès, Faux sinon. Si le déverrouillage est effectué sur une entité qui a été supprimée, sur un enregistrement non verrouillé ou sur un enregistrement verrouillé par un autre process ou une autre entité, success vaut Faux.

Exemple

```
var $employee : cs.EmployeeEntity
var $status : Object

$employee:=ds.Employee.get(725)
$status:=$employee.lock()
... //processing
$status:=$employee.unlock()
If($status.success)
    ALERT("L'entité est déverrouillée")
End if
```

EntitySelection

Une entity selection est un objet contenant une ou plusieurs référence(s) à des entités appartenant à la même Dataclass. Une entity selection peut contenir 0, 1 ou X entités de la dataclass - où X peut représenter le nombre total d'entités contenues dans la dataclass.

Les entity selections peuvent être créées à partir de sélections existantes à l'aide de diverses fonctions de la classe `DataClass` telles que `.all()` ou `.query()`, ou de la classe `EntityClass` elle-même, telles que `.and()` ou `orderBy()`. Vous pouvez également créer des entity selections vides à l'aide de la fonction `dataClass.newSelection()` ou de la commande `Create new selection`.

Sommaire

`[index] : 4D.Entity`

vous permet d'accéder aux entités de l'entity selection à l'aide de la syntaxe standard des collections

`.attributeName : Collection`

`.attributeName : 4D.EntitySelection`

une "projection" des valeurs de l'attribut dans l'entity selection

`.add(entity : 4D.Entity) : 4D.EntitySelection`

ajoute l'entity spécifiée à l'entity selection et retourne l'entity selection modifiée

`.and(entity : 4D.Entity) : 4D.EntitySelection`

`.and(entitySelection : 4D.EntitySelection) : 4D.EntitySelection`

combine l'entity selection avec le paramètre `entity` ou `entitySelection` à l'aide de l'opérateur logique ET

`.average(attributePath : Text) : Real`

retourne la moyenne arithmétique de toutes les valeurs non nulles de `attributePath` dans l'entity selection

`.contains(entity : 4D.Entity) : Boolean`

retourne vrai si la référence d'entité appartient à l'entity selection

`.count(attributePath : Text) : Real`

retourne le nombre d'entités dans l'entity selection pour lesquelles la valeur de `attributePath` n'est pas null

`.distinct(attributePath : Text { ; option : Integer }) : Collection`

renvoie une collection contenant uniquement les valeurs distinctes (différentes) dans la collection d'origine

`.drop({ mode : Integer }) : 4D.EntitySelection`

supprime les entités appartenant à l'entity selection de la table liée à sa dataclass dans le datastore

`.extract(attributePath : Text { ; option : Integer }) : Collection`

`.extract(attributePath { ; targetPath } { ; ...attributePathN : Text ; targetPathN : Text }) : Collection`

retourne une collection contenant les valeurs `attributePath` extraites de l'entity selection

`.first() : 4D.Entity`

retourne une référence vers l'entité en première position dans l'entity selection

`.getDataClass() : 4D.DataClass`

retourne la dataclass de l'entity selection

`.getRemoteContextAttributes() : Text`

retourne des informations relatives au contexte d'optimisation utilisé par l'entity

<code>.isAlterable() : Boolean</code>	retourne Vrai si l'entity selection est modifiable (alterable)
<code>.isOrdered() : Boolean</code>	retourne Vrai si l'entity selection est triée
<code>.last() : 4D.Entity</code>	retourne une référence vers l'entité en dernière position dans l'entity selection
<code>.length : Integer</code>	retourne le nombre d'entités dans l'entity selection
<code>.max(attributePath : Text) : any</code>	retourne la plus haute valeur (ou valeur maximale) parmi toutes les valeurs de <i>attributePath</i> dans l'entity selection
<code>.min(attributePath : Text) : any</code>	retourne la plus faible valeur (ou valeur minimale) parmi toutes les valeurs de <i>attributePath</i> dans l'entity selection
<code>.minus(entity : 4D.Entity) : 4D.EntitySelection</code>	
<code>.minus(entitySelection : 4D.EntitySelection) : 4D.EntitySelection</code>	exclut de l'entity selection à laquelle elle est appliquée l' <i>entity</i> ou les entités de l' <i>entitySelection</i> et retourne l'entity selection résultante
<code>.or(entity : 4D.Entity) : 4D.EntitySelection</code>	
<code>.or(entitySelection : 4D.EntitySelection) : 4D.EntitySelection</code>	combine l'entity selection avec le paramètre <i>entity</i> ou <i>entitySelection</i> en utilisant l'opérateur OU logique (non exclusif)
<code>.orderBy(pathString : Text) : 4D.EntitySelection</code>	
<code>.orderBy(pathObjects : Collection) : 4D.EntitySelection</code>	renvoie une nouvelle entity selection triée contenant toutes les entités de l'entity selection dans l'ordre spécifié par le paramètre <i>pathString</i> ou <i>pathObjects</i>
<code>.orderByFormula(formulaString : Text { ; sortOrder : Integer } { ; settings : Object}) : 4D.EntitySelection</code>	
<code>.orderByFormula(formulaObj : Object { ; sortOrder : Integer } { ; settings : Object}) : 4D.EntitySelection</code>	renvoie une nouvelle entity selection triée
<code>.query(queryString : Text { ; ...value : any } { ; querySettings : Object }) : 4D.EntitySelection</code>	
<code>.query(formula : Object { ; querySettings : Object }) : 4D.EntitySelection</code>	cherche les entités répondant aux critères de recherche spécifiés dans <i>queryString</i> ou <i>formula</i> et (optionnellement) dans <i>value</i> toutes les entités de l'entity selection
<code>.queryPath : Text</code>	contient la description détaillée du chemin de recherche réel utilisé par 4D
<code>.queryPlan : Text</code>	contient la description détaillée du plan de recherche prévu par 4D avant de l'exécuter
<code>.refresh()</code>	"invalidé" immédiatement les données de l'entity selection dans le cache local d'ORDA
<code>.selected(selectedEntities : 4D.EntitySelection) : Object</code>	retourne un objet décrivant la ou les positions de <i>selectedEntities</i> dans l'entity selection d'origine
<code>.slice(startFrom : Integer { ; end : Integer }) : 4D.EntitySelection</code>	retourne une partie d'une entity selection dans une nouvelle entity selection

.sum(*attributePath* : Text) : Real

retourne la somme de toutes les valeurs d'*attributePath* dans l'entity selection

.toCollection({ *options* : Integer { ; *begin* : Integer { ; *howMany* : Integer } } }) : Collection

.toCollection(*filterString* : Text {; *options* : Integer { ; *begin* : Integer { ; *howMany* : Integer } } } }) : Collection

.toCollection(*filterCol* : Collection {; *options* : Integer { ; *begin* : Integer { ; *howMany* : Integer } } } }) : Collection

crée et retourne une collection dans laquelle chaque élément est un objet contenant un ensemble de propriétés et de valeurs

Créer une entity selection

Create entity selection (*dsTable* : Table { ; *settings* : Object }) : 4D.EntitySelection

Paramètres	Type		Description
<i>dsTable</i>	Table	->	Table de la base 4D dont la sélection courante doit être utilisée pour construire l'entity selection
<i>settings</i>	Object	->	Option de création : contexte
Résultat	4D.EntitySelection	<-	Nouvelle entity selection liée à la dataclass de la table

Description

La commande `Create entity selection` construit et retourne une nouvelle entity selection **altérable** liée à la dataclass correspondant à la table *dsTable*, basée sur la sélection courante de cette table.

Si la sélection courante est triée, une entity selection **triée** est créée (l'ordre de la sélection courante est conservé). Si la sélection courante n'est pas triée, une entity selection non-triée est créée.

Si la table *dsTable* n'est pas exposée dans `ds`, une erreur est retournée. Cette commande ne peut pas être utilisée avec un datastore distant.

Dans le paramètre optionnel *settings*, vous pouvez passer un objet contenant la propriété suivante :

Propriété	Type	Description
<i>context</i>	Text	Nom du contexte d'optimisation appliqué à l'entity selection.

Exemple

```
var $employees : cs.EmployeeSelection
ALL RECORDS([Employee])
$employees:=Create entity selection([Employee])
// L'entity selection $employees contient maintenant un ensemble de
// références vers toutes les entités de la dataclass Employee
```

Voir aussi

[dataClass.newSelection\(\)](#)

USE ENTITY SELECTION

USE ENTITY SELECTION (*entitySelection*)

Paramètres	Type		Description
<i>entitySelection</i>	EntitySelection	->	Une entity selection

Description

La commande `USE ENTITY SELECTION` met à jour la sélection courante de la table correspondant à la dataclass du paramètre `entitySelection`, en fonction du contenu de l'entity selection.

Cette commande ne peut pas être utilisée avec un `datastore distant`.

Après un appel à `USE ENTITY SELECTION`, le premier enregistrement de la sélection courante mise à jour (s'il n'est pas vide) devient l'enregistrement courant, mais il n'est pas chargé en mémoire. Si vous avez besoin d'utiliser les valeurs des champs de l'enregistrement courant, utilisez la commande `LOAD RECORD` après la commande `USE ENTITY SELECTION`.

Exemple

```
var $entitySel : Object  
  
$entitySel:=ds.Employee.query("lastName = :1";"M@") // $entitySel est associé à la dataclass Employee  
REDUCE SELECTION([Employee];0)  
USE ENTITY SELECTION($entitySel) // La sélection courante de la table Employee est mise à jour
```

[index]

► Historique

`[index]` : 4D.Entity

Description

La notation `EntitySelection[index]` vous permet d'accéder aux entités de l'entity selection à l'aide de la syntaxe standard des collections : il vous suffit de passer la position de l'entité à laquelle vous souhaitez accéder dans le paramètre `index`.

A noter que l'entité correspondante est rechargée depuis le datastore.

`index` peut être tout nombre compris entre 0 et `.length -1`.

- Si `index` est en-dehors de ces limites, une erreur est retournée.
- Si `index` correspond à une entité supprimée, la valeur Null est retournée.

Attention : `EntitySelection[index]` est une expression non assignable, ce qui signifie qu'elle ne peut pas être utilisée comme référence modifiable de l'entité avec des fonctions telles que `.lock()` ou `.save()`. Pour travailler avec l'entité correspondante, vous devez assigner l'expression retournée à une expression assignable, comme une variable. Voici quelques exemples :

```
$sel:=ds.Employee.all() //creation de l'entity selection  
//code invalide:  
$result:=$sel[0].lock() //ne fonctionnera PAS  
$sel[0].lastName:="Smith" //ne fonctionnera PAS  
$result:=$sel[0].save() //ne fonctionnera PAS  
//code valide :  
$entity:=$sel[0] //OK  
$entity.lastName:="Smith" //OK  
$entity.save() //OK
```

Exemple

```

var $employees : cs.EmployeeSelection
var $employee : cs.EmployeeEntity
$employees:=ds.Employee.query("lastName = :1";"H@")
$employee:=$employees[2] // La 3e entité de l'entity selection $employees est rechargée depuis le data

```

.attributeName

► Historique

.attributeName : Collection
.attributeName : 4D.EntitySelection

Description

Tout attribut de dataclass peut être utilisé en tant que propriété d'une entity selection afin de retourner une "projection" des valeurs de l'attribut dans l'entity selection. Les valeurs projetées peuvent être une collection ou une nouvelle entity selection, selon le `kind` (`storage` ou `relation`) de l'attribut.

- Si le "kind" de `attributeName` est `storage` : `.attributeName` retourne une collection de valeurs du même type que `attributeName`.
- Si le "kind" de `attributeName` est `relatedEntity` : `.attributeName` retourne une nouvelle entity selection de valeurs liées du même type que `attributeName`. Les doublons sont supprimés (une entity selection non ordonnée est retournée).
- Si le "kind" de `attributeName` est `relatedEntities` : `.attributeName` retourne une nouvelle entity selection de valeurs liées du même type que `attributeName`. Les doublons sont supprimés (une entity selection non ordonnée est retournée).

Lorsqu'un attribut de relation est utilisé comme propriété d'une entity selection, le résultat est toujours une autre entity selection, même si une seule entité est retournée. Lorsqu'un attribut de relation est utilisé comme propriété d'une entity selection, le résultat est toujours une autre entity selection, même si une seule entité est retournée.

Si l'attribut n'existe pas dans l'entity selection, une erreur est retournée.

Exemple 1

Projection de valeurs de stockage :

```

var $firstNames : Collection
$entitySelection:=ds.Employee.all()
$firstNames:=$entitySelection.firstName // firstName est un texte

```

Le résultat est une collection de chaînes, par exemple :

```

[
  "Joanna",
  "Alexandra",
  "Rick"
]

```

Exemple 2

Projection d'entité liée :

```

var $es; $entitySelection : cs.EmployeeSelection
$entitySelection:=ds.Employee.all()
$es:=$entitySelection.employer // employer est lié à la dataclass Company

```

L'objet résultant est une entity selection de la dataclass Company sans doublons (s'il y en a).

Exemple 3

Projection d'entités liées :

```
var $es : cs.EmployeeSelection  
$es:=ds.Employee.all().directReports // directReports est récursif et lié à la dataclass Employee
```

L'objet résultant est une entity selection de la dataclass Employee sans doublons (s'il y en a).

.add()

► Historique

.add(*entity* : 4D.Entity) : 4D.EntitySelection

Paramètres	Type		Description
entity	4D.Entity	->	Entité à ajouter à l'entity selection
Résultat	4D.EntitySelection	->	Entity selection incluant l' <i>entity</i>

Description

La fonction `.add()` ajoute l'*entity* spécifiée à l'entity selection et retourne l'entity selection modifiée.

Les valeurs de type Date sont converties en valeurs numériques (secondes) et utilisées pour calculer la moyenne.

Attention : L'entity selection doit être *altérable*, c'est-à-dire qu'elle a été créée par exemple par `.newSelection()` ou `Create entity selection`, sinon `.add()` retournera une erreur. Les entity selections partageables n'acceptent pas l'ajout d'entités. Pour plus d'informations, reportez-vous au paragraphe [Entity selections partageables ou altérables](#).

- Si l'entity selection est ordonnée, le paramètre *entity* est ajouté à la fin de la sélection. Si une référence à la même entité appartient déjà à l'entity selection, elle est dupliquée et une nouvelle référence est ajoutée.
- Si l'entity selection est non ordonnée, le paramètre *entity* est ajouté n'importe où dans la sélection, sans ordre spécifique.

Pour plus d'informations, reportez-vous au paragraphe [Entity selections triées ou non triées](#).

L'entity selection modifiée est retournée par la fonction, afin que les appels vers la fonction puissent être chaînés.

Une erreur est générée si *entity* et l'entity selection ne sont pas liées à la même dataclass. Si *entity* est Null, aucune erreur n'est générée.

Exemple 1

```
var $employees : cs.EmployeeSelection  
var $employee : cs.EmployeeEntity  
$employees:=ds.Employee.query("lastName = :1";"S@")  
$employee:=ds.Employee.new()  
$employee.lastName:="Smith"  
$employee.save()  
$employees.add($employee) //L'entité $employee est ajoutée à l'entity selection $employees
```

Exemple 2

Les appels vers la fonction peuvent être chaînés :

```

var $sel : cs.ProductSelection
var $p1;$p2;$p3 : cs.ProductEntity

$p1:=ds.Product.get(10)
$p2:=ds.Product.get(11)
$p3:=ds.Product.get(12)
$sel:=ds.Product.query("ID > 50")
$sel:=$sel.add($p1).add($p2).add($p3)

```

.and()

► Historique

.and(*entity* : 4D.Entity) : 4D.EntitySelection
 .and(*entitySelection* : 4D.EntitySelection) : 4D.EntitySelection

Paramètres	Type		Description
entity	4D.Entity	->	Entité à intersecer
entitySelection	4D.EntitySelection	->	Entity selection à intersecer
Résultat	4D.EntitySelection	<-	Entity selection résultante de l'intersection à l'aide de l'opérateur logique ET

Description

La fonction `.and()` combine l'entity selection avec le paramètre *entity* ou *entitySelection* à l'aide de l'opérateur logique ET ; elle retourne une nouvelle entity selection non ordonnée qui ne contient que les entités qui sont référencées à la fois dans l'entity selection et le paramètre.

- Si vous passez *entity* comme paramètre, vous combinez cette entité avec l'entity selection. Si l'entité appartient à l'entity selection, une nouvelle entity selection contenant uniquement l'entité est retournée. Sinon, une entity selection vide est retournée.
- Si vous passez *entitySelection* comme paramètre, vous combinez les deux entity selections. Une nouvelle entity selection contenant uniquement les entités qui sont référencées dans les deux sélections est retournée. S'il n'y a pas d'entité intersectée, une entity selection vide est retournée.

Vous pouvez comparer des [entity selections ordonnées et/ou non ordonnées](#). La sélection résultante est toujours non ordonnée.

Si l'entity selection initiale ou celle du paramètre *entitySelection* est vide, ou si *entity* est Null, une entity selection vide est retournée.

Si l'entity selection initiale et le paramètre ne sont pas liés à la même dataclass, une erreur est retournée.

Exemple 1

```

var $employees; $result : cs.EmployeeSelection
var $employee : cs.EmployeeEntity
$employees:=ds.Employee.query("lastName = :1";"H@")
//l'entity selection $employees contient l'entité
//ayant la clé primaire 710 et d'autres entités
//par ex. "Colin Hetrick" / "Grady Harness" / "Sherlock Holmes" (clé primaire 710)
$employee:=ds.Employee.get(710) // Retourne "Sherlock Holmes"

$result:=$employees.and($employee) // $result est une entity selection contenant
// uniquement l'entité avec la clé primaire 710 ("Sherlock Holmes")

```

Exemple 2

Nous voulons obtenir une sélection d'employés nommés "Jones" qui vivent à New York :

```
var $sel1; $sel2; $sel3 : cs.EmployeeSelection
$sel1:=ds.Employee.query("name =:1";"Jones")
$sel2:=ds.Employee.query("city=:1";"New York")
$sel3:=$sel1.and($sel2)
```

.average()

► Historique

.average(*attributePath* : Text) : Real

Paramètres	Type		Description
attributePath	Text	->	Chemin de l'attribut à utiliser pour le calcul
Résultat	Réel	<-	Moyenne arithmétique des valeurs des entités pour l'attribut (Undefined pour une entity selection vide)

Description

La fonction `.average()` retourne la moyenne arithmétique de toutes les valeurs non nulles de *attributePath* dans l'entity selection.

Passez dans le paramètre *attributePath* le chemin de l'attribut à utiliser pour le calcul.

Seules les valeurs numériques sont utilisées pour le calcul. Notez cependant que, lorsque le *attributePath* de l'entity selection contient des valeurs de types variés, `.average()` tient compte de tous les éléments contenant des valeurs scalaires pour calculer la moyenne.

Les valeurs de type Date sont converties en valeurs numériques (secondes) et utilisées pour calculer la moyenne.

`.average()` retourne undefined si l'entity selection est vide ou si *attributePath* ne contient pas de valeurs numériques.

Une erreur est retournée si :

- *attributePath* est un attribut relatif,
- *attributePath* désigne un attribut qui n'existe pas dans la dataclass de l'entity selection.

Exemple

Nous voulons obtenir la liste des employés dont le salaire est supérieur au salaire moyen :

```
var $averageSalary : Real
var $moreThanAv : cs.EmployeeSelection
$averageSalary:=ds.Employee.all().average("salary")
$moreThanAv:=ds.Employee.query("salary > :1";$averageSalary)
```

.contains()

► Historique

.contains(*entity* : 4D.Entity) : Boolean

Paramètres	Type		Description
entity	4D.Entity	->	Entité à évaluer
Résultat	Booléen	<-	Vrai si l'entité appartient à l'entity selection, sinon Faux

Description

La fonction `.contains()` retourne vrai si la référence d'entité appartient à l'entity selection, et faux sinon.

Dans `entity`, spécifiez l'entité à rechercher dans l'entity selection. Si l'entité est Null, la fonction retournera faux.

Si `entity` et l'entity selection n'appartiennent pas à la même dataclass, une erreur est générée.

Exemple

```

var $employees : cs.EmployeeSelection
var $employee : cs.EmployeeEntity

$employees:=ds.Employee.query("lastName=:1";"H@")
$employee:=ds.Employee.get(610)

If($employees.contains($employee))
    ALERT("L'entité ayant la clé primaire 610 a un nom commençant par H")
Else
    ALERT("L'entité ayant la clé primaire 610 n'a pas un nom commençant par H")
End if

```

.count()

► Historique

`.count(attributePath : Text) : Real`

Paramètres	Type		Description
attributePath	Text	->	Chemin de l'attribut à utiliser pour le calcul
Résultat	Réel	<-	Nombre de valeurs de <code>attributePath</code> non null dans l'entity selection

Description

La fonction `.count()` retourne le nombre d'entités dans l'entity selection pour lesquelles la valeur de `attributePath` n'est pas null.

Seules les valeurs scalaires sont prises en compte. Les valeurs de type objet ou collection sont considérées comme des valeurs null.

Une erreur est renvoyée si :

- `attributePath` est un attribut relatif,
- `attributePath` n'est pas trouvé dans la dataclass de l'entity selection.

Exemple

Nous voulons trouver le nombre total d'employés d'une entreprise sans compter ceux dont l'intitulé du poste n'a pas été défini :

```

var $sel : cs.EmployeeSelection
var $count : Real

$sel:=ds.Employee.query("employer = :1";"Acme, Inc")
$count:=$sel.count("jobtitle")

```

.copy()

► Historique

.copy({ option : Integer }) : 4D.EntitySelection

Paramètres	Type		Description
option	Integer	->	ck shared : retourne une entity selection partageable
Résultat	4D.EntitySelection	<-	Copie de l'entity selection

Description

La fonction `.copy()` retourne une copie de l'entity selection d'origine.

Les entités d'une collection d'entités auxquelles on accède via [] ne sont pas rechargées depuis la base de données.

Par défaut, si le paramètre *option* est omis, la fonction retourne une nouvelle entity selection non partageable (même si la fonction est appliquée à une entity selection partageable). Passez la constante `ck shared` dans le paramètre *option* si vous souhaitez créer une entity selection partageable.

Cette fonction retourne toujours vrai lorsque l'entity selection provient d'un datastore distant.

Exemple

Vous créez une nouvelle entity selection de produits, vide lorsque le formulaire est chargé :

```

Case of
:(Form event code=On Load)
    Form.products:=ds.Products.newSelection()
End case

```

Cette entity selection est ensuite mise à jour avec les produits et vous souhaitez partager les produits entre plusieurs process. Copiez l'entity selection `Form.products` comme sélection partageable:

```

...
// L'entity selection Form.products est mise à jour
Form.products.add(Form.selectedProduct)

Use(Storage)
If(Storage.products=NULL)
    Storage.products:=New shared object()
End if

Use(Storage.products)
Storage.products:=Form.products.copy(ck shared)
End use
End use

```

.distinct()

► Historique

.distinct(*attributePath* : Text { ; *option* : Integer }) : Collection

Paramètres	Type		Description
attributePath	Text	->	Chemin de l'attribut dont vous souhaitez obtenir les valeurs distinctes
option	Integer	->	dk diacritical : évaluation diacritique ("A" # "a" par exemple)
Résultat	Collection	<-	Collection avec seulement les valeurs distinctes

Description

La fonction `.distinct()` renvoie une collection contenant uniquement les valeurs distinctes (différentes) de *attributePath* dans l'entity selection.

La collection renournée est automatiquement triée. Les valeurs Null ne sont pas renvoyées.

Dans le paramètre *attributePath* passez l'attribut d'entité dont vous voulez obtenir les valeurs distinctes. Seules les valeurs scalaires (texte, nombre, booléen ou date) peuvent être gérées. Les types sont renvoyés dans l'ordre suivant : Si *attributePath* est un attribut d'objet qui contient des valeurs de types différents, elles sont groupées par type et triées ensuite.

1. booléens
2. chaînes
3. nombres
4. dates

Vous pouvez utiliser la notation `[]` pour désigner une collection lorsque *attributePath* est un chemin dans un objet (cf. exemples).

Par défaut, une évaluation non diacritique est effectuée. Si vous souhaitez que l'évaluation soit sensible à la casse ou pour différencier des caractères accentués et non-accentués, passez la constante `dk diacritical` dans le paramètre *option*.

Une erreur est retournée si :

- *attributePath* est un attribut relatif,
- *attributePath* n'est pas trouvé dans la dataclass de l'entity selection.

Exemples

Vous souhaitez obtenir une collection contenant un seul élément par nom de pays :

```
var $countries : Collection  
$countries:=ds.Employee.all().distinct("address.country")
```

`nicknames` est une collection et `extra` est un attribut d'objet :

```
$values:=ds.Employee.all().distinct("extra.nicknames[].first")
```

.drop()

► Historique

.drop({ *mode* : Integer }) : 4D.EntitySelection

Paramètres	Type		Description
mode	Integer	->	dk stop dropping on first error : stoppe l'exécution de la fonction au niveau de la première entité non-supprimable
Résultat	4D.EntitySelection	<-	Entity selection vide si exécutée avec succès, sinon entity selection contenant la ou les entité(s) non supprimée(s)

Description

La fonction `.drop()` supprime les entités appartenant à l'entity selection de la table liée à sa dataclass dans le datastore. L'entity selection reste en mémoire.

La suppression d'entités est permanente et ne peut pas être annulée. Il est recommandé d'appeler cette action dans une transaction afin d'avoir une possibilité de récupération.

Si une entité verrouillée est rencontrée lors de l'exécution de `.drop()`, elle n'est pas supprimée. Par défaut, la fonction traite toutes les entités de l'entity selection et renvoie des entités non supprimables dans l'entity selection. Si vous souhaitez que la fonction arrête l'exécution au niveau de la première entité non supprimable rencontrée, passez la constante `dk stop dropping on first error` dans le paramètre `mode`.

Exemple

Exemple sans l'option `dk stop dropping on first error` :

```
var $employees; $notDropped : cs.EmployeeSelection
$employees:=ds.Employee.query("firstName=:1";"S@")
$notDropped:=$employees.drop() // $notDropped est une entity selection contenant toutes les entités non
If($notDropped.length=0) ///La suppression est un succès, toutes les entités ont été supprimées
    ALERT("Vous avez supprimé "+String($employees.length)+" employés") //L'entity selection supprimée re
Else
    ALERT("Problème durant la suppression, réessayez plus tard")
End if
```

Exemple avec l'option `dk stop dropping on first error` :

```
var $employees; $notDropped : cs.EmployeeSelection
$employees:=ds.Employee.query("firstName=:1";"S@")
$notDropped:=$employees.drop(dk stop dropping on first error) // $notDropped est une entity selection co
If($notDropped.length=0) // La suppression est un succès, toutes les entités ont été supprimées
    ALERT("Vous avez supprimé "+String($employees.length)+" employés") // L'entity selection supprimée re
Else
    ALERT("Problème durant la suppression, réessayez plus tard")
End if
```

.extract()

► Historique

`.extract(attributePath : Text { ; option : Integer }) : Collection`

`.extract(attributePath { ; targetPath } { ; ...attributePathN : Text ; targetPathN : Text }) : Collection`

Paramètres	Type		Description
attributePath	Text	->	Chemin d'attribut dont les valeurs doivent être extraites dans la nouvelle collection
targetPath	Text	->	Chemin ou nom d'attribut cible
option	Integer	->	<code>ck keep null</code> : inclure les attributs null dans la collection retournée (ignorés par défaut).
Résultat	Collection	<-	Collection contenant les valeurs extraites

Description

La fonction `.extract()` retourne une collection contenant les valeurs `attributePath` extraites de l'entity selection.

`attributePath` peut désigner :

- un attribut scalaire de dataclass,
- une entité liée,
- des entités liées.

Si `attributePath` est invalide, une collection vide est retournée.

Cette fonction accepte deux syntaxes.

`.extract(attributePath : Text { ; option : Integer }) : Collection`

Avec cette syntaxe, `.extract()` remplit la collection retournée avec des valeurs de `attributePath` de l'entity selection.

Par défaut, les entités pour lesquelles `attributePath` est `null` ou indéfini sont ignorées dans la collection résultante. Vous pouvez passer la constante `ck keep null` dans le paramètre `option` pour intégrer ces valeurs comme des éléments `null` dans la collection retournée.

- Les attributs dataclass avec `.kind = "relatedEntity"` sont extraits sous forme de collection d'entités (les duplications sont conservées).
- Les attributs dataclass avec `.kind = "relatedEntities"` sont extraits sous forme de collection d'entity selections.

`.extract (attributePath ; targetPath { ; ...attributePathN ; ... targetPathN}) : Collection`

Avec cette syntaxe, `.extract()` remplit la collection retournée avec les valeurs de `attributePath`. Chaque élément de la collection retournée est un objet avec les propriétés `targetPath` complétées par les propriétés `attributePath` correspondantes. Les valeurs `null` sont conservées (le paramètre `option` est ignoré avec cette syntaxe).

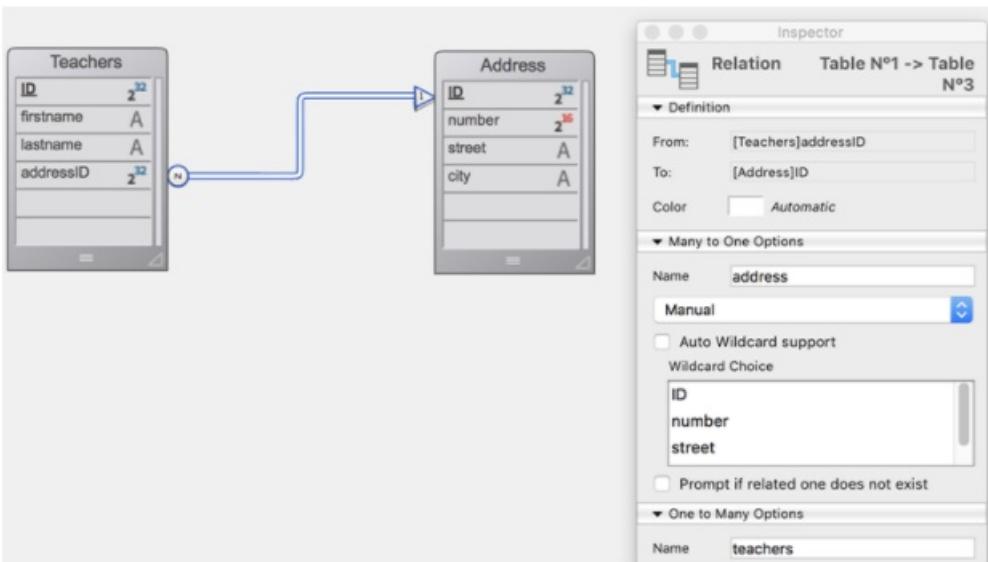
Si plusieurs `attributePath` sont renseignés, un `targetPath` doit être fourni pour chacun. Seules les paires `[attributePath, targetPath]` valides sont extraites.

- Les attributs dataclass avec `.kind = "relatedEntity"` sont extraits sous forme d'entity.
- Les attributs dataclass avec `.kind = "relatedEntities"` sont extraits sous forme d'entity selection.

Les valeurs `Null` sont évaluées comme inférieures aux autres valeurs.

Exemple

Considérons les tables et relations suivantes :



```

var $firstnames; $addresses; $mailing; $teachers : Collection
//
//
// $firstnames est une collection de Chaînes

$firstnames:=ds.Teachers.all().extract("firstname")
//
// $addresses est une collection d'entités liées à la dataclass Address
// Les valeurs null d'Address sont extraites
$addresses:=ds.Teachers.all().extract("address";ck keep null)
//
//
// $mailing est une collection d'objets avec les propriétés "who" et "to"
// Le contenu de la propriété "who" est de type Chaîne
// Le contenu de la propriété "to" est de type entity (dataclass Address)
$mailing:=ds.Teachers.all().extract("lastname";"who";"address";"to")
//
//
// $mailing est une collection d'objets avec les propriétés "who" et "city"
// Le contenu de la propriété "who" est de type Chaîne
// Le contenu de la propriété "city" est de type Chaîne
$mailing:=ds.Teachers.all().extract("lastname";"who";"address.city";"city")
//
// $teachers est une collection d'objets avec les propriétés "where" et "who"
// Le contenu de la propriété "where" est de type Chaîne
// Le contenu de la propriété "who" est une entity selection (dataclass Teachers)
$teachers:=ds.Address.all().extract("city";"where";"teachers";"who")
//
// $teachers est une collection d'entity selections
$teachers:=ds.Address.all().extract("teachers")

```

.first()

► Historique

.first() : 4D.Entity

Paramètres	Type		Description
Résultat	4D.Entity	<-	Référence vers la première entité de l'entity selection (Null si la selection est vide)

Description

La fonction `.first()` retourne une référence vers l'entité en première position dans l'entity selection.

Le résultat de cette fonction est similaire à :

```
$entity:=$entitySel[0]
```

Il existe cependant une différence entre les deux instructions lorsque la sélection est vide :

```
var $entitySel : cs.EmpSelection
var $entity : cs.EmpEntity
$entitySel:=ds.Emp.query("lastName = :1;"Nonexistentname") //aucune entité correspondante
//l'entity selection est alors vide
$entity:=$entitySel.first() //renvoie Null
$entity:=$entitySel[0] //génère une erreur
```

Exemple

```
var $entitySelection : cs.EmpSelection
var $entity : cs.EmpEntity
$entitySelection:=ds.Emp.query("salary > :1";100000)
If($entitySelection.length#0)
    $entity:=$entitySelection.first()
End if
```

.getDataClass()

► Historique

.getDataClass() : 4D.DataClass

Paramètres	Type		Description
Résultat	4D.DataClass	<-	Dataclass à laquelle appartient l'entity selection

Description

La fonction `.getDataClass()` retourne la dataclass de l'entity selection.

Cette fonction est utile principalement dans le contexte de code générique.

Exemple

Le code générique suivant duplique toutes les entités de l'entity selection :

```
//méthode duplicate_entities
//duplicate_entities($entity_selection)

#DECLARE ( $entitySelection : 4D.EntitySelection )
var $dataClass : 4D.DataClass
var $entity; $duplicate : 4D.Entity
var $status : Object
$dataClass:=$entitySelection.getDataClass()
For each($entity;$entitySelection)
    $duplicate:=$dataClass.new()
    $duplicate.fromObject($entity.toObject())
    $duplicate[$dataClass.getInfo().primaryKey]:=Null //réinitialise la clé primaire
    $status:=$duplicate.save()
End for each
```

.getRemoteContextAttributes()

► Historique

.getRemoteContextAttributes() : Text

Paramètres	Type		Description
result	Text	<-	Attributs de contexte associés à l'entity selection, séparés par une virgule

Mode avancé : Cette fonction est destinée aux développeurs qui souhaitent personnaliser les fonctionnalités par défaut de ORDA dans le cadre de configurations spécifiques. Dans la plupart des cas, vous n'aurez pas besoin de l'utiliser.

Description

La fonction `.getRemoteContextAttributes()` retourne des informations relatives au contexte d'optimisation utilisé par l'entity selection.

S'il n'existe pas de [contexte d'optimisation](#) pour l'entity selection, la fonction retourne un texte vide.

Exemple

```
var $ds : 4D.DataStoreImplementation
var $persons : cs.PersonsSelection
var $p : cs.PersonsEntity

var $info : Text
var $text : Text

$ds:=Open datastore(New object("hostname"; "www.myserver.com"); "myDS")

$persons:=$ds.Persons.all()
$text:=""
For each ($p; $persons)
    $text:=$p.firstname+" lives in "+$p.address.city+ " / "
End for each

$info:=$persons.getRemoteContextAttributes()
//$info = "firstname,address,address.city"
```

Voir aussi

[Entity.getRemoteContextAttributes\(\)](#)
[.clearAllRemoteContexts\(\)](#)
[.getRemoteContextInfo\(\)](#)
[.getAllRemoteContexts\(\)](#)
[.setRemoteContextInfo\(\)](#)

.isAlterable()

► Historique

.isAlterable() : Boolean

Paramètres	Type		Description
Résultat	Booléen	<-	Vrai si l'entity selection est modifiable, sinon Faux

Description

La fonction `.isAlterable()` retourne Vrai si l'entity selection est modifiable (alterable), et Faux si elle n'est pas

modifiable.

Pour plus d'informations, voir [Entity selections partageables ou modifiables](#).

Exemple

Vous êtes sur le point d'afficher l'entity selection `Form.products` dans une [list box](#) pour permettre à l'utilisateur d'ajouter de nouveaux produits. Vous voulez vous assurer qu'elle est modifiable afin que l'utilisateur puisse ajouter de nouveaux produits sans erreur :

```
If (Not(Form.products.isAlterable()))
    Form.products:=Form.products.copy()
End if
...
Form.products.add(Form.product)
```

.isOrdered()

► Historique

`.isOrdered()` : Boolean

Paramètres	Type	Description
Résultat	Booléen	<- Vrai si l'entity selection est triée, sinon Faux

Description

La fonction `.isOrdered()` retourne Vrai si l'entity selection est triée, et Faux si elle est non triée.

Cette fonction ne modifie pas l'entity selection d'origine.

Pour plus d'informations, voir [Entity selections triées vs Entity selections non-triées](#).

Exemple

```
var $employees : cs.EmployeeSelection
var $employee : cs.EmployeeEntity
var $isOrdered : Boolean
$employees:=ds.Employee.newSelection(dk keep ordered)
$employee:=ds.Employee.get(714) // // renvoie l'entité avec clé primaire 714

//Dans une entity selection triée, vous pouvez ajouter la même entité plusieurs fois (les duplications
$employees.add($employee)
$employees.add($employee)
$employees.add($employee)

$isOrdered:=$employees.isOrdered()
If($isOrdered)
    ALERT("L'entity selection est triée et contient "+String($employees.length)+" employés")
End if
```

.last()

► Historique

`.last()` : 4D.Entity

Paramètres	Type		Description
Résultat	4D.Entity	<-	Référence vers la dernière entité de l'entity selection (Null si l'entity selection est vide)

Description

La fonction `.last()` retourne une référence vers l'entité en dernière position dans l'entity selection.

Le résultat de cette fonction est similaire à :

```
$entity:=$entitySel[length-1]
```

Si l'entity selection est vide, la fonction renvoie Null.

Exemple

```
var $entitySelection : cs.EmpSelection
var $entity : cs.EmpEntity
$entitySelection:=ds.Emp.query("salary < :1";50000)
If($entitySelection.length#0)
    $entity:=$entitySelection.last()
End if
```

.length

► Historique

`.length` : Integer

Description

La propriété `.length` retourne le nombre d'entités dans l'entity selection. Si l'entity selection est vide, elle contient 0.

Les entity selections ont toujours une propriété `.length`.

Les entités d'une collection d'entités auxquelles on accède via [] ne sont pas rechargées depuis la base de données.

Exemple

```
var $vSize : Integer
$vSize:=ds.Employee.query("gender = :1";"male").length
ALERT(String(vSize)+" employés masculins trouvés.")
```

.max()

► Historique

`.max(attributePath : Text) : any`

Paramètres	Type		Description
attributePath	Text	->	Chemin de l'attribut à utiliser pour le calcul
Résultat	any	<-	Valeur la plus haute de l'attribut

Description

La fonction `.max()` retourne la plus haute valeur (ou valeur maximale) parmi toutes les valeurs de `attributePath` dans l'entity selection. Autrement dit, elle retourne la valeur de la dernière entité de la sélection si elle était triée par ordre croissant avec la fonction `.orderBy()`.

Si vous passez dans `attributePath` le chemin d'un attribut objet contenant des valeurs de différents types, la fonction `.max()` retournera la valeur maximale du premier type de valeur scalaire dans l'ordre par défaut de la liste des types 4D (voir `.sort()`).

`.max()` retourne undefined si l'entity selection est vide ou si `attributePath` n'est pas trouvé dans l'attribut objet.

Une erreur est retournée si :

- `attributePath` est un attribut relatif,
- `attributePath` désigne un attribut qui n'existe pas dans la dataclass de l'entity selection.

Exemple

Nous souhaitons connaître le salaire le plus élevé parmi les employées :

```
var $sel : cs.EmpSelection
var $maxSalary : Real
$sel:=ds.Employee.query("gender = :1";"female")
$maxSalary:=$sel.max("salary")
```

.min()

► Historique

`.min(attributePath : Text) : any`

Paramètres	Type		Description
attributePath	Text	->	Chemin de l'attribut à utiliser pour le calcul
Résultat	any	<-	Valeur la plus basse de l'attribut

Description

La fonction `.min()` retourne la plus faible valeur (ou valeur minimale) parmi toutes les valeurs de `attributePath` dans l'entity selection. Autrement dit, elle retourne la valeur de la première entité de la sélection si elle était triée par ordre croissant avec la fonction `.orderBy()` (hors valeurs Null).

Si vous passez dans `attributePath` le chemin d'un attribut objet contenant des valeurs de différents types, la fonction `.min()` retournera la valeur minimale du premier type de valeur scalaire dans l'ordre par défaut de la liste des types 4D (voir `.sort()`).

`.min()` retourne undefined si l'entity selection est vide ou si `attributePath` n'est pas trouvé dans l'attribut objet.

Une erreur est retournée si :

- `attributePath` est un attribut relatif,
- `attributePath` désigne un attribut qui n'existe pas dans la dataclass de l'entity selection.

Exemple

Nous souhaitons connaître le salaire le plus bas parmi les employées :

```
var $sel : cs.EmpSelection
var $minSalary : Real
$sel:=ds.Employee.query("gender = :1";"female")
$minSalary:=$sel.min("salary")
```

.minus()

► Historique

.minus(*entity* : 4D.Entity) : 4D.EntitySelection

.minus(*entitySelection* : 4D.EntitySelection) : 4D.EntitySelection

Paramètres	Type		Description
entity	4D.Entity	->	Entité à soustraire
entitySelection	4D.EntitySelection	->	Entity selection à soustraire
Résultat	4D.EntitySelection	<-	Nouvelle entity selection ou une nouvelle référence sur l'entity selection existante

Description

La fonction `.minus()` exclut de l'entity selection à laquelle elle est appliquée l'*entity* ou les entités de l'*entitySelection* et retourne l'entity selection résultante.

- Si vous passez *entity* en paramètre, la fonction crée une nouvelle entity selection sans *entity* (si *entity* appartient à l'entity selection). Si *entity* n'était pas incluse dans l'entity selection d'origine, une nouvelle référence à l'entity selection est renvoyée.
- Si vous passez *entitySelection* en paramètre, la fonction retourne une entity selection contenant les entités appartenant à l'entity selection d'origine, sans les entités appartenant à *entitySelection*.

Vous pouvez comparer des [entity selections ordonnées et/ou non ordonnées](#). La sélection résultante est toujours non ordonnée.

Si l'entity selection initiale ou l'entity selection initiale et celle du paramètre *entitySelection* sont vides, une entity selection vide est retournée.

Si *entitySelection* est vide ou si *entity* est Null, une nouvelle référence à l'entity selection d'origine est renvoyée.

Si l'entity selection initiale et le paramètre ne sont pas liés à la même dataclass, une erreur est retournée.

Exemple 1

```
var $employees; $result : cs.EmployeeSelection
var $employee : cs.EmployeeEntity

$employees:=ds.Employee.query("lastName = :1";"H@")
// l'entity selection $employees contient l'entité avec la clé primaire 710 ainsi que d'autres entités
// par ex. "Colin Hetrick", "Grady Harness", "Sherlock Holmes" (clé primaire 710)

$employee:=ds.Employee.get(710) // "Sherlock Holmes"

$result:=$employees.minus($employee) // $result contient "Colin Hetrick", "Grady Harness"
```

Exemple 2

Vous voulez avoir une sélection d'employées nommées "Jones" qui vivent à New York :

```
var $sel1; $sel2; $sel3 : cs.EmployeeSelection
$sel1:=ds.Employee.query("name =:1";"Jones")
$sel2:=ds.Employee.query("city=:1";"New York")
$sel3:=$sel1.and($sel2).minus(ds.Employee.query("gender='male'))
```

.or()

► Historique

```
.or( entity : 4D.Entity ) : 4D.EntitySelection  
.or( entitySelection : 4D.EntitySelection ) : 4D.EntitySelection
```

Paramètres	Type		Description
entity	4D.Entity	->	Entité à intersecer
entitySelection	4D.EntitySelection	->	Entity selection à intersecer
Résultat	4D.EntitySelection	<-	Nouvelle entity selection ou nouvelle référence à l'entity selection d'origine

Description

La fonction `.or()` combine l'entity selection avec le paramètre `entity` ou `entitySelection` en utilisant l'opérateur OU logique (non exclusif); elle retourne une nouvelle entity selection non triée contenant toutes les entités de l'entity selection et du paramètre.

- Si vous passez `entity` comme paramètre, vous comparez cette entité avec l'entity selection. Si l'entité appartient à l'entity selection, une nouvelle référence à l'entity selection est retournée. Sinon, une nouvelle entity selection contenant l'entity selection d'origine et l'entité est retournée.
- Si vous passez `entitySelection` comme paramètre, vous comparez les deux entity selections. Une nouvelle entity selection contenant les entités appartenant à la sélection d'entités d'origine ou à `entitySelection` est renvoyée (OU n'est pas exclusif, les entités référencées dans les deux sélections ne sont pas dupliquées dans la sélection résultante).

Vous pouvez comparer des [entity selections ordonnées et/ou non ordonnées](#). La sélection résultante est toujours non ordonnée.

Si l'entity selection initiale et celle du paramètre `entitySelection` sont vides, une entity selection vide est retournée. Si l'entity selection d'origine est vide, une référence à `entitySelection` ou une entity selection contenant uniquement `entity` est retournée.

Si `entitySelection` est vide ou si `entity` est Null, une nouvelle référence à l'entity selection d'origine est renvoyée.

Si l'entity selection initiale et le paramètre ne sont pas liés à la même dataclass, une erreur est retournée.

Exemple 1

```
var $employees1; $employees2; $result : cs.EmployeeSelection  
$employees1:=ds.Employee.query("lastName = :1";"H@") // "Colin Hetrick", "Grady Harness"  
$employees2:=ds.Employee.query("firstName = :1";"C@") // "Colin Hetrick", "Cath Kidston"  
$result:=$employees1.or($employees2) // $result contient "Colin Hetrick", "Grady Harness", "Cath Kidston"
```

Exemple 2

```
var $employees; $result : cs.EmployeeSelection  
var $employee : cs.EmployeeEntity  
$employees:=ds.Employee.query("lastName = :1";"H@") // "Colin Hetrick", "Grady Harness", "Sherlock Holmes"  
$employee:=ds.Employee.get(686) // l'entité avec clé primaire 686 n'appartient pas à l'entity selection  
// elle correspond à l'employée "Mary Smith"  
  
$result:=$employees.or($employee) // $result contient "Colin Hetrick", "Grady Harness", "Sherlock Holmes"
```

.orderBy()

► Historique

.orderBy(*pathString* : Text) : 4D.EntitySelection

.orderBy(*pathObjects* : Collection) : 4D.EntitySelection

Paramètres	Type		Description
<i>pathString</i>	Text	->	Chemin(s) d'attribut(s) et mode(s) de tri pour l'entity selection
<i>pathObjects</i>	Collection	->	Collection d'objets critère
Résultat	4D.EntitySelection	<-	Nouvelle entity selection dans l'ordre spécifié

Description

La fonction `.orderBy()` renvoie une nouvelle entity selection triée contenant toutes les entités de l'entity selection dans l'ordre spécifié par le paramètre *pathString* ou *pathObjects*.

- Cette fonction ne modifie pas l'entity selection d'origine.
- Pour plus d'informations, reportez-vous au paragraphe [Entity selections triées ou non triées](#).

Vous devez utiliser un paramètre de critère pour définir la manière dont les entités doivent être triées. Deux paramètres différents sont pris en charge :

- *pathString* (Texte) : Ce paramètre contient une formule composée de chemins d'attribut de 1 à n et (optionnellement) de tri, séparés par des virgules. La syntaxe est :

```
"attributePath1 {desc or asc}, attributePath2 {desc or asc},..."
```

L'ordre dans lequel les attributs sont passés détermine la priorité de tri des entités. Par défaut, les attributs sont triés par ordre croissant. Vous pouvez définir l'ordre de tri de chaque propriété dans la formule de critère, séparée du chemin de propriété par un simple espace : passez "asc" pour trier par ordre croissant ou "desc" pour un ordre décroissant.

- *pathObjects* (collection) : chaque élément de la collection contient un objet structuré de la façon suivante :

```
{
  "propertyPath": string,
  "descending": boolean
}
```

Par défaut, les attributs sont triés par ordre croissant ("descending" est false).

Vous pouvez ajouter autant d'objets que nécessaire dans la collection de critères.

Cette fonction est utilisable uniquement avec un datastore distant (client/serveur ou connexion [Open datastore](#)).

Exemple

```

// tri avec formule
$sortedEntitySelection:=$entitySelection.orderBy("firstName asc, salary desc")
$sortedEntitySelection:=$entitySelection.orderBy("firstName")

// tri avec collection avec ou sans ordres de tri
$orderColl:=New collection
$orderColl.push(New object("propertyPath";"firstName";"descending";False))
$orderColl.push(New object("propertyPath";"salary";"descending";True))
$sortedEntitySelection:=$entitySelection.orderBy($orderColl)

$orderColl:=New collection
$orderColl.push(New object("propertyPath";"manager.lastName"))
$orderColl.push(New object("propertyPath";"salary"))
$sortedEntitySelection:=$entitySelection.orderBy($orderColl)

```

.orderByFormula()

► Historique

`.orderByFormula(formulaString : Text { ; sortOrder : Integer } { ; settings : Object}) : 4D.EntitySelection`
`.orderByFormula(formulaObj : Object { ; sortOrder : Integer } { ; settings : Object}) : 4D.EntitySelection`

Paramètres	Type		Description
formulaString	Text	->	Chaîne formule
formulaObj	Object	->	Objet formule
sortOrder	Integer	->	dk ascending (par défaut) ou dk descending
settings	Object	->	Paramètre(s) de la formule
Résultat	4D.EntitySelection	<-	Nouvelle entity selection triée

Description

La fonction `.orderByFormula()` renvoie une nouvelle entity selection triée contenant toutes les entités de l'entity selection dans l'ordre spécifié par le paramètre `formulaString` ou `formulaObj` et, éventuellement, les paramètres `sortOrder` et `settings`.

Les entités d'une collection d'entités auxquelles on accède via [] ne sont pas rechargées depuis la base de données.

Vous pouvez utiliser soit le paramètre `formulaString`, soit le paramètre `formulaObj` :

- `formulaString` : passez une expression 4D telle que "Year of(this.birthDate)".
- `formulaObj` : passez un objet formule valide créé à l'aide de la commande `Formula` ou `Formula from string`.

La formule de `formulaString` ou `formulaObj` est exécutée pour chaque entité de l'entity selection et son résultat sert à définir la position de l'entité dans l'entity selection retournée. Le résultat doit être de type triable (booléen, date, numérique, texte, heure, null).

Un résultat null est toujours la plus petite valeur.

Par défaut, si vous omettez le paramètre `sortOrder`, l'entity selection résultante est triée par ordre croissant. Vous pouvez optionnellement passer l'une des valeurs suivantes dans le paramètre `sortOrder` :

Constante	Valeur	Commentaire
dk ascending	0	Ordre de tri croissant (défaut)
dk descending	1	Ordre de tri décroissant

Dans `formulaString` et `formulaObj`, l'entité qui est traitée ainsi que ses attributs sont disponibles via la commande `This` (par exemple, `This.lastName`).

Vous pouvez passer un ou plusieurs paramètre(s) à la formule à l'aide de la propriété `args` (objet) du paramètre `settings` : la formule reçoit l'objet `settings.args` dans \$1.

Exemple 1

Pour effectuer un tri d'étudiants à l'aide d'une formule texte :

```
var $es1; $es2 : cs.StudentsSelection
$es1:=ds.Students.query("nationality=:1";"French")
$es2:=$es1.orderByFormula("length(this.lastname)") //croissant par défaut
$es2:=$es1.orderByFormula("length(this.lastname)";dk descending)
```

Pour effectuer le même tri dans le même ordre, mais à l'aide d'un objet formule :

```
var $es1; $es2 : cs.StudentsSelection
var $formula : Object
$es1:=ds.Students.query("nationality=:1";"French")
$formula:=Formula(Length(This.lastname))
$es2:=$es1.orderByFormula($formula) // croissant par défaut
$es2:=$es1.orderByFormula($formula;dk descending)
```

Exemple 2

Une formule est donnée sous forme d'objet formule avec des paramètres; l'objet `settings.args` est reçu dans \$1 dans la méthode `computeAverage`.

Dans cet exemple, le champ objet "marks" de la dataclass `Students` contient les notes des étudiants dans chaque matière. Un seul objet formule est utilisé pour calculer la note moyenne des étudiants à l'aide de différents coefficients pour `schoolA` et `schoolB`.

```
var $es1; $es2 : cs.StudentsSelection
var $formula; $schoolA; $schoolB : Object
$es1:=ds.Students.query("nationality=:1";"French")
$formula:=Formula(computeAverage($1))

$schoolA:=New object() //objet settings
$schoolA.args:=New object("english";1;"math";1;"history";1) // Coefficients permettant de calculer la m

//Trier les étudiants en fonction du critère schoolA
$es2:=$es1.entitySelection.orderByFormula($formula;$schoolA)

$schoolB:=New object() //objet settings
$schoolB.args:=New object("english";1;"math";2;"history";3) // Coefficients permettant de calculer une

//Trier les étudiants en fonction du critère schoolB
$es2:=$es1.entitySelection.orderByFormula($formula;dk descending;$schoolB)
```

```

// 
// méthode computeAverage
// -----
#DECLARE ($coefList : Object) -> $result : Integer
var $subject : Text
var $average; $sum : Integer

$average:=0
$sum:=0

For each($subject;$coefList)
    $sum:=$sum+$coefList[$subject]
End for each

For each($subject;This.marks)
    $average:=$average+(This.marks[$subject]*$coefList[$subject])
End for each

$result:=$average/$sum

```

.query()

► Historique

.query(*queryString* : Text { ; ...*value* : any } { ; *querySettings* : Object }) : 4D.EntitySelection
 .query(*formula* : Object { ; *querySettings* : Object }) : 4D.EntitySelection

Paramètres	Type		Description
<i>queryString</i>	Text	->	Critères de recherche en texte
<i>formula</i>	Object	->	Critères de recherche en objet formule
<i>value</i>	any	->	Valeur(s) à utiliser comme placeholder(s)
<i>querySettings</i>	Object	->	Options de recherche : parameters, attributes, args, allowFormulas, context, queryPath, queryPlan
Résultat	4D.EntitySelection	<-	Nouvelle entity selection composée d'entités issues de l'entity selection répondant aux critères de recherche spécifiés dans <i>queryString</i> ou <i>formula</i>

Description

La fonction `.query()` recherche les entités répondant aux critères de recherche spécifiés dans *queryString* ou *formula* et (optionnellement) dans *value* toutes les entités de l'entity selection, et renvoie un nouvel objet de type `EntitySelection` contenant toutes les entités trouvées. Le mode lazy loading est appliqué.

Les entités d'une collection d'entités auxquelles on accède via `[]` ne sont pas rechargées depuis la base de données.

Si aucune entité correspondante n'est trouvée, une `EntitySelection` vide est retournée.

Si aucune entité correspondante n'est trouvée, une `EntitySelection` vide est retournée.

Par défaut, si vous omettez la déclaration `order by` dans *queryString*, l'entity selection retournée n'est **pas triée**. A noter cependant qu'en client/server, elle se comporte comme une entity selection triée (les entités sont ajoutées à la fin de la sélection).

Exemple 1

```
var $entitySelectionTemp : cs.EmployeeSelection
$entitySelectionTemp:=ds.Employee.query("lastName = :1";"M@")
Form.emps:=$entitySelectionTemp.query("manager.lastName = :1";"S@")
```

Exemple 2

Pour plus d'informations sur la génération d'une requête à l'aide des paramètres `queryString`, `value`, et `querySettings`, reportez-vous à la description de la fonction de dataclass [.query\(\)](#).

Voir aussi

Vous trouverez plus d'exemples de requêtes dans la page [.query\(\)](#).

.queryPath

► Historique

.queryPath : Text

Description

La propriété `.queryPath` contient la description détaillée du chemin de recherche réel utilisé par 4D. Cette propriété est disponible pour les objets de type `EntitySelection` générés via des recherches si la propriété `"queryPath":true` a été passée dans le paramètre `querySettings` de la fonction [.query\(\)](#).

Pour plus d'informations, veuillez vous reporter au paragraphe `querySettings` de la fonction de dataclass [.query\(\)](#).

.queryPlan

► Historique

.queryPlan : Text

Description

La propriété `.queryPlan` contient la description détaillée du plan de recherche prévu par 4D avant de l'exécuter. Cette propriété est disponible pour les objets de type `EntitySelection` générés via des recherches si la propriété `"queryPlan":true` a été passée dans le paramètre `querySettings` de la fonction [.query\(\)](#).

Pour plus d'informations, veuillez vous reporter au paragraphe `querySettings` de la fonction de dataclass [.query\(\)](#).

.refresh()

► Historique

.refresh()

Paramètres	Type		Description		-----		----		::	-----			Ne requiert aucun paramètre
------------	------	--	-------------	--	-------	--	------	--	----	-------	--	--	-----------------------------

Description

Cette fonction est utilisable uniquement avec un datastore distant (client/serveur ou connexion [Open datastore](#)).

La fonction `.refresh()` "invalidise" immédiatement les données de l'entity selection dans le cache local d'ORDA de sorte que, la prochaine fois que 4D accède à l'entity selection, elle soit rechargée à partir de la base.

Par défaut, le cache local d'ORDA est invalidé après 30 secondes. Dans le contexte des applications client/serveur à l'aide d'ORDA et du langage classique, cette fonction vous permet d'être certain que l'application distante fonctionne toujours avec les données les plus récentes.

Exemple 1

Dans cet exemple, les langages classiques et ORDA modifient simultanément les mêmes données :

```
//Sur un 4D distant

var $selection : cs.StudentsSelection
var $student : cs.StudentsEntity

$selection:=ds.Students.query("lastname=:1";"Collins")
//La première entité est chargée dans le cache d'ORDA
$student:=$selection.first()

//Mise à jour avec un 4D classique, le cache ORDA cache n'en est pas informé
QUERY([Students];[Students]lastname="Collins")
[Students]lastname:="Colin"
SAVE RECORD([Students])

//pour obtenir la dernière version, le cache d'ORDA doit être invalidé
$selection.refresh()
// Même si le cache n'a pas expiré, la première entité est rechargée à partir du disque
$student:=$selection.first()

//$student.lastname contient "Colin"
```

Exemple 2

Dans cet exemple, les langages classiques et ORDA modifient simultanément les mêmes données :

```
// Methode formulaire:
Case of
:(Form event code=On Load)
    Form.students:=ds.Students.all()
End case
//
//
// Sur client #1, l'utilisateur charge, met à jour et sauvegarde la première entité
// Sur client #2, l'utilisateur charge, met à jour et sauvegarde la même entité
//
//
// Sur client #1:
Form.students.refresh() // Invalidate le cache ORDA pour l'entity selection Form.students
// Le contenu de la list box est rafraîchi à partir de la base avec les mises à jour effectuées par le
```

.selected()

► Historique

.selected(*selectedEntities* : 4D.EntitySelection) : Object

Paramètres	Type		Description
selectedEntities	4D.EntitySelection	->	Entity selection avec des entités dont il faut connaître le rang dans l'entity selection
Résultat	Object	<-	Plage(s) d'entités sélectionnées dans l'entity selection

Description

La fonction `.selected()` retourne un objet décrivant la ou les positions de *selectedEntities* dans l'entity selection d'origine.

Les entités d'une collection d'entités auxquelles on accède via [] ne sont pas rechargées depuis la base de données.

Passez, dans le paramètre `selectedEntities` une entity selection contenant des entités dont vous souhaitez connaître la position dans l'entity selection d'origine. `selectedEntities` doit être une entity selection appartenant à la même dataclass que l'entity selection d'origine, sinon une erreur 1587 - "La sélection d'entités provient d'une dataclass incompatible" est générée.

Résultat

L'objet retourné contient les propriétés suivantes :

Propriété	Type	Description
<code>ranges</code>	Collection	Collection d'objets plage
<code>ranges[]</code> .start	Integer	Indice de la première entité de la plage
<code>ranges[]</code> .end	Integer	Indice de la dernière entité de la plage

Si une propriété `ranges` contient une seule entité, `start` = `end`. L'indice démarre à 0.

La fonction retourne une collection vide dans la propriété `ranges` si l'entity selection d'origine ou l'entity selection `selectedEntities` est vide.

Exemple

```
var $invoices; $cashSel; $creditSel : cs.Invoices
var $result1; $result2 : Object

$invoices:=ds.Invoices.all()

$cashSelection:=ds.Invoices.query("payment = :1"; "Cash")
$creditSel:=ds.Invoices.query("payment IN :1"; New collection("Cash"; "Credit Card"))

$result1:=$invoices.selected($cashSelection)
$result2:=$invoices.selected($creditSel)

//$result1 = {ranges: [{start:0;end:0},{start:3;end:3},{start:6;end:6}]}
//$result2 = {ranges: [{start:0;end:1},{start:3;end:4},{start:6;end:7}]}
```

.slice()

► Historique

`.slice(startFrom : Integer { ; end : Integer }) : 4D.EntitySelection`

Paramètres	Type		Description
<code>startFrom</code>	Integer	->	Position à laquelle démarrer l'opération (inclus)
<code>end</code>	Integer	->	Position de fin (non inclus)
Résultat	4D.EntitySelection	<-	Nouvelle entity selection contenant les entités copiées (shallow copy)

Description

La fonction `.slice()` retourne une partie d'une entity selection dans une nouvelle entity selection, sélectionnée de l'index `startFrom` à l'index de `fin` (`fin` non inclus) ou à la dernière entité de l'entity selection. Cette fonction effectue une shallow copy (copie superficielle) de l'entity selection (les mêmes références d'entités sont utilisées).

Les entités d'une collection d'entités auxquelles on accède via [] ne sont pas rechargées depuis la base de données.

L'entity selection retournée contient les entités comprises entre l'entité désignée par *startFrom* et, sans la contenir, celle désignée par *end*. Si seul le paramètre *startFrom* est spécifié, l'entity selection retournée contient toutes les entités entre *startFrom* et la dernière entité de l'entity selection d'origine.

- Si *startFrom* < 0, il est recalculé comme *startFrom:=startFrom+length* (il est considéré comme partant de la fin de l'entity selection). Si la valeur calculée est négative, *startFrom* prend la valeur 0.
- Si *startFrom* >= *length*, la fonction retourne une entity selection vide.
- Si *end* < 0, le paramètre est recalculé comme *end:=end+length*.
- Si *end* < *startFrom* (valeurs passées ou recalculées), la fonction ne fait rien.

Si l'entity selection contient des entités qui ont été supprimées entre-temps, elles sont également retournées.

Exemple 1

Si l'entity selection contient des entités qui ont été supprimées entre-temps, elles sont également retournées.

```
var $sel; $sliced : cs.EmployeeSelection  
$sel:=ds.Employee.query("salary > :1";50000)  
$sliced:=$sel.slice(0;9)
```

Exemple 2

Vous souhaitez obtenir une sous-sélection des 9 premières entités de l'entity selection :

```
var $slice : cs.EmployeeSelection  
$slice:=ds.Employee.all().slice(-1;-2) //tente de retourner les entités de position 9 à 8, mais comme 9
```

.sum()

► Historique

.sum(*attributePath* : Text) : Real

Paramètres	Type		Description
<i>attributePath</i>	Text	->	Chemin de l'attribut à utiliser pour le calcul
Résultat	Réel	<-	Somme des valeurs de l'entity selection

Description

La fonction `.toCollection()` crée et retourne une collection dans laquelle chaque élément est un objet contenant un ensemble de propriétés et de valeurs correspondant aux noms et valeurs d'attributs de l'entity selection.

La fonction `.sum()` retourne la somme de toutes les valeurs d'*attributePath* dans l'entity selection.

La somme peut uniquement être effectuée sur des valeurs numériques. Si *attributePath* est de type objet, seules les valeurs numériques qu'il contient seront prises en compte (les autres types de valeurs sont ignorés). Dans ce cas, si *attributePath* désigne une propriété qui n'existe pas dans l'objet ou qui ne contient pas de valeurs numériques, `.sum()` retourne 0.

Une erreur est retournée si :

- *attributePath* est un attribut qui n'est ni de type numérique ni de type objet,
- *attributePath* est un attribut relatif,

- *attributePath* n'est pas trouvé dans la dataclass de l'entity selection.

Exemple

```
var $sel : cs.EmployeeSelection
var $sum : Real

$sel:=ds.Employee.query("salary < :1";20000)
$sum:=$sel.sum("salary")
```

.toCollection()

► Historique

.toCollection({ options : Integer { ; begin : Integer { ; howMany : Integer } } }) : Collection
.toCollection(filterString : Text {; options : Integer { ; begin : Integer { ; howMany : Integer } } }) : Collection
.toCollection(filterCol : Collection {; options : Integer { ; begin : Integer { ; howMany : Integer } } }) : Collection

Paramètres	Type		Description
filterString	Text	->	Chemin(s) d'attribut(s) à extraire
filterCol	Collection	->	Collection d'attribut(s) à extraire
options	Integer	->	dk with primary key : ajoute la clé primaire dk with stamp : ajoute le marqueur
begin	Integer	->	Désigne la position de début
howMany	Integer	->	Nombre d'entités à extraire
Résultat	Collection	<-	Collection d'objets contenant les attributs et valeurs de l'entity selection

Description

La fonction `.toCollection()` crée et retourne une collection dans laquelle chaque élément est un objet contenant un ensemble de propriétés et de valeurs correspondant aux noms et valeurs d'attributs de l'entity selection.

Si aucun paramètre de filtre n'est passé ou si le paramètre contient une chaîne vide ou "*", tous les attributs sont extraits. Les attributs dont la propriété `kind` est "relatedEntity" sont extraits avec la forme simple : un objet avec la propriété `__KEY` (clé primaire). Les attributs dont la propriété "kind" est "relatedEntities" ne sont pas extraits.

Vous pouvez également indiquer les attributs à extraire à l'aide d'un paramètre de filtre. Deux types de filtres sont utilisables :

- *filterString* --une chaîne avec les chemins des propriétés séparés par des virgules : "propertyPath1, propertyPath2, ...".
- *filterCol* --une collection de chaînes contenant des chemins de propriétés : ["propertyPath1","propertyPath2",...]

Si un filtre est spécifié pour un attribut de type `relatedEntity` :

- `propertyPath = "relatedEntity"` -> l'extraction se fait dans une forme simple
- `propertyPath = "relatedEntity.*"` -> toutes les propriétés sont extraites
- `propertyPath = "relatedEntity.propertyName1, relatedEntity.propertyName2, ..."` -> seules ces propriétés sont extraites

Si un filtre est spécifié pour un attribut de type `relatedEntity` :

- `propertyPath = "relatedEntities.*"` -> toutes les propriétés sont extraites
- `propertyPath = "relatedEntities.propertyName1, relatedEntities.propertyName2, ..."` -> seules ces propriétés sont extraites

Si un filtre est spécifié pour un attribut de type `relatedEntities` :

Le paramètre `begin` vous permet d'indiquer la position de départ des entités à extraire. Vous pouvez passer toute valeur

comprise entre 0 et la longueur de l'entity selection -1.

Le paramètre *howMany* vous permet de spécifier le nombre d'entités à extraire, à partir de celle désignée par *begin*. Les entités supprimées ne sont pas retournées mais sont prises en compte dans *howMany*. Par exemple, si *howMany= 3* et s'il y a une entité supprimée, seulement 2 entités sont extraites.

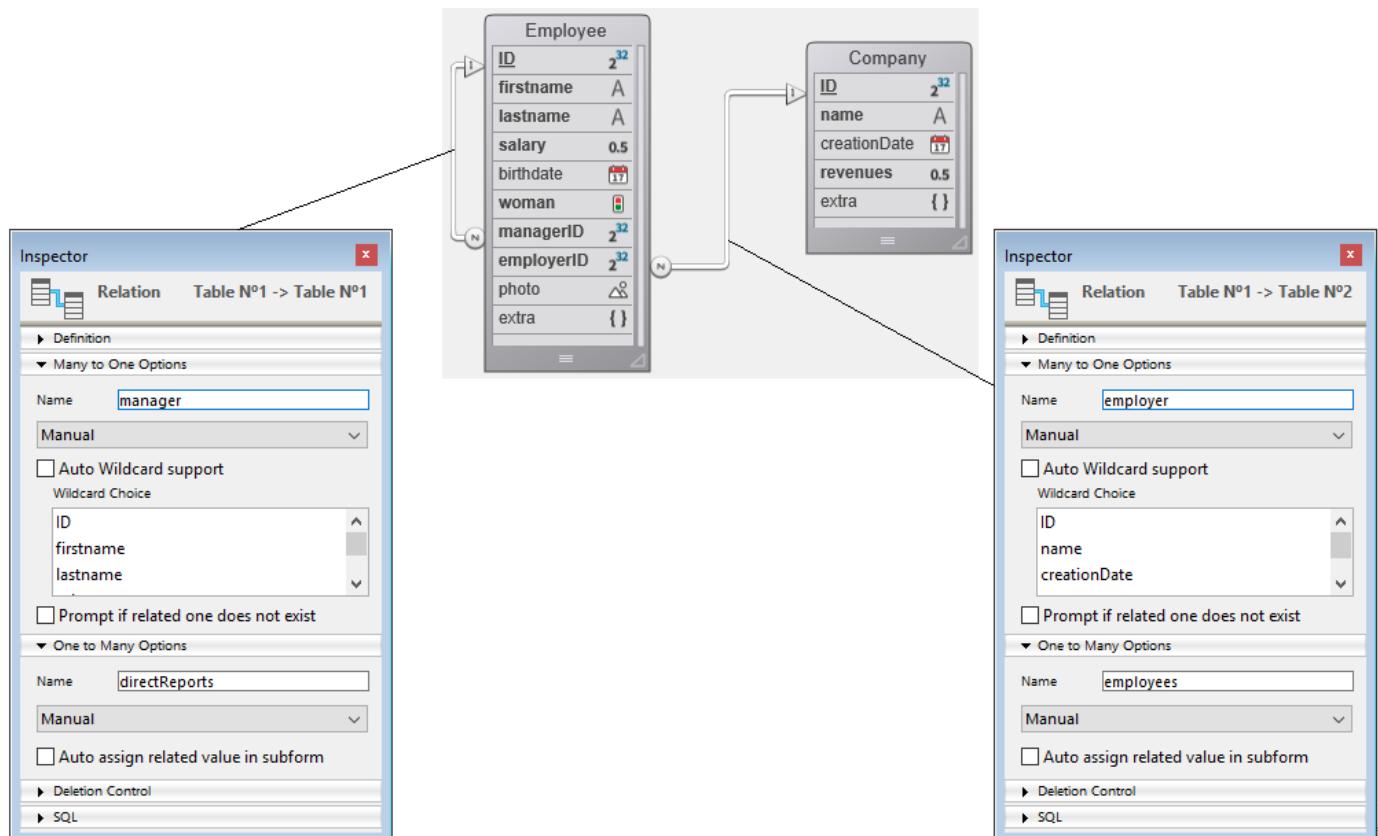
Si le "kind" de *attributeName* est `storage` : `.attributeName` retourne une collection de valeurs du même type que *attributeName*.

Si *howMany > length* de l'entity selection, la fonction retourne (*length - begin*) objects.

- l'entity selection est vide, ou
- *begin* est supérieur à la longueur de l'entity selection.

Exemple 1

La structure suivante sera utilisée pour les exemples de cette section :



Exemple sans paramètres de filtre ni d'options :

```
var $employeesCollection : Collection
var $employees : cs.EmployeeSelection

$employeesCollection:=New collection
$employees:=ds.Employee.all()
$employeesCollection:=$employees.toCollection()
```

Retourne :

```
[
  {
    "ID": 416,
    "firstName": "Gregg",
    "lastName": "Wahl",
    "salary": 79100,
    "birthDate": "1963-02-01T00:00:00.000Z",
    "woman": false,
    "managerID": 412,
    "employerID": 20,
    "photo": "[object Picture]",
    "extra": null,
    "employer": {
      "__KEY": 20
    },
    "manager": {
      "__KEY": 412
    }
  },
  {
    "ID": 417,
    "firstName": "Irma",
    "lastName": "Durham",
    "salary": 47000,
    "birthDate": "1992-06-16T00:00:00.000Z",
    "woman": true,
    "managerID": 412,
    "employerID": 20,
    "photo": "[object Picture]",
    "extra": null,
    "employer": {
      "__KEY": 20
    },
    "manager": {
      "__KEY": 412
    }
  }
]
```

Exemple 2

Retourne :

```
var $employeesCollection : Collection
var $employees : cs.EmployeeSelection

$employeesCollection:=New collection
$employees:=ds.Employee.all()
$employeesCollection:=$employees.toCollection("");dk with primary key+dk with stamp)
```

Retourne :

```
[
  {
    "__KEY": 416,
    "__STAMP": 1,
    "ID": 416,
    "firstName": "Gregg",
    "lastName": "Wahl",
    "salary": 79100,
    "birthDate": "1963-02-01T00:00:00.000Z",
    "woman": false,
    "managerID": 412,
    "employerID": 20,
    "photo": "[object Picture]",
    "extra": null,
    "employer": {
      "__KEY": 20
    },
    "manager": {
      "__KEY": 412
    }
  },
  {
    "__KEY": 417,
    "__STAMP": 1,
    "ID": 417,
    "firstName": "Irma",
    "lastName": "Durham",
    "salary": 47000,
    "birthDate": "1992-06-16T00:00:00.000Z",
    "woman": true,
    "managerID": 412,
    "employerID": 20,
    "photo": "[object Picture]",
    "extra": null,
    "employer": {
      "__KEY": 20
    },
    "manager": {
      "__KEY": 412
    }
  }
]
```

Exemple 3

Retourne :

```
var $employeesCollection; $filter : Collection
var $employees : cs.EmployeeSelection

$employeesCollection:=New collection
$filter:=New collection

$filter.push("firstName")
$filter.push("lastName")

$employees:=ds.Employee.all()
$employeesCollection:=$employees.toCollection($filter;0;0;2)
```

Retourne :

```
[  
  {  
    "firstName": "Gregg",  
    "lastName": "Wahl"  
  },  
  {  
    "firstName": "Irma",  
    "lastName": "Durham"  
  }  
]
```

Exemple 4

Retourne :

```
var $employeesCollection : Collection  
$employeesCollection:=New collection  
$employeesCollection:=$employees.toCollection("firstName,lastName,employer")
```

Exemple avec le type `relatedEntity` avec une forme simple :

```
[  
  {  
    "firstName": "Gregg",  
    "lastName": "Wahl",  
    "employer": {  
      "__KEY": 20  
    }  
  },  
  {  
    "firstName": "Irma",  
    "lastName": "Durham",  
    "employer": {  
      "__KEY": 20  
    }  
  },  
  {  
    "firstName": "Lorena",  
    "lastName": "Boothe",  
    "employer": {  
      "__KEY": 20  
    }  
  }  
]
```

Exemple 5

retourne :

```
var $employeesCollection; $coll : Collection  
$employeesCollection:=New collection  
$coll:=New collection("firstName";"lastName")  
$employeesCollection:=$employees.toCollection($coll)
```

Retourne :

```
[
  {
    "firstName": "Joanna",
    "lastName": "Cabrera"
  },
  {
    "firstName": "Alexandra",
    "lastName": "Coleman"
  }
]
```

Exemple 6

Retourne :

```
var $employeesCollection; $coll : Collection
$employeesCollection:=New collection
$coll:=New collection
$coll.push("firstName")
$coll.push("lastName")
$coll.push("employer.*")
$employeesCollection:=$employees.toCollection($coll)
```

Retourne :

```
[
  {
    "firstName": "Gregg",
    "lastName": "Wahl",
    "employer": {
      "ID": 20,
      "name": "India Astral Secretary",
      "creationDate": "1984-08-25T00:00:00.000Z",
      "revenues": 12000000,
      "extra": null
    }
  },
  {
    "firstName": "Irma",
    "lastName": "Durham",
    "employer": {
      "ID": 20,
      "name": "India Astral Secretary",
      "creationDate": "1984-08-25T00:00:00.000Z",
      "revenues": 12000000,
      "extra": null
    }
  },
  {
    "firstName": "Lorena",
    "lastName": "Boothe",
    "employer": {
      "ID": 20,
      "name": "India Astral Secretary",
      "creationDate": "1984-08-25T00:00:00.000Z",
      "revenues": 12000000,
      "extra": null
    }
  }
]
```

Exemple 7

Retourne :

```
var $employeesCollection : Collection
$employeesCollection:=New collection
$employeesCollection:=$employees.toCollection("firstName, lastName, employer.name")
```

```
[  
  {  
    "firstName": "Gregg",  
    "lastName": "Wahl",  
  
    "employer": {  
      "name": "India Astral Secretary"  
    }  
  },  
  {  
    "firstName": "Irma",  
    "lastName": "Durham",  
    "employer": {  
      "name": "India Astral Secretary"  
    }  
  },  
  {  
    "firstName": "Lorena",  
    "lastName": "Boothe",  
    "employer": {  
      "name": "India Astral Secretary"  
    }  
  }]  
}]
```

Exemple 8

Exemple avec extraction de quelques propriétés de relatedEntity :

```
var $employeesCollection : Collection
$employeesCollection:=New collection
$employeesCollection:=$employees.toCollection("firstName, lastName, directReports.firstName")
```

Retourne :

```
[
  {
    "firstName": "Gregg",
    "lastName": "Wahl",
    "directReports": []
  },
  {
    "firstName": "Mike",
    "lastName": "Phan",
    "directReports": [
      {
        "firstName": "Gary"
      },
      {
        "firstName": "Sadie"
      },
      {
        "firstName": "Christie"
      }
    ]
  },
  {
    "firstName": "Gary",
    "lastName": "Reichert",
    "directReports": [
      {
        "firstName": "Rex"
      },
      {
        "firstName": "Jenny"
      },
      {
        "firstName": "Lowell"
      }
    ]
  }
]
```

Exemple 9

Retourne :

```
var $employeesCollection : Collection
$employeesCollection:=New collection
$employeesCollection:=$employees.toCollection("firstName, lastName, directReports.*")
```

```
[
  {
    "firstName": "Gregg",
    "lastName": "Wahl",
    "directReports": []
  },
  {
    "firstName": "Mike",
    "lastName": "Phan",
    "directReports": [
      {
        "ID": 425,
        "firstName": "Gary",
        "lastName": "Reichert",
        "directReports": []
      }
    ]
  }
]
```

```
        "salary": 65800,
        "birthDate": "1957-12-23T00:00:00.000Z",
        "woman": false,
        "managerID": 424,
        "employerID": 21,
        "photo": "[object Picture]",
        "extra": null,
        "employer": {
            "__KEY": 21
        },
        "manager": {
            "__KEY": 424
        }
    },
    {
        "ID": 426,
        "firstName": "Sadie",
        "lastName": "Gallant",
        "salary": 35200,
        "birthDate": "2022-01-03T00:00:00.000Z",
        "woman": true,
        "managerID": 424,
        "employerID": 21,
        "photo": "[object Picture]",
        "extra": null,
        "employer": {
            "__KEY": 21
        },
        "manager": {
            "__KEY": 424
        }
    }
],
},
{
    "firstName": "Gary",
    "lastName": "Reichert",
    "directReports": [
        {
            "ID": 428,
            "firstName": "Rex",
            "lastName": "Chance",
            "salary": 71600,
            "birthDate": "1968-08-09T00:00:00.000Z",
            "woman": false,

            "managerID": 425,
            "employerID": 21,
            "photo": "[object Picture]",
            "extra": null,
            "employer": {
                "__KEY": 21
            },
            "manager": {
                "__KEY": 425
            }
        },
        {
            "ID": 429,
            "firstName": "Jenny",
            "lastName": "Parks",
            "salary": 51300,
            "birthDate": "1984-05-25T00:00:00.000Z",
            "woman": true,
            "managerID": 425,
            "employer": {
                "__KEY": 21
            },
            "photo": "[object Picture]"
        }
    ]
}
]
```

```
"employerID": 21,
"photo": "[object Picture]",
"extra": null,
"employer": {
    "__KEY": 21
},
"manager": {
    "__KEY": 425
}
}
]
}
```

File

Les objets `File` sont créés avec la commande `File`. Ils contiennent des références à des fichiers du disque qui peuvent exister réellement ou non sur le disque. Par exemple, lorsque vous exécutez la commande `File` pour créer un nouveau fichier, un objet `File` valide est créé mais rien n'est réellement stocké sur le disque jusqu'à ce que vous appviez la fonction `file.create()`.

Exemple

L'exemple suivant crée un fichier de préférences dans le dossier du projet :

```
var $created : Boolean  
$created:=File("//PACKAGE/SpecialPrefs/"+"Current user"+"myPrefs").create()
```

Objet File

`.copyTo(destinationFolder : 4D.Folder { ; newName : Text } { ; overwrite : Integer }) : 4D.File`
copie l'objet `File` (fichier source) vers le `destinationFolder` spécifié

`.create() : Boolean`
crée un fichier sur disque en fonction des propriétés de l'objet `File`

`.createAlias(destinationFolder : 4D.Folder ; aliasName : Text { ; aliasType : Integer }) : 4D.File`
crée un alias (macOS) ou un raccourci (Windows)

`.creationDate : Date`
la date de création du fichier

`.creationTime : Time`
l'heure de création du fichier

`.delete()`
supprime le fichier

`.exists : Boolean`
vrai si le fichier existe sur le disque

`.extension : Text`
l'extension du nom de fichier (le cas échéant)

`.fullName : Text`
le nom complet du fichier, extension comprise (le cas échéant)

`.getAppInfo() : Object`
retourne le contenu d'un fichier d'information .exe, .dll ou .plist sous forme d'objet

`.getContent() : 4D.Blob`
retourne un objet `4D.Blob` contenant l'intégralité du contenu d'un fichier

`.getIcon({ size : Integer }) : Picture`
l'icône du fichier

`notTout / charSetName : Tout / : breakMode : Integer / / / : Tout`

`.getText(charSetName : Text , breakMode : Integer) : Text`

retourne le contenu du fichier en tant que texte

`.hidden : Boolean`

vrai si le fichier est identifié comme "caché" au niveau du système

`.isAlias : Boolean`

vrai si le fichier est un alias, un raccourci ou un lien symbolique

`.isFile : Boolean`

toujours vrai pour un fichier

`.isFolder : Boolean`

toujours faux pour un fichier

`.isWritable : Boolean`

vrai si le fichier existe sur disque et s'il est modifiable

`.modificationDate : Date`

la date de la dernière modification apportée au fichier

`.modificationTime : Time`

l'heure de la dernière modification du fichier

`.moveTo(destinationFolder : 4D.Folder { ; newName : Text }) : 4D.File`

déplace ou renomme l'objet `File` vers le dossier `destinationFolder`

`.name : Text`

le nom du fichier, sans l'extension (le cas échéant)

`.original : 4D.File`

`.original : 4D.Folder`

l'élément cible d'un alias, d'un raccourci ou d'un lien symbolique

`.parent : 4D.Folder`

l'objet dossier parent du fichier

`.path : Text`

le chemin POSIX du fichier

`.platformPath : Text`

le chemin du fichier exprimé dans la syntaxe de la plate-forme

`.rename(newName : Text) : 4D.File`

renomme le fichier avec le nom que vous avez passé dans le paramètre `newName` et retourne l'objet `File` renommé

`.setAppInfo(info : Object)`

écrit les propriétés `info` sous forme de contenu d'information d'un fichier .exe, .dll ou .plist

`.setContent (content : Blob)`

réécrit le contenu intégral du fichier à l'aide des données stockées dans le BLOB `content`

`.setText (text : Text {; charSetName : Text { ; breakMode : Integer } })`

`.setText (text : Text {; charSetNum : Integer { ; breakMode : Integer } })`

.size : Real

la taille du fichier exprimée en octets

File

► Historique

File (*path* : Text { ; *pathType* : Integer }{ ; * }) : 4D.File

File (*fileConstant* : Integer { ; * }) : 4D.File

Paramètres	Type		Description
<i>path</i>	Text	->	Chemin de fichier
<i>fileConstant</i>	Integer	->	Constante de fichier 4D
<i>pathType</i>	Integer	->	fk posix path (par défaut) ou fk platform path
*		->	* pour retourner le fichier de la base hôte
Résultat	4D.File	<-	Nouvel objet fichier

Description

La commande `File` crée et retourne un nouvel objet de type `4D.File`. La commande accepte deux syntaxes :

`File (path { ; pathType } { ; * })`

Dans le paramètre *path*, passez un chemin de fichier. Vous pouvez utiliser une chaîne personnalisée ou un "filesystem" (ex : "/DATA/myfile.txt").

Seuls les noms de chemin absolu sont pris en charge par la commande `File`.

Par défaut, 4D attend un chemin exprimé avec la syntaxe POSIX. Si vous travaillez avec des chemins de plate-forme (Windows ou macOS), vous devez les déclarer à l'aide du paramètre *pathType*. Les constantes suivantes sont disponibles :

Constante	Valeur	Commentaire
fk platform path	1	Chemin exprimé dans une syntaxe spécifique à la plate-forme (obligatoire en cas de chemin de plate-forme)
fk posix path	0	Chemin exprimé avec la syntaxe POSIX (par défaut)

`File (fileConstant { ; * })`

Dans le paramètre *fileConstant*, passez un fichier 4D interne ou un fichier système, à l'aide d'une des constantes suivantes :

Constante	Valeur	Commentaire
Backup history file	19	Fichier d'historique des sauvegardes (voir Fichiers de configuration et de suivi). Stocké dans le dossier de destination de sauvegarde.
Backup log file	13	Fichier journal des sauvegardes courant. Stocké dans le dossier Logs de l'application.
Backup settings file	1	Fichier backup.4DSettings par défaut (format xml), stocké dans le dossier Settings du projet
Backup settings file for data	17	fichier backup.4DSettings du fichier de données (format xml), stocké dans le dossier Settings du dossier data
Build application log file	14	Fichier d'historique courant au format xml du générateur d'application. Stocké dans le dossier Logs.

Log file	Value	Description
Constante Build application settings file	20	Commentaire Fichier de configuration par défaut du générateur d'application ("buildApp.4DSettings"). Stocké dans le dossier Settings du projet.
Compacting log file	6	Fichier d'historique du compactage le plus récent de la base, effectué avec la commande Compact fichier données ou le Centre de sécurité et de maintenance (CSM). Stocké dans le dossier Logs.
Current backup settings file	18	fichier backup.4DSettings utilisé actuellement par l'application. Il peut s'agir du fichier backup.4DSettings par défaut ou d'un fichier de settings de backup utilisateur défini pour le fichier de données
Debug log file	12	Fichier d'enregistrement des événements pour le débogage créé par la commande <code>SET DATABASE PARAMETER(Debug log recording)</code> . Stocké dans le dossier Logs.
Diagnostic log file	11	Fichier de diagnostic de 4D, créé par la commande <code>SET DATABASE PARAMETER(Diagnostic log recording)</code> . Stocké dans le dossier Logs.
Directory file	16	fichier directory.json, contenant la description des groupes et utilisateurs (le cas échéant) du projet. Il se situe soit dans le dossier Settings de l'utilisateur (par défaut, s'applique à tout le projet), soit dans le dossier Settings du data (spécifique à un fichier de données).
HTTP debug log file	9	Fichier d'enregistrement des requêtes Web créé par la commande <code>WEB SET OPTION(Web log recording)</code> . Stocké dans le dossier Logs.
HTTP log file	8	Fichier de débogage des requêtes HTTP, créé par la commande <code>WEB SET OPTION(Web debug log)</code> . Stocké dans le dossier Logs.
IMAP Log file	23	Fichier d'historique créé par la commande <code>SET DATABASE PARAMETER(IMAP Log)</code> . Stocké dans le dossier Logs.
Last backup file	2	Dernier fichier de sauvegarde généré, nommé <applicationName>[bkpNum].4BK, stocké à un emplacement personnalisé.
Last journal integration log file	22	Chemin complet du dernier fichier journal d'intégration de l'historique (stocké dans le dossier Logs de l'application restaurée), le cas échéant. Ce fichier est créé en mode auto-repair, dès qu'une intégration de fichier d'historique a lieu
Repair log file	7	Fichier d'historique des réparations effectuées sur la base par le Centre de sécurité et de maintenance (CSM). Stocké dans le dossier Logs.
Request log file	10	Fichier des requêtes client/server standard (hors requêtes Web), créé par <code>SET DATABASE PARAMETER(4D Server log recording)</code> ou <code>SET DATABASE PARAMETER(Client log recording)</code> . Si la commande est appelée sur le serveur, le chemin du fichier des requêtes du serveur est retourné (stocké dans le dossier Logs du serveur). Si la commande est appelée sur un client, le chemin du fichier des requêtes du client est retourné (stocké dans le dossier Logs local du client).
SMTP log file	15	Fichier des requêtes SMTP créé par la commande <code>SET DATABASE PARAMETER(SMTP Log)</code> . Stocké dans le dossier Logs.
User settings file	3	Fichier settings.4DSettings pour tous les fichiers de données (si activé), stocké dans le dossier Preferences à côté du fichier de structure.
User settings file for data	4	Fichier settings.4DSettings file pour le fichier de données courant, stocké dans le dossier Preferences à côté du fichier de données.
Verification log file	5	Fichier d'historique de vérification le plus récent de la base, créé par les commandes <code>VERIFY CURRENT DATA FILE</code> et <code>VERIFY DATA FILE</code> ou via le Centre de sécurité et de maintenance de la base (CSM). Stocké dans le dossier Logs.

Si le fichier `fileConstant` cible n'existe pas, un objet null est retourné. Aucune erreur n'est générée.

Si la commande est appelée à partir d'un composant, passez le paramètre optionnel * pour lire le chemin de la base hôte. Sinon, si vous omettez le paramètre *, un objet null est systématiquement retourné.

4D.File.new()

► Historique

4D.File.new (*path* : Text { ; *pathType* : Integer }{ ; * }) : 4D.File

4D.File.new (*fileConstant* : Integer { ; * }) : 4D.File

Description

La fonction `4D.File.new()` crée et retourne un nouvel objet de type `4D.File`. Elle est identique à la commande `File` (raccourci).

Il est recommandé d'utiliser la commande raccourci `File` au lieu de `4D.File.new()`.

.copyTo()

► Historique

`.copyTo(destinationFolder : 4D.Folder { ; newName : Text } { ; overwrite : Integer }) : 4D.File`

Paramètres	Type		Description
<i>destinationFolder</i>	4D.Folder	->	Dossier de destination
<i>newName</i>	Text	->	Nom de la copie
<i>overwrite</i>	Integer	->	<code>fk overwrite</code> pour écraser les éléments existants
Résultat	4D.File	<-	Fichier copié

Description

La fonction `.copyTo()` copie l'objet `File` (fichier source) vers le *destinationFolder* spécifié.

Le *destinationFolder* doit exister sur disque, sinon une erreur est générée.

Par défaut, le fichier est copié avec le nom du fichier original. Si vous souhaitez renommer la copie, passez le nouveau nom dans le paramètre *newName*. Le nouveau nom doit être conforme aux règles de nommage (ex : il ne doit pas contenir de caractères tels que ":", "/", etc.), sinon une erreur est retournée.

S'il existe déjà un fichier portant le même nom dans *destinationFolder*, par défaut 4D génère une erreur. Vous pouvez passer la constante `fk overwrite` dans le paramètre *overwrite* pour ignorer et écraser le dossier existant :

Constante	Valeur	Commentaire
<code>fk overwrite</code>	4	Écrase les éléments existants, le cas échéant

Valeur renournée

L'objet `File` copié.

Exemple

Vous souhaitez copier un *file* image, à partir du dossier Documents de l'utilisateur vers le dossier de la base :

```
var $source; $copy : Object
$source:=Folder(fk documents folder).file("Pictures/photo.png")
$copy:=$source.copyTo(Folder("/PACKAGE");fk overwrite)
```

.create()

► Historique

Non disponible pour les archives ZIP

.create() : Boolean

Paramètres	Type		Description
Résultat	Booléen	<-	Vrai si le fichier a été créé avec succès, sinon Faux

Description

La fonction `.create()` crée un fichier sur disque en fonction des propriétés de l'objet `File`.

Le cas échéant, la fonction crée la hiérarchie du dossier en se basant sur la description des propriétés `platformPath` ou `path`. Si le fichier existe déjà sur disque, la fonction ne fait rien (aucune erreur n'est générée) et retourne faux.

Valeur renournée

- Vrai si le fichier est créé avec succès ;
- Faux si un fichier du même nom existe déjà ou si une erreur s'est produite.

Exemple

Création d'un fichier de préférences dans le dossier principal :

```
var $created : Boolean  
$created:=File("/PACKAGE/SpecialPrefs/"+Current user+".myPrefs").create()
```

.createAlias()

► Historique

`.createAlias(destinationFolder : 4D.Folder ; aliasName : Text { ; aliasType : Integer }) : 4D.File`

Paramètres	Type		Description
destinationFolder	4D.Folder	->	Dossier de destination pour l'alias ou le raccourci
aliasName	Text	->	Nom de l'alias ou du raccourci
aliasType	Integer	->	Type de lien de l'alias
Résultat	4D.File	<-	Référence du fichier de l'alias ou du raccourci

Description

La fonction `.createAlias()` crée un alias (macOS) ou un raccourci (Windows) pour le fichier nommé `aliasName` dans le dossier désigné par l'objet `destinationFolder`.

Passez le nom de l'alias ou du raccourci à créer dans le paramètre `aliasName`.

Par défaut sur macOS, la fonction crée un alias standard. Vous pouvez également créer un lien symbolique à l'aide du paramètre `aliasType`. Les constantes suivantes sont disponibles :

Constante	Valeur	Commentaire
<code>fk alias link</code>	0	Lien alias (macOS uniquement)(par défaut)
<code>fk symbolic link</code>	1	Lien symbolique (macOS uniquement)

Sur Windows, un raccourci (fichier .lnk) est toujours créé (le paramètre `aliasType` est ignoré).

Objet renvoyé

Un objet `4D.File` avec la propriété `isAlias` mise à true.

Exemple

Vous souhaitez créer un alias pour un fichier contenu dans votre dossier principal :

```
$myFile:=Folder(fk documents folder).file("Archives/ReadMe.txt")
$aliasFile:=$myFile.createAlias(File("/PACKAGE");"ReadMe")
```

.creationDate

► Historique

.creationDate : Date

Description

La propriété `.creationDate` retourne la date de création du fichier.

Cette propriété est en lecture seule.

.creationTime

► Historique

.creationTime : Time

Description

La propriété `.creationTime` retourne l'heure de création du fichier (exprimée en nombre de secondes à partir de 00:00).

Cette propriété est en lecture seule.

.delete()

► Historique

.delete()

Paramètres Type Description ----- ---- ----- Ne requiert aucun paramètre
--

Description

La fonction `.delete()` supprime le fichier.

Si le fichier est ouvert, une erreur est générée.

Si le fichier existe déjà sur disque, la fonction ne fait rien (aucune erreur n'est générée).

ATTENTION : `.delete()` peut supprimer n'importe quel fichier sur un disque, y compris les documents créés avec d'autres applications ainsi que les applications elles-mêmes. `.delete()` doit être utilisé avec prudence. La suppression d'un fichier est une opération permanente et irréversible.

Exemple

Vous souhaitez supprimer un fichier spécifique dans un sous-dossier :

```
$tempo:=File("/PACKAGE/SpecialPrefs/"+Current user+".prefs")
If($tempo.exists)
    $tempo.delete()
    ALERT("User preference file deleted.")
End if
End if
```

.exists

► Historique

.exists : Boolean

Description

La propriété `.exists` retourne vrai si le fichier existe sur le disque, sinon elle retourne faux.

Cette propriété est en lecture seule.

.extension

► Historique

.extension : Text

Description

La propriété `.extension` retourne l'extension du nom de fichier (le cas échéant). Une extension commence toujours par `".`. La propriété retourne une chaîne vide si le nom du fichier ne contient pas d'extension.

Cette propriété est en lecture seule.

.fullName

► Historique

.fullName : Text

Description

La propriété `.fullName` retourne le nom complet du fichier, extension comprise (le cas échéant).

Cette propriété est en lecture seule.

.getAppInfo()

► Historique

.getAppInfo() : Object

Paramètres	Type		Description
Résultat	Object	<-	Contenu du fichier de ressource version .exe/.dll ou .plist

Description

La fonction `.getAppInfo()` retourne le contenu d'un fichier d'information .exe, .dll ou .plist sous forme d'objet.

Cette fonction doit être utilisée avec un fichier .exe, .dll ou .plist existant. Si le fichier n'existe pas sur disque ou n'est pas un fichier .exe, .dll ou .plist valide, la fonction retourne un objet vide (aucune erreur n'est générée).

Cette fonction ne prend en charge que les fichiers .plist au format xml (texte). Une erreur est renvoyée si elle est utilisée avec un fichier .plist au format binaire.

Objet retourné dans le cas d'un fichier .exe ou .dll

La lecture d'un fichier .exe ou .dll est possible uniquement sous Windows.

Toutes les valeurs de propriétés sont de type Texte.

Propriété	Type
InternalName	Text
ProductName	Text
CompanyName	Text
LegalCopyright	Text
ProductVersion	Text
FileDescription	Text
FileVersion	Text
OriginalFilename	Text

Objet retourné dans le cas d'un fichier .plist

Le contenu xml du fichier est analysé et les clés sont retournées en tant que propriétés de l'objet, en préservant leur type (texte, booléen, numérique). `.plist dict` est retourné sous forme d'objet JSON et `.plist array` est retourné sous forme de tableau JSON.

Exemple

```
// display copyright info of application .exe file (windows)
var $exeFile : 4D.File
var $info : Object
$exeFile:=File(Application file; fk platform path)
$info:=$exeFile.getAppInfo()
ALERT($info.LegalCopyright)

// display copyright info of an info.plist (any platform)
var $infoPlistFile : 4D.File
var $info : Object
$infoPlistFile:=File("/RESOURCES/info.plist")
$info:=$infoPlistFile.getAppInfo()
ALERT($info.Copyright)
```

Voir aussi

[.setAppInfo\(\)](#)

.getContent()

► Historique

`.getContent()` : 4D.Blob

Paramètres	Type		Description
Résultat	4D.Blob	<-	Contenu du fichier

Description

La fonction `.getContent()` retourne un objet `4D.Blob` contenant l'intégralité du contenu d'un fichier. Pour plus d'informations sur les BLOB, veuillez vous reporter à la section [BLOB](#).

Valeur retournée

Un objet `4D.Blob`.

Exemple

Pour sauvegarder le contenu d'un document dans un champ **BL0B** :

```
var $vPath : Text
$vPath:=Select document("");*"Select a document";0)
If(OK=1) //Si un document a été sélectionné
    [aTable]aBlobField:=File($vPath;fk platform path).getContent()
End if
```

.getIcon()

► Historique

.getIcon({ size : Integer }) : Picture

Paramètres	Type		Description
size	Integer	->	Longueur du côté de l'image retournée (pixels)
Résultat	Image	<-	Ikône

Description

La fonction `.getIcon()` retourne l'icône du fichier.

Le paramètre optionnel `size` spécifie les dimensions en pixels de l'icône retournée. Cette valeur représente la longueur latérale du côté du carré contenant l'icône. La taille des icônes est généralement de 32x32 pixels ("grandes icônes") ou de 16x16 pixels ("petites icônes"). Si vous passez 0 ou si vous omettez ce paramètre, la version "grandes icônes" est retournée.

Si le fichier n'existe pas sur disque, une icône par défaut vide est retournée.

Valeur retournée

Image de l'icône du fichier.

.getText()

► Historique

.getText({ charSetName : Text { ; breakMode : Integer } }) : Text
.getText({ charSetNum : Integer { ; breakMode : Integer } }) : Text

Paramètres	Type		Description
charSetName	Text	->	Nom du jeu de caractères
charSetNum	Integer	->	Numéro du jeu de caractères
breakMode	Integer	->	Mode de traitement des retours à la ligne
Résultat	Text	<-	Texte du document

Description

La fonction `.getText()` retourne le contenu du fichier en tant que texte .

Optionnellement, vous pouvez indiquer le jeu de caractères à utiliser pour la lecture du contenu. Vous pouvez passer soit :

- dans `charSetName`, une chaîne contenant le nom de jeu standard (par exemple "ISO-8859-1" ou "UTF-8"),
- ou dans `charSetNum`, l'ID MIBEnum (numéro) du nom du jeu standard.

Pour consulter la liste des jeux de caractères pris en charge par 4D, veuillez vous reporter à la description de la commande `CONVERT FROM TEXT`.

Si le document contient un BOM (Byte Order Mark), 4D utilise le jeu de caractères inséré au lieu de celui qui est indiqué dans `charSetName` or `charSetNum` (ce paramètre est alors ignoré). Si le document ne contient pas de BOM et si le paramètre `charSetName` ou `charSetNum` est omis, 4D utilise par défaut le jeu de caractères "UTF-8".

Dans le paramètre `breakMode`, vous pouvez passer une valeur numérique indiquant le traitement à appliquer aux caractères de fin de ligne du document. Les constantes suivantes du thème "Documents système" sont disponibles :

Constante	Valeur	Commentaire
Document unchanged	0	Aucun traitement
Document with native format	1	(Défaut) Les fins de ligne sont convertis au format natif de la plate-forme d'exécution : CR (carriage return) sous OS X, CRLF (carriage return + line feed) sous Windows
Document with CRLF	2	Les fins de ligne sont convertis au format Windows : CRLF (carriage return + line feed)
Document with CR	3	Les fins de ligne sont convertis au format OS X : CR (carriage return)
Document with LF	4	Les fins de ligne sont convertis au format Unix : LF (line feed)

Par défaut, lorsque vous omettez le paramètre `breakMode` les retours à la ligne sont traités en mode natif (1).

Valeur renournée

Texte du fichier.

Exemple

Considérons le document texte suivant (les champs sont séparés par des tabulations) :

```
id name price vat
3 thé 1.06€ 19.6
2 café 1.05€ 19.6
```

Lorsque vous exécutez ce code :

```
$myFile:=Folder(fk documents folder).file("Billing.txt") //UTF-8 par défaut
$txt:=$myFile.getText()
```

... vous obtenez pour `$txt` :

"id\tname\tprice\tvat\r\n3\tthé\t1.06€\t19.6\r\n2\tcafé\t1.05€\t19.6"

avec `\t` (tab) comme séparateur et `\r\n` (CRLF) comme délimiteur de ligne.

Voici un autre exemple avec le même fichier, mais un délimiteur de ligne différent :

```
$txt:=$myFile.getText("UTF-8", Document with LF)
```

Dans ce cas, le contenu de `$txt` est :

"id\tname\tprice\tvat\n3\tthé\t1.06€\t19.6\n2\tcafé\t1.05€\t19.6"

Cette fois-ci `\n` (LF) est utilisé comme délimiteur de ligne.

.hidden

► Historique

.hidden : Boolean

Description

La propriété `.hidden` retourne vrai si le fichier est identifié comme "caché" au niveau du système, sinon elle retourne faux.

Cette propriété est en lecture seule.

.isAlias

► Historique

.isAlias : Boolean

Description

La propriété `.isAlias` retourne vrai si le fichier est un alias, un raccourci ou un lien symbolique, sinon elle retourne faux.

Cette propriété est en lecture seule.

.isFile

► Historique

.isFile : Boolean

Description

La propriété `.isFile` retourne toujours vrai pour un fichier.

Cette propriété est en lecture seule.

.isFolder

► Historique

.isFolder : Boolean

Description

La propriété `.isFolder` retourne toujours faux pour un fichier.

Cette propriété est en lecture seule.

.isWritable

► Historique

.isWritable : Boolean

Description

La propriété `.isWritable` retourne vrai si le fichier existe sur disque et s'il est modifiable.

Cette propriété vérifie la capacité de l'application 4D à écrire sur le disque (droits d'accès). elle ne se base pas uniquement sur l'attribut *writable* du fichier.

Cette propriété est en lecture seule.

Exemple

```

$myFile:=File("C:\\Documents\\Archives\\ReadMe.txt";fk platform path)
If($myFile.isWritable)
    $myNewFile:=$myFile.setText("Added text")
End if

```

.modificationDate

► Historique

.modificationDate : Date

Description

La propriété `.modificationDate` retourne la date de la dernière modification apportée au fichier.

Cette propriété est en lecture seule.

.modificationTime

► Historique

.modificationTime : Time

Description

La propriété `.modificationTime` retourne l'heure de la dernière modification du fichier (exprimée en nombre de secondes depuis 00:00).

Cette propriété est en lecture seule.

.moveTo()

► Historique

.moveTo(*destinationFolder* : 4D.Folder { ; *newName* : Text }) : 4D.File

Paramètres	Type		Description
<i>destinationFolder</i>	4D.Folder	->	Dossier de destination
<i>newName</i>	Text	->	Nom complet du fichier déplacé
Résultat	4D.File	<-	Fichier déplacé

Description

La fonction `.moveTo()` déplace ou renomme l'objet `File` vers le dossier *destinationFolder*.

Le *destinationFolder* doit exister sur disque, sinon une erreur est générée.

Par défaut, le fichier garde le même nom lorsqu'il est déplacé. Si vous souhaitez renommer le fichier déplacé, passez le nom complet dans le paramètre *newName*. Le nouveau nom doit être conforme aux règles de nommage (ex : il ne doit pas contenir de caractères tels que ":" , "/" , etc.), sinon une erreur est retournée.

Objet retourné

L'objet `File` déplacé.

Exemple

```

$DocFolder:=Folder(fk documents folder)
$myFile:=$DocFolder.file("Current/Infos.txt")
$myFile.moveTo($DocFolder.folder("Archives");"Infos_old.txt")

```

.name

► Historique

.name : Text

Description

La propriété `.name` retourne le nom du fichier, sans l'extension (le cas échéant).

Cette propriété est en lecture seule.

.original

► Historique

.original : 4D.File

.original : 4D.Folder

Description

La propriété `.original` retourne l'élément cible d'un alias, d'un raccourci ou d'un lien symbolique. L'élément cible peut être :

- un objet File
- un objet Folder

Pour les fichiers sans alias, la propriété retourne le même objet File que le fichier.

Cette propriété est en lecture seule.

.parent

► Historique

.parent : 4D.Folder

Description

La propriété `.parent` retourne l'objet dossier parent du fichier. Si le chemin représente un filesystem (ex : "/DATA/"), le filesystem est retourné.

Cette propriété est en lecture seule.

.path

► Historique

.path : Text

Description

La propriété `.path` retourne le chemin POSIX du fichier. Si le chemin représente un filesystem (ex : "/DATA/"), le filesystem est retourné.

Cette propriété est en lecture seule.

.platformPath

► Historique

.platformPath : Text

Description

La propriété `.platformPath` retourne le chemin du fichier exprimé dans la syntaxe de la plate-forme.

Cette propriété est en lecture seule.

.rename()

► Historique

.rename(*newName* : Text) : 4D.File

Paramètres	Type		Description
<i>newName</i>	Text	->	Nouveau nom complet du fichier
Résultat	4D.File	<-	Fichier renommé

Description

La fonction `.rename()` renomme le fichier avec le nom que vous avez passé dans le paramètre *newName* et retourne l'objet `File` renommé.

Le paramètre *newName* doit être conforme aux règles de nommage (ex : il ne doit pas contenir des caractères tels que ":" , "/" , etc.), sinon une erreur est retournée. S'il existe déjà un fichier portant le même nom, une erreur est retournée.

A noter que la fonction modifie le nom complet du fichier, c'est-à-dire que si vous ne passez pas une extension dans le paramètre *newName*, le fichier aura un nom sans extension.

Objet retourné

L'objet `File` renommé.

Exemple

Vous souhaitez que "ReadMe.txt" soit renommé "ReadMe_new.txt" :

```
$toRename:=File("C:\\Documents\\\\Archives\\\\ReadMe.txt";fk platform path)
$newName:=$toRename.rename($toRename.name+"_new"+$toRename.extension)
```

.setAppInfo()

► Historique

.setAppInfo(*info* : Object)

Paramètres	Type		Description
<i>info</i>	Object	->	Propriétés à écrire dans le fichier .plist ou la ressource version du fichier .exe/.dll

Description

La fonction `.setAppInfo()` écrit les propriétés *info* sous forme de contenu d'information d'un fichier .exe, .dll ou .plist.

Cette fonction doit être utilisée avec un fichier .exe, .dll ou .plist existant. Cette fonction doit être utilisée avec un fichier .exe, .dll ou .plist existant.

Cette fonction ne prend en charge que les fichiers .plist au format xml (texte). Une erreur est retournée si elle est utilisée avec un fichier .plist au format binaire.

Paramètre *info* avec un fichier .exe or .dll

Ecrire les informations de fichiers .exe ou .dll est possible uniquement sous Windows.

Chaque propriété valide définie dans le paramètre objet *info* est écrite dans la ressource de version du fichier .exe ou .dll. Les propriétés disponibles sont (toute autre propriété sera ignorée) :

Propriété	Type
InternalName	Text
ProductName	Text
CompanyName	Text
LegalCopyright	Text
ProductVersion	Text
FileDescription	Text
FileVersion	Text
OriginalFilename	Text

Si vous passez null ou un texte vide comme valeur, une chaîne vide est écrite dans la propriété. Si vous passez une valeur de type autre que Texte, elle est "stringifiée".

Paramètre *info* avec un fichier .plist

Chaque propriété valide définie dans le paramètre objet *info* est écrite dans le fichier . plist sous forme de clé. Tous les noms de clés sont acceptés. Les types des valeurs sont préservés si possible.

Si une clé définie dans le paramètre *info* est déjà définie dans le fichier .plist, sa valeur est mise à jour tout en conservant son type d'origine. Les autres clés définies dans le fichier .plist ne sont pas modifiées.

Pour définir une valeur de type Date, le format à utiliser est chaîne de timestamp json formatée en ISO UTC sans les millisecondes ("2003-02-01T01:02:03Z") comme dans l'éditeur de plist Xcode.

Exemple

```
// définir copyright et version d'un fichier.exe (Windows)
var $exeFile : 4D.File
var $info : Object
$exeFile:=File(Application file; fk platform path)
$info:=New object
$info.LegalCopyright:="Copyright 4D 2021"
$info.ProductVersion:="1.0.0"
$exeFile.setAppInfo($info)
```

```
// définir des clés dans un fichier info.plist (toutes plates-formes)
var $infoPlistFile : 4D.File
var $info : Object
$infoPlistFile:=File("/RESOURCES/info.plist")
$info:=New object
$info.Copyright:="Copyright 4D 2021" //text
$info.ProductVersion:=12 //integer
$info.ShipmentDate:="2021-04-22T06:00:00Z" //timestamp
$infoPlistFile.setAppInfo($info)
```

Voir aussi

[.getAppInfo\(\)](#)

.setContent()

► Historique

[.setContent \(content : Blob \)](#)

Paramètres	Type		Description
content	BLOB	->	Nouveau contenu du fichier

Description

La fonction `.setContent()` réécrit le contenu intégral du fichier à l'aide des données stockées dans le BLOB `content`. Pour plus d'informations sur les BLOB, veuillez vous reporter à la section [BLOB](#).

Exemple

```
$myFile:=Folder(fk documents folder).file("Archives/data.txt")
$myFile.setContent([aTable]aBlobField)
```

.setText()

► Historique

```
.setText( text : Text {; charSetName : Text { ; breakMode : Integer } } )
.setText( text : Text {; charSetNum : Integer { ; breakMode : Integer } } )
```

Paramètres	Type		Description
Texte	Text	->	Texte à stocker dans le fichier
charSetName	Text	->	Nom du jeu de caractères
charSetNum	Integer	->	Numéro du jeu de caractères
breakMode	Integer	->	Mode de traitement des retours à la ligne

Description

|

Si le fichier référencé dans l'objet `File` n'existe pas sur disque, il est créé par la fonction. Lorsque le fichier existe déjà sur disque, son contenu antérieur est supprimé, sauf s'il est déjà ouvert, auquel cas son contenu est verrouillé et une erreur est générée.

Dans le paramètre `text`, passez le texte à écrire dans le fichier. Cela peut être un texte littéral ("my text"), ou un champ / variable texte 4D.

Optionnellement, vous pouvez indiquer le jeu de caractères à utiliser pour l'écriture du contenu. Vous pouvez passer soit :

- dans `charSetName`, une chaîne contenant le nom de jeu standard (par exemple "ISO-8859-1" ou "UTF-8"),
- ou dans `charSetNum`, l'ID MIBEnum (numéro) du nom du jeu standard.

Pour consulter la liste des jeux de caractères pris en charge par 4D, veuillez vous reporter à la description de la commande `CONVERT FROM TEXT`.

Si une marque d'ordre d'octet (BOM) existe pour le jeu de caractères, 4D l'insère dans le fichier, sauf si le jeu de caractères utilisé contient le suffixe "-no-bom" (par exemple "UTF-8-no-bom"). Si vous n'indiquez pas un jeu de caractères, 4D utilise par défaut le jeu de caractères "UTF-8" sans BOM.

Dans le paramètre `breakMode`, vous pouvez passer une valeur numérique indiquant le traitement à appliquer aux caractères de fin de ligne avant de les stocker dans le fichier. Les constantes suivantes du thème Documents système sont disponibles :

Constante	Valeur	Commentaire
Document unchanged	0	Aucun traitement
Document with native format	1	(Défaut) Les fins de ligne sont convertis au format natif de la plate-forme d'exécution : LF (line feed) sous macOS, CRLF (carriage return + line feed) sous Windows
Document with CRLF	2	Les fins de ligne sont converties en CRLF (retour chariot + saut de ligne), le format par défaut de Windows
Document with CR	3	Les fins de ligne sont converties en CR (retour chariot), le format MacOS classique par défaut
Document with LF	4	Les fins de ligne sont converties en LF (line feed), le format Unix et macOS par défaut

Par défaut, lorsque vous omettez le paramètre *breakMode* les retours à la ligne sont traités en mode natif (1).

Note de compatibilité : Des options de compatibilité sont disponibles pour la gestion des fins de ligne et des BOM. Voir la [page Compatibilité](#) sur doc.4d.com.

Exemple

```
$myFile:=File("C:\\Documents\\Hello.txt";fk platform path)
$myFile.setText("Hello world")
```

.size

► Historique

.size : Real

Description

La propriété `.size` retourne la taille du fichier exprimée en octets. Si le fichier n'existe pas sur le disque, la taille est de 0.

Cette propriété est en lecture seule.

Folder

Les objets `Folder` sont créés avec la commande `Folder`. Ils contiennent des références à des dossiers qui peuvent exister réellement ou non sur le disque. Par exemple, lorsque vous exécutez la commande `Folder` pour créer un nouveau dossier, un objet `Folder` valide est créé mais rien n'est réellement stocké sur le disque jusqu'à ce que vous appeliez la fonction `folder.create()`.

Exemple

L'exemple suivant crée un dossier "JohnSmith" :

```
Form.curfolder:=Folder(fk database folder)
Form.curfolder:=Folder("C:\\\\Users\\\\JohnSmith\\\\";fk platform path)
```

Objet Folder

`.copyTo(destinationFolder : 4D.Folder { ; newName : Text } { ; overwrite : Integer }) : 4D Folder`
copie l'objet `Folder` (dossier source) vers le `destinationFolder` spécifié

`.create() : Boolean`
crée un dossier sur disque en fonction des propriétés de l'objet `Folder`

`.createAlias(destinationFolder : 4D.Folder ; aliasName : Text { ; aliasType : Integer }) : 4D.File`
crée un alias (macOS) ou un raccourci (Windows) du dossier

`.creationDate : Date`
la date de création du dossier.

`.creationTime : Time`
l'heure de création du dossier

`.delete({ option : Integer })`
supprime le dossier

`.exists : Boolean`
true si le dossier existe sur le disque

`.extension : Text`
retourne l'extension du nom du dossier (le cas échéant)

`.file(path : Text) : 4D.File`
crée crée un objet `File` dans l'objet `Folder` et retourne sa référence

`.files({ options : Integer }) : Collection`
une collection d'objets `File` contenus dans le dossier

`.folder(path : Text) : 4D.Folder`
crée un nouvel objet `Folder` dans l'objet `Folder` parent et retourne sa référence

`.folders({ options : Integer }) : Collection`
retourne une collection d'objets `Folder` contenus dans le dossier parent

<code>.fullName : Text</code>	retourne le nom complet du dossier, y compris son extension (le cas échéant)
<code>.getIcon({ size : Integer }) : Picture</code>	retourne l'icône du dossier
<code>.hidden : Boolean</code>	true si le dossier est défini comme "hidden" au niveau du système
<code>.isAlias : Boolean</code>	toujours false pour un objet <code>Folder</code>
<code>.isFile : Boolean</code>	
<code>.isFolder : Boolean</code>	toujours false pour un dossier
<code>.isPackage : Boolean</code>	true si le dossier est un package sous macOS (et s'il existe sur le disque)
<code>.modificationDate : Date</code>	la date de la dernière modification du dossier
<code>.modificationTime : Time</code>	l'heure de la dernière modification du dossier
<code>.name : Text</code>	le nom du dossier, sans extension (le cas échéant)
<code>.original : 4D.Folder</code>	le même objet dossier que le dossier original
<code>.parent : 4D.Folder</code>	l'objet dossier parent du dossier
<code>.path : Text</code>	le chemin POSIX du dossier
<code>.platformPath : Text</code>	le chemin du dossier exprimé dans la syntaxe de la plate-forme courante
<code>.moveTo(destinationFolder : 4D.Folder { ; newName : Text }) : 4D.Folder</code>	déplace ou déplace et renomme l'objet <code>Folder</code> source dans le dossier <code>destinationFolder</code>
<code>.rename(newName : Text) : 4D.Folder</code>	renomme le dossier avec le nom que vous avez passé dans le paramètre <code>newName</code> et retourne l'objet <code>Folder</code> renommé

Folder

► Historique

`Folder (path : Text { ; pathType : Integer }{ ; * }) : 4D.Folder`

`Folder (folderConstant : Integer { ; * }) : 4D.Folder`

Paramètres	Type		Description
path	Text	->	Chemin du dossier
folderConstant	Integer	->	Constante de dossier 4D
pathType	Integer	->	<code>fk posix path</code> (par défaut) ou <code>fk platform path</code>
*		->	* pour retourner le dossier de la base hôte
Résultat	4D.Folder	<-	Nouvel objet dossier

Description

La commande `Folder` crée et retourne un nouvel objet de type `4D.Folder`. La commande accepte deux syntaxes :

`Folder (path { ; pathType } { ; * })`

Dans le paramètre *path*, passez un chemin de dossier. Vous pouvez utiliser une chaîne personnalisée ou un "filesystem" (ex : "/DATA").

Seuls les noms de chemin absolu sont pris en charge par la commande `Folder`.

Par défaut, 4D attend un chemin exprimé avec la syntaxe POSIX. Si vous travaillez avec des chemins de plate-forme (Windows ou macOS), vous devez les déclarer à l'aide du paramètre *pathType*. Les constantes suivantes sont disponibles :

Constante	Valeur	Commentaire
<code>fk platform path</code>	1	Chemin exprimé dans une syntaxe spécifique à la plate-forme (obligatoire en cas de chemin de plate-forme)
<code>fk posix path</code>	0	Chemin exprimé avec la syntaxe POSIX (par défaut)

`Folder (folderConstant { ; * })`

Dans le paramètre *folderConstant*, passez un dossier 4D interne ou un dossier système, à l'aide d'une des constantes suivantes :

Constante	Valeur	Commentaire
fk applications folder	116	
fk data folder	9	Filesystem associé : "/DATA"
fk database folder	4	Filesystem associé : "/PACKAGE"
fk desktop folder	115	
fk documents folder	117	Dossier Documents de l'utilisateur
fk licenses folder	1	Dossier contenant les fichiers de licence 4D de la machine
fk logs folder	7	Filesystem associé : "/LOGS"
fk mobileApps folder	10	
fk remote database folder	3	Dossier de la base de données 4D créé sur chaque machine 4D distante
fk resources folder	6	Filesystem associé : "/RESOURCES"
fk system folder	100	
fk user preferences folder	0	Dossier 4D qui stocke les préférences utilisateur dans le <username> répertoire de l'utilisateur.
fk web root folder	8	Dossier racine web courant du projet : "/PACKAGE/chemin" si son emplacement se trouve dans le package, sinon chemin complet

Si la commande est appelée à partir d'un composant, passez le paramètre optionnel * pour lire le chemin de la base hôte. Sinon, si vous omettez le paramètre *, un objet null est systématiquement retourné.

4D.Folder.new()

► Historique

```
4D.Folder.new ( path : Text { ; pathType : Integer }{ ; * } ) : 4D.Folder
4D.Folder.new ( folderConstant : Integer { ; * } ) : 4D.Folder
```

Description

La fonction `4D.Folder.new()` crée et retourne un nouvel objet de type `4D.Folder`. Elle est identique à la commande `Folder` (raccourci).

Il est recommandé d'utiliser la commande raccourci `Folder` au lieu de `4D.Folder.new()`.

.copyTo()

► Historique

```
.copyTo( destinationFolder : 4D.Folder { ; newName : Text } { ; overwrite : Integer } ) : 4D.Folder
```

Paramètres	Type		Description
destinationFolder	4D.Folder	->	Dossier de destination
newName	Text	->	Nom de la copie
overwrite	Integer	->	<code>fk overwrite</code> pour écraser les éléments existants
Résultat	4D.Folder	<-	Dossier copié

Description

La fonction `.copyTo()` copie l'objet `Folder` (dossier source) vers le *destinationFolder* spécifié.

Le *destinationFolder* doit exister sur disque, sinon une erreur est générée.

Par défaut, le dossier est copié avec le nom du dossier original. Si vous souhaitez renommer la copie, passez le nouveau nom dans le paramètre *newName*. Le nouveau nom doit être conforme aux règles de nommage (ex : il ne doit pas contenir de caractères tels que ":", "/", etc.), sinon une erreur est retournée.

S'il existe déjà un dossier portant le même nom dans *destinationFolder*, par défaut 4D génère une erreur. Vous pouvez passer la constante `fk overwrite` dans le paramètre *overwrite* pour ignorer et écraser le dossier existant :

Constante	Valeur	Commentaire
<code>fk overwrite</code>	4	Écrase les éléments existants, le cas échéant

Valeur retournée

L'objet `Folder` copié.

Exemple

Vous souhaitez copier un dossier `Images`, à partir du dossier `Document` de l'utilisateur vers le dossier de la base :

```
var $userImages; $copiedImages : 4D.Folder
$userImages:=Folder(fk documents folder+"/Pictures/")
$copiedImages:=$userImages.copyTo(Folder(fk database folder);fk overwrite)
```

.create()

► Historique

`.create() : Boolean`

Paramètres	Type		Description
Résultat	Booléen	<-	Vrai si le dossier a été créé avec succès, sinon Faux

Description

La fonction `.create()` crée un dossier sur disque en fonction des propriétés de l'objet `Folder`.

Le cas échéant, la fonction crée la hiérarchie du dossier en se basant sur la description des propriétés `platformPath` ou `path`. Si le dossier existe déjà sur disque, la fonction ne fait rien (aucune erreur n'est générée) et retourne faux.

Valeur retournée

- Vrai si le dossier est créé avec succès ;
- Faux si un dossier du même nom existe déjà ou si une erreur s'est produite.

Exemple 1

Créer un dossier vide dans le dossier principal :

```

var $created : Boolean
$created:=Folder("/PACKAGE/SpecialPrefs").create()

```

Exemple 2

Création d'un dossier "/Archives2019/January/" dans le dossier principal :

```

$newFolder:=Folder("/PACKAGE/Archives2019/January")
If($newFolder.create())
    ALERT("The "+$newFolder.name+" folder was created.")
Else
    ALERT("Impossible to create a "+$newFolder.name+" folder.")
End if
Else
    ALERT("Impossible to create a "+$newFolder.name+" folder.")
End if

```

.createAlias()

► Historique

.createAlias(*destinationFolder* : 4D.Folder ; *aliasName* : Text { ; *aliasType* : Integer }) : 4D.File

Paramètres	Type		Description
<i>destinationFolder</i>	4D.Folder	->	Dossier de destination pour l'alias ou le raccourci
<i>aliasName</i>	Text	->	Nom de l'alias ou du raccourci
<i>aliasType</i>	Integer	->	Type de lien de l'alias
Résultat	4D.File	<-	Référence de l'alias ou du raccourci du dossier

Description

La fonction `.createAlias()` crée un alias (macOS) ou un raccourci (Windows) du dossier avec le nom *aliasName* dans le dossier désigné par l'objet *destinationFolder*.

Passez le nom de l'alias ou du raccourci à créer dans le paramètre *aliasName*.

Par défaut sur macOS, la fonction crée un alias standard. Vous pouvez également créer un lien symbolique à l'aide du paramètre *aliasType*. Les constantes suivantes sont disponibles :

Constante	Valeur	Commentaire
<code>fk alias link</code>	0	Lien alias (macOS uniquement)(par défaut)
<code>fk symbolic link</code>	1	Lien symbolique (macOS uniquement)

Sur Windows, un raccourci (fichier .lnk) est toujours créé (le paramètre *aliasType* est ignoré).

Objet retourné

Un objet `4D.File` avec la propriété `isAlias` mise à true.

Exemple

Vous souhaitez créer un alias pour un dossier d'archives dans votre dossier principal :

```

$myFolder:=Folder("C:\\\\Documents\\\\Archives\\\\2019\\\\January";fk platform path)
$aliasFile:=$myFolder.createAlias(Folder("/PACKAGE");"Jan2019")

```

.creationDate

► Historique

.creationDate : Date

Description

La propriété `.creationDate` retourne la date de création du dossier.

Cette propriété est en lecture seule.

.creationTime

► Historique

.creationTime : Time

Description

La propriété `.creationTime` retourne l'heure de création du dossier (exprimée en nombre de secondes, commençant à 00:00).

Cette propriété est en lecture seule.

.delete()

► Historique

.delete({ *option* : Integer })

Paramètres	Type		Description
option	Integer	->	Option de suppression du dossier

Description

La fonction `.delete()` supprime le dossier.

Par défaut, pour des raisons de sécurité, si vous omettez le paramètre option, `.delete()` permet uniquement de supprimer les dossiers vides. Si vous souhaitez que la commande supprime des dossiers qui ne sont pas vides, vous devez utiliser le paramètre option avec l'une des constantes suivantes :

Constante	Valeur	Commentaire
Delete only if empty	0	Supprime le dossier uniquement s'il est vide
Delete with contents	1	Supprime le dossier ainsi que son éventuel contenu

Lorsque la constante `Delete only if empty` est passée ou si vous omettez le paramètre option :

- Le dossier n'est supprimé que s'il est vide ; sinon, la commande ne fait rien et une erreur -47 est générée.
- Si le dossier n'existe pas, l'erreur -120 est générée.

Lorsque la commande `Delete with contents` est passée :

- Le dossier, ainsi que tout son contenu, est supprimé. Attention : Même si ce dossier et/ou son contenu sont verrouillés ou définis comme étant en lecture seule, si l'utilisateur dispose des droits d'accès appropriés, le dossier (et son contenu) est supprimé malgré tout.
- Si ce dossier, ou l'un des fichiers qu'il contient, ne peut être supprimé, la suppression est interrompue dès que le premier élément inaccessible est détecté, et une erreur(*) est retournée. Dans ce cas, le dossier ne peut être que partiellement supprimé. Lorsque la suppression est interrompue, vous pouvez utiliser la commande `GET LAST ERROR STACK` pour récupérer le nom et le chemin d'accès du dossier incriminé.
- Si le dossier n'existe pas, la commande ne fait rien et aucune erreur n'est retournée. (*) Windows : -54 (Tentative d'ouverture en écriture d'un fichier verrouillé) macOS : -45 (Le fichier est verrouillé ou le chemin d'accès n'est pas

correct)

.exists

► Historique

.exists : Boolean

Description

La propriété `.exists` retourne true si le dossier existe sur le disque, et retourne false dans le cas contraire.

Cette propriété est en lecture seule.

.extension

► Historique

.extension : Text

Description

La propriété `.extension` retourne l'extension du nom du dossier (le cas échéant). Une extension commence toujours par `".`. La propriété retourne une chaîne vide si le nom du dossier n'a pas d'extension.

Cette propriété est en lecture seule.

.file()

► Historique

.file(*path* : Text) : 4D.File

Paramètres	Type		Description
<i>path</i>	Text	->	Chemin POSIX relatif
Résultat	4D.File	<-	Objet <code>File</code> (null si chemin invalide)

Description

La fonction `.file()` crée crée un objet `File` dans l'objet `Folder` et retourne sa référence.

Dans le paramètre *path*, passez un chemin relatif POSIX pour désigner le fichier à retourner. Le chemin sera évalué à partir du dossier parent en tant que racine.

Valeur renournée

Un objet `File` ou null si *path* n'est pas valide.

Exemple

```
var $myPDF : 4D.File  
$myPDF:=Folder(fk documents folder).file("Pictures/info.pdf")
```

.files()

► Historique

.files({ *options* : Integer }) : Collection

Paramètres	Type		Description
options	Integer	->	Options de liste de fichiers
Résultat	Collection	<-	Collection d'objets dossier enfant

Description

La fonction `.files()` retourne une collection d'objets `File` contenus dans le dossier.

Les alias ou les liens symboliques ne sont pas résolus.

Par défaut, si vous omettez le paramètre `options`, seuls les fichiers à la racine du dossier sont retournés dans la collection, ainsi que les fichiers et dossiers invisibles. Vous pouvez modifier cela en passant, dans le paramètre `options` parameter, une ou plusieurs des constantes suivantes :

Constante	Valeur	Commentaire
<code>fk recursive</code>	1	La collection contient les fichiers du dossier spécifié ainsi que de ses sous-dossiers
<code>fk ignore invisible</code>	8	Les fichiers invisibles ne sont pas répertoriés

Valeur renournée

Collection d'objets `File`.

Exemple 1

Vous souhaitez savoir s'il y a des fichiers invisibles dans le dossier de la base :

```
var $all; $noInvisible : Collection
$all:=Folder(fk database folder).files()
$noInvisible:=Folder(fk database folder).files(fk ignore invisible)
If($all.length#$noInvisible.length)
    ALERT("Il y a des fichiers invisibles dans le dossier de la base.")
End if
```

Exemple 2

Vous souhaitez lire tous les fichiers qui ne sont pas invisibles dans le dossier Documents :

```
var $recursive : Collection
$recursive:=Folder(fk documents folder).files(fk recursive+fk ignore invisible)
```

.folder()

► Historique

`.folder(path : Text) : 4D.Folder`

Paramètres	Type		Description
path	Text	->	Chemin POSIX relatif
Résultat	4D.Folder	<-	Objet dossier (null si <code>path</code> invalide)

Description

La fonction `.folder()` crée un nouvel objet `Folder` dans l'objet `Folder` parent et retourne sa référence.

Dans le paramètre *path*, passez un chemin relatif POSIX pour désigner le dossier à retourner. Le chemin sera évalué à partir du dossier parent en tant que racine.

Valeur renournée

Un objet `Folder` object ou null si *path* est invalide.

Exemple

```
var $mypicts : 4D.Folder  
$mypicts:=Folder(fk documents folder).folder("Pictures")
```

.folders()

► Historique

`.folders({ options : Integer }) : Collection`

Paramètres	Type		Description
options	Integer	->	Options de liste des dossiers
Résultat	Collection	<-	Collection d'objets dossier enfant

Description

La fonction `.folders()` retourne une collection d'objets `Folder` contenus dans le dossier parent.

Par défaut, si vous omettez le paramètre *options* , seuls les dossiers à la racine du dossier sont retournés dans la collection. Vous pouvez modifier cela en passant, dans le paramètre *options* parameter, une ou plusieurs des constantes suivantes :

Constante	Valeur	Commentaire
<code>fk recursive</code>	1	La collection contient les dossiers du dossier spécifié ainsi que de ses sous-dossiers
<code>fk ignore invisible</code>	8	Les dossiers invisibles ne sont pas répertoriés

Valeur renournée

Collection d'objets `Folder`.

Exemple

Vous souhaitez obtenir la collection de tous les dossiers et sous-dossiers du dossier de la base :

```
var $allFolders : Collection  
$allFolders:=Folder("/PACKAGE").folders(fk recursive)
```

.fullName

► Historique

`.fullName : Text`

Description

La propriété `.fullName` retourne le nom complet du dossier, y compris son extension (le cas échéant).

Cette propriété est en lecture seule.

.getIcon()

► Historique

.getIcon({ size : Integer }) : Picture

Paramètres	Type		Description
size	Integer	->	Longueur du côté de l'image retournée (pixels)
Résultat	Image	<-	Icône

Description

La fonction `.getIcon()` retourne l'icône du dossier.

Le paramètre optionnel `size` spécifie les dimensions en pixels de l'icône retournée. Cette valeur représente la longueur latérale du côté du carré contenant l'icône. La taille des icônes est généralement de 32x32 pixels ("grandes icônes") ou de 16x16 pixels ("petites icônes"). Si vous passez 0 ou si vous omettez ce paramètre, la version "grandes icônes" est retournée.

Si le dossier n'existe pas sur disque, une icône vide est retournée par défaut.

Valeur retournée

[Image](#) de l'icône du dossier.

.hidden

► Historique

.hidden : Boolean

Description

La propriété `.hidden` retourne true si le dossier est défini comme "hidden" au niveau du système, et retourne false dans le cas contraire.

Cette propriété est en lecture seule.

.isAlias

► Historique

.isAlias : Boolean

Description

La propriété `.isAlias` retourne toujours false pour un objet `Folder`.

Cette propriété est en lecture seule.

.isFile

► Historique

.isFile : Boolean

Description

La propriété `.isFile` retourne toujours false pour un dossier.

Cette propriété est en lecture seule.

.isFolder

► Historique

.isFolder : Boolean

Description

La propriété `.isFolder` retourne toujours false pour un dossier.

Cette propriété est en lecture seule.

.isPackage

► Historique

.isPackage : Boolean

Description

La propriété `.isPackage` retourne true si le dossier est un package sous macOS (et s'il existe sur le disque). Sinon, elle retourne false.

Sous Windows, `.isPackage` retourne toujours false.

Cette propriété est en lecture seule.

.modificationDate

► Historique

.modificationDate : Date

Description

La propriété `.modificationDate` retourne la date de la dernière modification du dossier.

Cette propriété est en lecture seule.

.modificationTime

► Historique

.modificationTime : Time

Description

La propriété `.modificationTime` retourne l'heure de la dernière modification du dossier (exprimée en nombre de secondes à partir de 00:00).

Cette propriété est en lecture seule.

.moveTo()

► Historique

.moveTo(*destinationFolder* : 4D.Folder { ; *newName* : Text }) : 4D.Folder

Paramètres	Type		Description
<i>destinationFolder</i>	4D.Folder	->	Dossier de destination
<i>newName</i>	Text	->	Nom complet du dossier déplacé
Résultat	4D.Folder	<-	Dossier déplacé

Description

La fonction `.moveTo()` déplace ou déplace et renomme l'objet `Folder` source dans le dossier `destinationFolder`.

Le `destinationFolder` doit exister sur disque, sinon une erreur est générée.

Par défaut, le dossier garde le même nom lorsqu'il est déplacé. Si vous souhaitez renommer le dossier déplacé, passez le nouveau nom complet dans le paramètre `newName`. Le nouveau nom doit être conforme aux règles de nommage (ex : il ne doit pas contenir de caractères tels que ":", "/", etc.), sinon une erreur est retournée.

Objet retourné

L'objet `Folder` déplacé.

Exemple

Vous souhaitez déplacer et renommer un dossier :

```
var $tomeove; $moved : Object  
$docs:=Folder(fk documents folder)  
$tomeove:=$docs.folder("Pictures")  
$tomeove2:=$tomeove.moveTo($docs.folder("Archives");"Pic_Archives")
```

.name

► Historique

.name : Text

Description

La propriété `.name` retourne le nom du dossier, sans extension (le cas échéant).

Cette propriété est en lecture seule.

.original

► Historique

.original : 4D.Folder

Description

La propriété `.original` retourne le même objet dossier que le dossier original.

Cette propriété est en lecture seule.

Cette propriété est disponible sur les dossiers pour permettre au code générique de traiter les dossiers ou les fichiers.

.parent

► Historique

.parent : 4D.Folder

Description

La propriété `.parent` retourne l'objet dossier parent du dossier. Si le chemin représente un filesystem (ex : "/DATA/"), le filesystem est retourné.

Si le dossier n'a pas de parent (racine), la valeur nulle est retournée.

Cette propriété est en lecture seule.

.path

► Historique

.path : Text

Description

La propriété `.path` retourne le chemin POSIX du dossier. Si le chemin représente un filesystem (ex : `"/DATA/"`), le filesystem est retourné.

Cette propriété est en lecture seule.

.platformPath

► Historique

.platformPath : Text

Description

La propriété `.platformPath` retourne le chemin du dossier exprimé dans la syntaxe de la plate-forme courante.

Cette propriété est en lecture seule.

.rename()

► Historique

.rename(*newName* : Text) : 4D.Folder

Paramètres	Type		Description
<i>newName</i>	Text	->	Nouveau nom complet du dossier
Résultat	4D.Folder	<-	Dossier renommé

Description

La fonction `.rename()` renomme le dossier avec le nom que vous avez passé dans le paramètre `newName` et retourne l'objet `Folder` renommé.

Le paramètre `newName` doit être conforme aux règles de nommage (ex : il ne doit pas contenir des caractères tels que `:`, `/`, etc.), sinon une erreur est retournée. S'il existe déjà un fichier portant le même nom, une erreur est retournée.

Objet retourné

L'objet `Folder` renommé.

Exemple

```
var $toRename : 4D.Folder
$toRename:=Folder("/RESOURCES/Pictures").rename("Images")
```

Formula

Les commandes [Formula](#) et [Formula from string](#) vous permettent de créer des [objets 4D.Function](#) natifs pour exécuter toute expression ou code 4D exprimé sous forme de texte.

Objets Formula

Les objets Formula peuvent être encapsulés dans des propriétés d'objet :

```
var $f : 4D.Function  
$f:=New object  
$f.message:=Formula(ALERT("Hello world"))
```

Cette propriété est une "fonction objet", c'est-à-dire une fonction qui est liée à son objet parent. Pour exécuter une fonction stockée dans une propriété objet, utilisez l'opérateur () après le nom de la propriété, comme par exemple :

```
$f.message() //affiche "Hello world"
```

La syntaxe avec des crochets est également prise en charge :

```
$f["message"]() //affiche "Hello world"
```

A noter que, même si elle n'a pas de paramètres (voir ci-dessous), une fonction objet à exécuter doit être appelée avec des parenthèses (). En appelant uniquement une seule propriété, une nouvelle référence à la formule sera retournée (et ne sera pas exécutée) :

```
$o:=$f.message //retourne l'objet formule en $o
```

Vous pouvez également exécuter une fonction à l'aide des fonctions [apply\(\)](#) et [call\(\)](#) :

```
$f.message.apply() //affiche "Hello world"
```

Passer des paramètres

Vous pouvez passer des paramètres à vos formules en utilisant [la syntaxe séquentielle des paramètres](#) basée sur \$1, \$2...\$n. Par exemple, vous pouvez écrire :

```
var $f : Object  
$f:=New object  
$f.message:=Formula(ALERT("Hello "+$1))  
$f.message("John") //affiche "Hello John"
```

Ou en utilisant la fonction [.call\(\)](#) :

```
var $f : Object  
$f:=Formula($1+" "+$2)  
$text:=$f.call(Null;"Hello";"World") //retourne "Hello World"  
$text:=$f.call(Null;"Welcome to";String(Year of(Current date))) //retourne "Welcome to 2019" (par exemp
```

Paramètres d'une seule méthode

Pour plus de commodité, lorsque la formule est constituée d'une seule méthode projet, les paramètres peuvent être omis dans l'initialisation de l'objet formule. Ils peuvent simplement être passés lorsque la formule est appelée. Par exemple :

```
var $f : 4D.Function

$f:=Formula(myMethod)
//Ecrire Formula(myMethod($1;$2)) n'est pas nécessaire
$text:=$f.call(Null;"Hello";"World") //retourne "Hello World"
$text:=$f.call() //retourne "How are you?"

//myMethod
#DECLARE ($param1 : Text; $param2 : Text)->$return : Text
If(Count parameters=2
    $return:=$param1+" "+$param2
Else
    $return:="How are you?"
End if
```

Les paramètres sont reçus dans la méthode, dans l'ordre où ils sont spécifiés dans l'appel.

À propos des objets 4D.Function

Un objet `4D.Function` contient un morceau de code qui peut être exécuté à partir d'un objet, soit en utilisant l'opérateur `()`, soit en utilisant les fonctions `apply()` et `call()`. 4D propose trois types d'objets Function :

- les fonctions natives, c'est-à-dire les fonctions intégrées de diverses classes 4D telles que `collection.sort()` ou `file.copyTo()`.
- les fonctions utilisateur, créées dans les `classes` utilisateur à l'aide du mot-clé `Function`.
- les fonctions de formule, c'est-à-dire les fonctions qui peuvent exécuter n'importe quelle formule 4D.

Sommaire

<code>.apply() : any</code>	
<code>.apply(thisObj : Object { ; formulaParams : Collection }) : any</code>	exécute l'objet <code>formula</code> auquel elle est appliquée et retourne la valeur résultante
<code>.call() : any</code>	
<code>.call(thisObj : Object { ; ...params : any }) : any</code>	exécute l'objet <code>formula</code> auquel elle est appliquée et retourne la valeur résultante
<code>.source : Text</code>	contient l'expression source de <code>formula</code> sous forme de texte

Formula

► Historique

`Formula (formulaExp : Expression) : 4D.Function`

Paramètres	Type		Description
<code>formulaExp</code>	<code>Expression</code>	<code>-></code>	Formule à retourner comme objet
<code>Résultat</code>	<code>4D.Function</code>	<code><-</code>	Fonction native encapsulant la formule

Description

La commande `Formula` crée un objet `4D Function` basé sur l'expression `formulaExp`. `formulaExp` peut être aussi simple qu'une valeur unique ou complexe, comme une méthode projet avec des paramètres.

Le fait d'avoir une formule en tant qu'objet permet de la passer en tant que paramètre (champ calculé) à des commandes ou à des méthodes ou de l'exécuter à partir de divers composants sans avoir à les déclarer comme "partagés par les composants et la base de données hôte". Lorsqu'il est appelé, l'objet `formula` est évalué dans le contexte de la base de données ou du composant qui l'a créé.

La formule retournée peut être appelée avec :

- les méthodes `.call()` ou `.apply()`, ou
- la syntaxe de la notation objet (voir [objet formula](#)).

```
var $f : 4D.Function
$f:=Formula(1+2)
$o:=New object("myFormula";$f)

//three different ways to call the formula
$f.call($o) //retourne 3
$f.apply($o) //retourne 3
$o.myFormula() //retourne 3
```

Vous pouvez passer des [paramètres](#) à `Formula`, comme le montre l'[exemple 4](#) ci-dessous.

Vous pouvez indiquer l'objet sur lequel la formule est exécutée, comme le montre l'[exemple 5](#). Les propriétés de l'objet sont alors accessibles via la commande `This`.

Si `formulaExp` utilise des variables locales, leurs valeurs sont copiées et stockées dans l'objet formule retourné lors de sa création. Lors de son exécution, la formule utilise ces valeurs copiées plutôt que la valeur courante des variables locales. A noter que l'utilisation de tableaux comme variables locales n'est pas prise en charge.

L'objet créé par `Formula` peut être enregistré par exemple, dans un champ de base de données ou dans un document blob.

Exemple 1

Une formule simple :

```
var $f : 4D.Function
$f:=Formula(1+2)

var $o : Object
$o:=New object("f";$f)

$result:=$o.f() // retourne 3
```

Exemple 2

Une formule utilisant des variables locales :

```
$value:=10
$o:=New object("f";Formula($value))
$value:=20

$result:=$o.f() // retourne 10
```

Exemple 3

Une formule simple utilisant des paramètres :

```
$o:=New object("f";Formula($1+$2))
$result:=$o.f(10;20) //retourne 30
```

Exemple 4

Une formule utilisant une méthode projet avec des paramètres :

```
$o:=New object("f";Formula(myMethod))
$result:=$o.f("param1";"param2") // équivalent à $result:=myMethod("param1";"param2")
```

Exemple 5

Avec l'utilisation de `This` :

```
$o:=New object("fullName";Formula(This.firstName+" "+This.lastName))
$o.firstName:="John"
$o.lastName:="Smith"
$result:=$o.fullName() //retourne "John Smith"
```

Exemple 6

Appeler une formule à l'aide de la notation objet :

```
var $feta; $robot : Object
var $calc : 4D.Function
$robot:=New object("name";"Robot";"price";543;"quantity";2)
$feta:=New object("name";"Feta";"price";12.5;"quantity";5)

$calc:=Formula(This.total:=This.price*This.quantity)

//définit la formule aux propriétés d'objet
$feta.calc:=$calc
$robot.calc:=$calc

//appeler la formule
$feta.calc() // $feta={name:Feta,price:12.5,quantity:5,total:62.5,calc:"[object Formula]"}
$robot.calc() // $robot={name:Robot,price:543,quantity:2,total:1086,calc:"[object Formula]"}
```

Formula from string

► Historique

Formula from string(*formulaString* : Text) : 4D.Function

Paramètres	Type		Description
formulaString	Text	->	Formule texte à retourner comme objet
Résultat	4D.Function	<-	Objet natif encapsulant la formule

Description

La commande `Formula from string` crée un objet 4D.Function basé sur *formulaString*. *formulaString* peut être aussi simple qu'une valeur unique ou complexe, comme une méthode projet avec des paramètres.

Cette commande est similaire à `Formula`, sauf qu'elle traite une formule de type texte. Dans la plupart des cas, il est

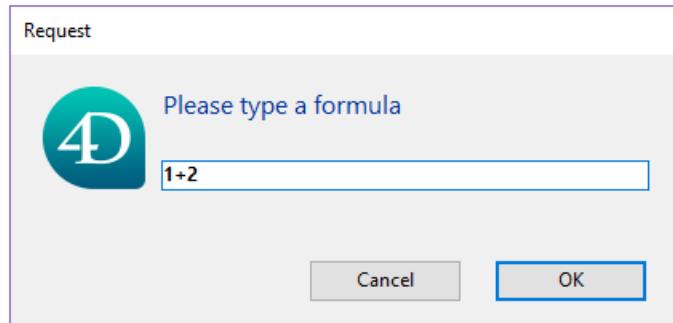
recommandé d'utiliser la commande `Formula`. `Formula from string` ne doit être utilisée que lorsque la formule d'origine a été exprimée sous forme de texte (par exemple, stockée en externe dans un fichier JSON). Dans ce contexte, l'utilisation de la syntaxe avec des tokens est fortement conseillée.

Le contenu des variables locales n'étant pas accessible par son nom en mode compilé, il ne peut pas être utilisé dans la `formulaString`. Si vous tentez d'accéder à une variable locale avec `Formula from string`, cela générera une erreur (-10737).

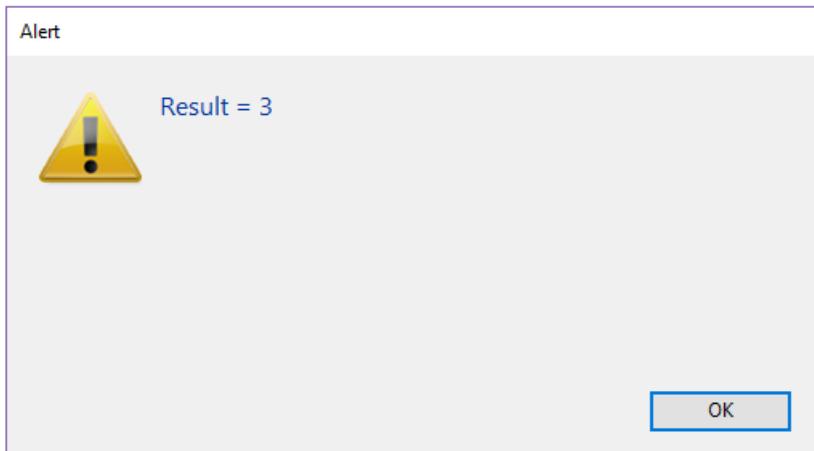
Exemple

Le code suivant permettra de créer un dialogue acceptant une formule dans un format texte :

```
var $textFormula : Text
var $f : 4D.Function
$textFormula:=Request("Please type a formula")
If(ok=1)
  $f:=Formula from string($textFormula)
  ALERT("Result = "+String($f.call()))
End if
```



... et exécute la formule :



.apply()

► Historique

`.apply() : any`

`.apply(thisObj : Object { ; formulaParams : Collection }) : any`

Paramètres	Type		Description
thisObj	Object	->	Objet à retourner par la commande This dans la formule
formulaParams	Collection	->	Collection des valeurs à passer en tant que \$1...\$n lorsque formula est exécuté
Résultat	any	<-	Valeur obtenue à partir de l'exécution de la formule

Description

La fonction `.apply()` exécute l'objet `formula` auquel elle est appliquée et retourne la valeur résultante. L'objet `formula` peut être créé à l'aide des commandes `Formula` ou `Formula from string`.

Dans le paramètre `thisObj`, vous pouvez passer une référence à l'objet à utiliser en tant que `This` dans la formule.

Vous pouvez également passer une collection à utiliser comme paramètres `$1...$n` dans la formule à l'aide du paramètre facultatif `formulaParams`.

A noter que `.apply()` est similaire à `.call()`, néanmoins les paramètres sont passés sous forme de collection. Cela peut être utile pour passer des résultats calculés.

Exemple 1

```
var $f : 4D.Function
$f:=Formula($1+$2+$3)

$c:=New collection(10;20;30)
$result:=$f.apply(NULL;$c) // retourne 60
```

Exemple 2

```
var $calc : 4D.Function
var $feta; $robot : Object
$robot:=New object("name";"Robot";"price";543;"quantity";2)
$feta:=New object("name";"Feta";"price";12.5;"quantity";5)

$calc:=Formula(This.total:=This.price*This.quantity)

$calc.apply($feta) // $feta={name:Feta,price:12.5,quantity:5,total:62.5}
$calc.apply($robot) // $robot={name:Robot,price:543,quantity:2,total:1086}
```

.call()

► Historique

`.call()` : any

`.call(thisObj : Object { ; ...params : any })` : any

Paramètres	Type		Description
thisObj	Object	->	Objet à retourner par la commande This dans la formule
params	any	->	Valeur(s) à passer en tant que \$1...\$n lorsque formula est exécuté
Résultat	any	<-	Valeur obtenue à partir de l'exécution de la formule

Description

La fonction `.call()` exécute l'objet `formula` auquel elle est appliquée et retourne la valeur résultante. L'objet `formula` peut être créé à l'aide des commandes `Formula` ou `Formula from string`.

Dans le paramètre `thisObj`, vous pouvez passer une référence à l'objet à utiliser en tant que `This` dans la formule.

Vous pouvez également passer des valeurs à utiliser comme paramètres `$1...$n` dans la formule à l'aide du paramètre facultatif `params`.

A noter que `.call()` est similaire à `.apply()`, néanmoins les paramètres sont passés directement.

Exemple 1

```
var $f : 4D.Function  
$f:=Formula(Uppercase($1))  
$result:=$f.call(Null;"hello") // retourne "HELLO"
```

Exemple 2

```
$o:=New object("value";50)  
$f:=Formula(This.value*2)  
$result:=$f.call($o) // retourne 100
```

.SOURCE

► Historique

`.source` : Text

Description

La propriété `.source` contient l'expression source de `formula` sous forme de texte.

Cette propriété est en lecture seule.

Exemple

```
var $of : 4D.Function  
var $tf : Text  
$of:=Formula(String(Current time;HH MM AM PM))  
$tf:=$of.source //"String(Current time;HH MM AM PM)"
```

IMAPTransporter

La classe `IMAPTransporter` vous permet de récupérer des messages à partir d'un serveur de messagerie IMAP.

Objet IMAP Transporter

Les objets IMAP Transporter sont instanciés avec la commande [IMAP New transporter](#). Leurs propriétés et fonctions sont les suivantes :

`.acceptUnsecureConnection : Boolean`

True si 4D est autorisé à établir une connexion non chiffrée

`.addFlags(msgIDs : Collection ; keywords : Object) : Object`

`.addFlags(msgIDs : Text ; keywords : Object) : Object`

`.addFlags(msgIDs : Longint ; keywords : Object) : Object`

ajoute des flags (drapeaux) aux `msgIDs` pour les `keywords` définis

`.append(mailObj : Object ; destinationBox : Text ; options : Object) : Object`

ajoute l'objet `mailObj` à la boîte `destinationBox`

`.authenticationMode : Text`

le mode d'authentification utilisé pour ouvrir la session sur le serveur de mail

`.checkConnection() : Object`

vérifie la connexion à l'aide des informations stockées dans l'objet transporteur

`.checkConnectionDelay : Integer`

durée maximale (en secondes) autorisée avant vérification de la connexion au serveur

`.connectionTimeOut : Integer`

la durée maximale (en secondes) autorisée avant vérification de la connexion au serveur

`.copy(msgsIDs : Collection ; destinationBox : Text) : Object`

`.copy(allMsgs : Integer ; destinationBox : Text) : Object`

copie les messages définis dans `msgsIDs` ou `allMsgs` dans la `destinationBox` sur le serveur IMAP

`.createBox(name : Text) : Object`

crée une nouvelle mailbox avec le `name` passé en paramètre

`.delete(msgsIDs : Collection) : Object`

`.delete(allMsgs : Integer) : Object`

associe le marqueur "deleted" aux messages désignés par `msgsIDs` ou `allMsgs`

`.deleteBox(name : Text) : Object`

supprime définitivement la boîte de réception nommée `name` sur le serveur IMAP

`.expunge() : Object`

supprime tous les messages marqués "deleted" du serveur de messagerie IMAP.

`.getBoxInfo({ name : Text }) : Object`

retourne un objet `boxInfo` correspondant à la boîte de réception courante ou à la boîte nommée `name`

`.getBoxList({ parameters : Object }) : Collection`

retourne une collection de boîtes de réception décrivant toutes les boîtes de réception disponibles

<code>.getDelimiter() : Text</code>	retourne le caractère utilisé pour délimiter les niveaux de hiérarchie dans le nom de la boîte de réception
<code>.getMail(msgNumber: Integer { ; options : Object }) : Object</code> <code>.getMail(msgID: Text { ; options : Object }) : Object</code>	retourne l'objet <code>Email</code> correspondant au <code>msgNumber</code> ou <code>msgID</code> dans la boîte de réception désignée par <code>IMAP_transporter</code>
<code>.getMails(ids : Collection { ; options : Object }) : Object</code> <code>.getMails(startMsg : Integer ; endMsg : Integer { ; options : Object }) : Object</code>	retourne un objet contenant une collection d'objets <code>Email</code>
<code>.getMIMEAsBlob(msgNumber : Integer { ; updateSeen : Boolean }) : Blob</code> <code>.getMIMEAsBlob(msgID : Text { ; updateSeen : Boolean }) : Blob</code>	retourne un BLOB avec le contenu MIME du message correspondant au <code>msgNumber</code> ou <code>msgID</code> dans la boîte de réception désignée par le <code>IMAP_transporter</code>
<code>.host : Text</code>	contient le nom ou l'adresse IP du serveur hôte
<code>.logFile : Text</code>	le chemin complet du fichier d'historique qui a été défini (le cas échéant) pour la connexion
<code>.move(msgsIDs : Collection ; destinationBox : Text) : Object</code> <code>.move(allMsgs : Integer ; destinationBox : Text) : Object</code>	déplace les messages définis dans <code>msgsIDs</code> ou <code>allMsgs</code> vers la <code>destinationBox</code> sur le serveur IMAP
<code>.numToID(startMsg : Integer ; endMsg : Integer) : Collection</code>	convertit les numéros de séquence en IDs uniques IMAP pour les messages de la plage séquentielle désignée par <code>startMsg</code> et <code>endMsg</code>
<code>.removeFlags(msgIDs : Collection ; keywords : Object) : Object</code> <code>.removeFlags(msgIDs : Text ; keywords : Object) : Object</code> <code>.removeFlags(msgIDs : Longint ; keywords : Object) : Object</code>	supprime des flags (drapeaux) des <code>msgIDs</code> pour les <code>keywords</code> définis
<code>.renameBox(currentName : Text ; newName : Text) : Object</code>	renomme une mailbox sur le serveur IMAP
<code>.port : Integer</code>	le numéro de port utilisé pour les transactions d'emails
<code>.searchMails(searchCriteria : Text) : Collection</code>	recherche les messages qui correspondent aux critères de recherche <code>searchCriteria</code> dans la boîte de réception courante
<code>.selectBox(name : Text { ; state : Integer }) : Object</code>	sélectionne la mailbox <code>name</code> comme étant la mailbox courante
<code>.subscribe(name : Text) : Object</code>	permet d'ajouter la mailbox spécifiée à l'ensemble des mailboxes auxquelles vous avez "souscrit" sur le serveur IMAP
<code>.unsubscribe(name : Text) : Object</code>	supprime la mailbox spécifiée de l'ensemble des mailboxes auxquelles vous avez "souscrit" sur le serveur IMAP
<code>.user : Text</code>	

Le nom d'utilisateur employé pour l'authentification sur le serveur mail
--

IMAP New transporter

► Historique

IMAP New transporter(*server* : Object) : 4D.IMAPTransporter

Paramètres	Type		Description
server	Object	->	Informations sur le serveur IMAP
Résultat	4D.IMAPTransporter	<-	Objet IMAP Transporter

Description

La commande `IMAP New transporter` configure une nouvelle connexion IMAP en fonction du paramètre *server* et retourne un nouvel objet *transporter*. L'objet transporteur retourné sera alors utilisé pour la réception d'emails.

Dans le paramètre *server*, passez un objet contenant les propriétés suivantes :

<code>server</code>		Valeur par défaut (si omise)
<code>.acceptUnsecureConnection : Boolean</code> True si 4D est autorisé à établir une connexion non chiffrée		Faux
<code>.accessTokenOAuth2: Text</code> <code>.accessTokenOAuth2: Object</code> Chaîne de texte ou objet token représentant les informations d'autorisation OAuth2. Il est utilisé uniquement avec <code>OAUTH2 authenticationMode</code> . Si <code>accessTokenOAuth2</code> est utilisé mais que <code>authenticationMode</code> est omis, le protocole OAuth 2 est utilisé (si le serveur l'autorise). Non retourné en objet <i>IMAP transporter</i> .		aucune
<code>.authenticationMode : Text</code> le mode d'authentification utilisé pour ouvrir la session sur le serveur de mail		le mode d'authentification le plus sûr pris en charge par le serveur est utilisé
<code>.checkConnectionDelay : Integer</code> durée maximale (en secondes) autorisée avant vérification de la connexion au serveur	300	
<code>.connectionTimeOut : Integer</code> la durée maximale (en secondes) autorisée avant vérification de la connexion au serveur	30	
<code>.host : Text</code> contient le nom ou l'adresse IP du serveur hôte		<i>obligatoire</i>
<code>.logFile : Text</code> le chemin complet du fichier d'historique qui a été défini (le cas échéant) pour la connexion		aucune
<code>.password : Text</code> Mot de passe de l'utilisateur pour l'authentification sur le serveur. Non retourné en objet <i>IMAP transporter</i> .		aucune
<code>.port : Integer</code> le numéro de port utilisé pour les transactions d'emails	993	
<code>.user : Text</code> le nom d'utilisateur employé pour l'authentification sur le serveur mail		aucune

Attention : Assurez-vous que le timeout défini est inférieur au timeout du serveur, sinon le timeout du client sera inutile.

Résultat

La fonction retourne un [objet IMAP transporter](#). Toutes les propriétés retournées sont en lecture seule.

La connexion IMAP est automatiquement fermée lorsque l'objet transporteur est détruit.

Exemple

```

$server:=New object
$server.host:="imap.gmail.com" //Obligatoire
$server.port:=993
$server.user:="4d@gmail.com"
$server.password:="XXXXXXX"
$server.logFile:="LogTest.txt" //log à sauvegarder dans le dossier Logs

var $transporter : 4D.IMAPTransporter
$transporter:=IMAP New transporter($server)

$status:=$transporter.checkConnection()
If(Not($status.success))
    ALERT("An error occurred: "+$status.statusText)
End if

```

4D.IMAPTransporter.new()

4D.IMAPTransporter.new(server : Object) : 4D.IMAPTransporter

Paramètres	Type		Description
server	Object	->	Informations sur le serveur IMAP
Résultat	4D.IMAPTransporter	<-	Objet IMAP Transporter

Description

La fonction `4D.IMAPTransporter.new()` crée et retourne un nouvel objet de type `4D.IMAPTransporter`. Elle est identique à la commande `IMAP New transporter` (raccourci).

.acceptUnsecureConnection

► Historique

.acceptUnsecureConnection : Boolean

Description

La propriété `.acceptUnsecureConnection` contient True si 4D est autorisé à établir une connexion non chiffrée lorsqu'une connexion chiffrée n'est pas possible.

Elle contient False si les connexions non chiffrées ne sont pas autorisées, auquel cas une erreur est retournée lorsque la connexion chiffrée n'est pas possible.

Ports sécurisés disponibles :

- SMTP
 - 465: SMTPS
 - 587 ou 25 : SMTP avec mise à niveau STARTTLS si le serveur le prend en charge.
- IMAP
 - 143 : Port IMAP non chiffré
 - 993 : IMAP avec mise à niveau STARTTLS si le serveur le prend en charge
- POP3
 - 110 : Port POP3 non chiffré
 - 995 : POP3 avec mise à niveau STARTTLS si le serveur le prend en charge.

.addFlags()

► Historique

`.addFlags(msgIDs : Collection ; keywords : Object) : Object`

`.addFlags(msgIDs : Text ; keywords : Object) : Object`

`.addFlags(msgIDs : Longint ; keywords : Object) : Object`

Paramètres	Type	Description
msgIDs	Collection	-> Collection de chaînes : IDs uniques des messages (texte) Texte : ID unique d'un message Numérique (IMAP all) : Tous les messages de la boîte sélectionnée
keywords	Object	-> Mots-clés de flags à ajouter
Résultat	Object	<- Statut de l'opération addFlags

Description

La fonction `.addFlags()` ajoute des flags (drapeaux) aux `msgIDs` pour les `keywords` définis.

Dans le paramètre `msgIDs`, vous pouvez passer soit :

- une *collection* contenant les IDs uniques de messages spécifiques, ou
- l'ID unique (*texte*) d'un seul message ou
- la constante suivante (*entier long*) pour tous les messages de la boîte sélectionnée :

Constante	Valeur	Commentaire
IMAP all	1	Sélectionner tous les messages de la boîte sélectionnée

Le paramètre `keywords` vous permet de passer un objet avec des valeurs de mots-clés pour les flags spécifiques à ajouter à `msgIDs`. Vous pouvez utiliser les mots-clés suivants :

Paramètres	Type	Description
\$draft	Boolean	True pour ajouter le marqueur "draft" au message
\$seen	Booléen	True pour ajouter le marqueur "seen" au message
\$flagged	Booléen	True pour ajouter le marqueur "flagged" au message
\$answered	Booléen	True pour ajouter le marqueur "répondu" au message
\$deleted	Booléen	True pour ajouter le marqueur "supprimé" au message

- Les valeurs à faux sont ignorées.
- L'interprétation des indicateurs de mots-clés peut varier selon le client de messagerie.

Objet retourné

La fonction retourne un objet décrivant le statut IMAP :

Propriété	Type	Description
success	Boolean	Vrai si l'opération est réussie, sinon Faux
statusText	Text	Message du statut retourné par le serveur IMAP, ou dernière erreur retournée dans la pile d'erreurs 4D
errors	Collection	Pile d'erreurs 4D (non retournée si une réponse du serveur IMAP est reçue)
[].errcode	Number	Code d'erreur 4D
[].message	Text	Description de l'erreur 4D
[].componentSignature	Text	Signature du composant interne qui a retourné l'erreur

Exemple

```
var $options;$transporter;$boxInfo;$status : Object  
  
$options:=New object  
$options.host:="imap.gmail.com"  
$options.port:=993  
$options.user:="4d@gmail.com"  
$options.password:="xxxxx"  
  
// Créer transporteur  
$transporter:=IMAP New transporter($options)  
  
// Sélectionner la boîte de messagerie  
$boxInfo:=$transporter.selectBox("INBOX")  
  
// Marquer tous les messages de la boîte de réception (INBOX) comme étant lus/vus  
$flags:=New object  
$flags["$seen"]:=True  
$status:=$transporter.addFlags(IMAP all;$flags)
```

.append()

► Historique

.append(*mailObj* : Object ; *destinationBox* : Text ; *options* : Object) : Object

Paramètres	Type		Description
mailObj	Object	->	Objet email
destinationBox	Text	->	Mailbox devant recevoir l'objet email
options	Object	->	Objet contenant les informations de charset
Résultat	Object	<-	Statut de l'opération

Description

La fonction `.append()` ajoute l'objet `mailObj` à la boîte `destinationBox`.

Dans le paramètre `mailObj`, passez un objet email. Dans le paramètre `mailObj`, passez un objet email. La fonction `.append()` prend en charge les marqueurs de mots-clés dans l'attribut `keywords` des objets email.

Le paramètre optionnel `destinationBox` vous permet de passer le nom de la boîte de réception dans laquelle l'objet `mailObj` sera ajouté. S'il est omis, la mailbox courante sera utilisée.

Dans le paramètre optionnel `options`, vous pouvez passer un objet permettant de définir le charset et l'encodage des différentes parties de l'email. Propriétés disponibles :

Propriété	Type	Description
headerCharset	Text	Charset et encodage utilisés pour les parties de mail suivantes : le sujet, les noms de fichiers joints et le nom du mail. Valeurs possibles : voir le tableau des charsets possibles
bodyCharset	Text	Charset et encodage utilisés pour le contenu html et le texte du body du mail. Valeurs possibles : voir le tableau des charsets possibles

Charsets possibles :

Constante	Valeur	Commentaire
mail mode ISO2022JP	US-ASCII_ISO-2022-JP_UTF8_QP	<ul style="list-style-type: none"> headerCharset : US-ASCII si possible, japonais (ISO-2022-JP) & Quoted-printable si possible, sinon UTF-8 & Quoted-printable bodyCharset : US-ASCII si possible, japonais (ISO-2022-JP) et 7 bits si possible, sinon UTF-8 et Quoted-printable
mail mode ISO88591	ISO-8859-1	<ul style="list-style-type: none"> headerCharset: ISO-8859-1 & Quoted-printable bodyCharset: ISO-8859-1 & 8-bit
mail mode UTF8	US-ASCII_UTF8_QP	headerCharset & bodyCharset : US-ASCII si possible, sinon UTF-8 & Quoted-printable (valeur par défaut)
mail mode UTF8 in base64	US-ASCII_UTF8_B64	headerCharset & bodyCharset : US-ASCII si possible, sinon UTF-8 & base64

Objet retourné

La fonction retourne un objet décrivant le statut IMAP :

Propriété		Type	Description
success		Booléen	Vrai si l'opération est réussie, sinon Faux
statusText		Text	Message du statut retourné par le serveur IMAP, ou dernière erreur retournée dans la pile d'erreurs 4D
errors		Collection	Pile d'erreurs 4D (non retournée si une réponse du serveur IMAP est reçue)
	[].errcode	Nombre	Code d'erreur 4D
	[].message	Text	Description de l'erreur 4D
	[].componentSignature	Text	Signature du composant interne qui a retourné l'erreur

Exemple

Pour enregistrer un email dans la boîte Drafts :

```

var $settings; $status; $msg; $imap: Object

$settings:=New object("host"; "domain.com"; "user"; "xxxx"; "password"; "xxxx"; "port"; 993)

$imap:=IMAP New transporter($settings)

$msg:=New object
$msg.from:="xxxx@domain.com"
$msg.subject:="Lorem Ipsum"
$msg.textBody:="Lorem ipsum dolor sit amet, consectetur adipiscing elit."
$msg.keywords:=New object
$msg.keywords["$seen"]:=True//marquer le message comme étant lu
$msg.keywords["$draft"]:=True//marquer le message comme brouillon

$status:=$imap.append($msg; "Drafts")

```

.authenticationMode

► Historique

.authenticationMode : Text

Description

La propriété `.authenticationMode` contient le mode d'authentification utilisé pour ouvrir la session sur le serveur de mail.

Par défaut, le mode le plus sécurisé pris en charge par le serveur est utilisé.

Valeurs possibles :

Valeur	Constantes	Commentaire
CRAM-MD5	<code>IMAP authentication CRAM MD5</code>	Authentification à l'aide du protocole CRAM-MD5
LOGIN	<code>IMAP authentication login</code>	Authentification à l'aide du protocole LOGIN
OAuth2	<code>IMAP authentication OAuth2</code>	Authentification à l'aide du protocole OAuth2
PLAIN	<code>IMAP authentication plain</code>	Authentification à l'aide du protocole PLAIN

.checkConnection()

► Historique

`.checkConnection()` : Object

Paramètres	Type		Description
Résultat	Object	<-	Statut de la connexion de l'objet transporteur

Description

La fonction `.checkConnection()` vérifie la connexion à l'aide des informations stockées dans l'objet transporteur, recréée la connexion si nécessaire et retourne son statut. Cette fonction vous permet de vérifier que les valeurs fournies par l'utilisateur sont valides et cohérentes.

Objet retourné

La fonction envoie une requête au serveur de mail et retourne un objet décrivant le statut. Cet objet peut avoir les propriétés suivantes :

Propriété		Type	Description
success		boolean	Vrai si la vérification a été effectuée avec succès, sinon Faux
status		number	(SMTP uniquement) Code du statut retourné par le serveur de messagerie (0 en cas de problème non lié au traitement du mail)
statusText		text	Message du statut retourné par le serveur de messagerie, ou dernière erreur retournée dans la pile d'erreurs 4D
errors		collection	Pile d'erreurs 4D (non retournée si une réponse du serveur de messagerie est reçue)
	[].errCode	number	Code d'erreur 4D
	[].message	text	Description de l'erreur 4D
	[].componentSignature	text	Signature du composant interne qui a retourné l'erreur

.checkConnectionDelay

► Historique

`.checkConnectionDelay` : Integer

Description

La propriété `.checkConnectionDelay` contient la durée maximale (en secondes) autorisée avant vérification de la connexion au serveur. Si cette durée est dépassée entre deux appels de méthodes, la connexion au serveur sera vérifiée. Par défaut, si la propriété n'a pas été définie dans l'objet `server`, la valeur est de 300.

Attention : Assurez-vous que le timeout défini est inférieur au timeout du serveur, sinon le timeout du client sera inutile.

`.connectionTimeOut`

► Historique

`.connectionTimeOut` : Integer

Description

La propriété `.connectionTimeOut` contient la durée maximale (en secondes) autorisée avant vérification de la connexion au serveur. Par défaut, si la propriété n'a pas été définie dans l'objet `server` (utilisé pour créer l'objet transporteur avec `SMTP New transporter`, `POP3 New transporter`, `IMAP New transporter`), la valeur utilisée est 30.

`.copy()`

► Historique

`.copy(msgsIDs : Collection ; destinationBox : Text) : Object`

`.copy(allMsgs : Integer ; destinationBox : Text) : Object`

Paramètres	Type		Description
msgsIDs	Collection	->	Collection d'IDs uniques de messages (texte)
allMsgs	Integer	->	<code>IMAP all</code> : tous les messages de la boîte de réception sélectionnée
destinationBox	Text	->	Boîte de réception recevant les messages copiés
Résultat	Object	<-	Statut de l'opération de copie

Description

La fonction `.copy()` copie les messages définis dans `msgsIDs` ou `allMsgs` dans la `destinationBox` sur le serveur IMAP.

Vous pouvez passer :

- dans le paramètre `msgsIDs`, une collection contenant les ID uniques des messages spécifiques à copier, ou
- dans le paramètre `allMsgs`, la constante `IMAP all` (integer) pour copier tous les messages de la boîte de réception sélectionnée.

Le paramètre `destinationBox` vous permet de passer une valeur texte avec le nom de la boîte de réception dans laquelle seront placés les messages copiés.

Objet retourné

La fonction retourne un objet décrivant le statut IMAP :

Propriété		Type	Description
success		Booléen	Vrai si l'opération est réussie, sinon Faux
statusText		Text	Message du statut retourné par le serveur IMAP, ou dernière erreur retournée dans la pile d'erreurs 4D
errors		Collection	Pile d'erreurs 4D (non retournée si une réponse du serveur IMAP est reçue)
	[].errcode	Nombre	Code d'erreur 4D
	[].message	Text	Description de l'erreur 4D
	[].componentSignature	Text	Signature du composant interne qui a retourné l'erreur

Exemple 1

Pour copier une sélection de messages :

```

var $server;$boxInfo;$status : Object
var $mailIds : Collection
var $transporter : 4D.IMAPTransporter

$server:=New object
$server.host:="imap.gmail.com" //Obligatoire
$server.port:=993
$server.user:="4d@gmail.com"
$server.password:="XXXXXXX"

$transporter:=IMAP New transporter($server)

//sélectionner la boite de réception
$boxInfo:=$transporter.selectBox("inbox")

//obtenir la collection des IDs uniques de message
$mailIds:=$transporter.searchMails("subject \\"4D new feature:\\\"")

// copier les messages identifiés dans la boîte de réception "documents"
$status:=$transporter.copy($mailIds;"documents")

```

Exemple 2

Pour copier tous les messages de la boîte de réception courante :

```

var $server;$boxInfo;$status : Object
var $transporter : 4D.IMAPTransporter

$server:=New object
$server.host:="imap.gmail.com" //Obligatoire
$server.port:=993
$server.user:="4d@gmail.com"
$server.password:="XXXXXXX"

$transporter:=IMAP New transporter($server)

//sélectionner la boite de réception

$boxInfo:=$transporter.selectBox("inbox")

// copier les messages dans la boîte de réception "documents"
$status:=$transporter.copy(IMAP all;"documents")

```

.createBox()

► Historique

.createBox(*name* : Text) : Object

Paramètres	Type		Description
name	Text	->	Nom de la nouvelle mailbox
Résultat	Object	<-	Statut de l'opération de création de mailbox

Description

La fonction `.createBox()` crée une nouvelle mailbox avec le `name` passé en paramètre. Si le caractère séparateur de hiérarchie du serveur IMAP apparaît dans le nom de la mailbox, le serveur IMAP créera tous les boîtes parentes nécessaires pour créer la boîte donnée.

Par exemple, si vous essayez de créer "Projects/IMAP/Doc" dans un serveur dont le "/" est le caractère séparateur de hiérarchie, les éléments suivants seront créés :

- Uniquement la boîte "Doc" si "Projects" & "IMAP" existent déjà.
- Les boîtes "IMAP" & "Doc" si seule "Projects" existe déjà.
- Les boîtes "Projects" & "IMAP" & "Doc" si elles n'existent pas déjà.

Dans le paramètre `name`, passez le nom de la mailbox à créer.

Objet retourné

La fonction retourne un objet décrivant le statut IMAP :

Propriété		Type	Description
success		Booléen	Vrai si l'opération est réussie, sinon Faux
statusText		Text	Message du statut retourné par le serveur IMAP, ou dernière erreur retournée dans la pile d'erreurs 4D
errors		Collection	Pile d'erreurs 4D (non retournée si une réponse du serveur IMAP est reçue)
	[].errcode	Nombre	Code d'erreur 4D
	[].message	Text	Description de l'erreur 4D
	[].componentSignature	Text	Signature du composant interne qui a retourné l'erreur

Exemple

Pour créer une nouvelle boîte "Invoices" :

```

var $pw : text
var $options; $transporter; $status : object

$options:=New object

$pw:=Request("Please enter your password:")
If(OK=1)
$options.host:="imap.gmail.com"
$options.user:="test@gmail.com"
$options.password:=$pw

$transporter:=IMAP New transporter($options)

$status:=$transporter.createBox("Invoices")

If ($status.success)
ALERT("Mailbox creation successful!")
End for each
End if

```

.delete()

► Historique

.delete(*msgsIDs* : Collection) : Object
 .delete(*allMsgs* : Integer) : Object

Paramètres	Type		Description
<i>msgsIDs</i>	Collection	->	Collection d'IDs uniques de messages (texte)
<i>allMsgs</i>	Integer	->	IMAP all : tous les messages de la boîte de réception sélectionnée
Résultat	Object	<-	Statut de l'opération de suppression

Description

La fonction `.delete()` associe le marqueur "deleted" aux messages désignés par `msgsIDs` ou `allMsgs`.

Vous pouvez passer :

- dans le paramètre `msgsIDs`, une collection contenant les ID uniques des messages spécifiques à supprimer, ou
- dans le paramètre `allMsgs`, la constante `IMAP all` (integer) pour supprimer tous les messages de la boîte de réception sélectionnée.

L'exécution de cette fonction ne supprime pas réellement les messages. Les messages ayant le marqueur "deleted" peuvent toujours être trouvés par la fonction `.searchMails()`. Les messages marqués sont supprimés du serveur IMAP uniquement avec `.expunge()` ou en sélectionnant une autre boîte de réception ou lorsque l'objet `transporter` (créé avec `IMAP New transporter`) est détruit.

Objet retourné

La fonction retourne un objet décrivant le statut IMAP :

Propriété		Type	Description
success		Booléen	Vrai si l'opération est réussie, sinon Faux
statusText		Text	Message du statut retourné par le serveur IMAP, ou dernière erreur retournée dans la pile d'erreurs 4D
errors		Collection	Pile d'erreurs 4D (non retournée si une réponse du serveur IMAP est reçue)
	[].errcode	Nombre	Code d'erreur 4D
	[].message	Text	Description de l'erreur 4D
	[].componentSignature	Text	Signature du composant interne qui a retourné l'erreur

Exemple 1

Pour supprimer une sélection de messages :

```

var $server;$boxInfo;$status : Object
var $mailIds : Collection
var $transporter : 4D.IMAPTransporter

$server:=New object
$server.host:="imap.gmail.com" //Obligatoire
$server.port:=993
$server.user:="4d@gmail.com"
$server.password:="XXXXXXX"

$transporter:=IMAP New transporter($server)

//sélectionner la boite de réception
$boxInfo:=$transporter.selectBox("Inbox")

//obtenir la collection d'IDs unique des messages
$mailIds:=$transporter.searchMails("subject \"Reports\"")

// Supprimer les messages sélectionnés
$status:=$transporter.delete($mailIds)

```

Exemple 2

Pour supprimer tous les messages de la boîte de réception courante :

```

var $server;$boxInfo;$status : Object
var $transporter : 4D.IMAPTransporter

$server:=New object
$server.host:="imap.gmail.com" //Obligatoire
$server.port:=993
$server.user:="4d@gmail.com"
$server.password:="XXXXXXX"

$transporter:=IMAP New transporter($server)

//selection de la boîte
$boxInfo:=$transporter.selectBox("Junk Email")

// suppression de tous les messages de la boîte courante
$status:=$transporter.delete(IMAP all)

```

.deleteBox()

► Historique

.deleteBox(*name* : Text) : Object

Paramètres	Type		Description
name	Text	->	Nom de la boîte de réception à supprimer

|Result|Object|<-|Statut de l'opération de suppression|

Description

La fonction `.deleteBox()` supprime définitivement la boîte de réception nommée `name` sur le serveur IMAP. Si vous tentez de supprimer la boîte INBOX ou une boîte de réception qui n'existe pas, une erreur sera générée.

Dans le paramètre `name`, passez le nom de la mailbox à supprimer.

- La fonction ne peut pas supprimer une boîte de réception contenant des boîtes "enfant" si la boîte "parente" dispose de l'attribut "`¥Noselect`".
- Tous les messages présents dans la boîte seront également supprimés.
- La possibilité de supprimer une boîte de réception dépend du serveur de messagerie.

Objet retourné

La fonction retourne un objet décrivant le statut IMAP :

Propriété		Type	Description
success		Booléen	Vrai si l'opération est réussie, sinon Faux
statusText		Text	Message du statut retourné par le serveur IMAP, ou dernière erreur retournée dans la pile d'erreurs 4D
errors		Collection	Pile d'erreurs 4D (non retournée si une réponse du serveur IMAP est reçue)
	[].errcode	Nombre	Code d'erreur 4D
	[].message	Text	Description de l'erreur 4D
	[].componentSignature	Text	Signature du composant interne qui a retourné l'erreur

Exemple

Vous souhaitez supprimer la boîte enfant "Nova Orion Industries" à l'intérieur de la boîte "Bills" :

```

var $pw; $name : text
var $options; $transporter; $status : object

$options:=New object

$pw:=Request("Please enter your password:")

If(OK=1) $options.host:="imap.gmail.com"
$options.user:="test@gmail.com"
$options.password:=$pw

$transporter:=IMAP New transporter($options)

// delete mailbox
$name:="Bills"+$transporter.getDelimiter()+"Nova Orion Industries"
$status:=$transporter.deleteBox($name)

If ($status.success)
    ALERT("Mailbox deletion successful!")
Else
    ALERT("Error: "+$status.statusText)
End if
End if
Else
    ALERT("Error: "+$status.statusText)
End if
End if

```

.expunge()

► Historique

.expunge() : Object

Paramètres	Type		Description
Résultat	Object	<-	Statut de l'opération expunge

Description

La fonction `.expunge()` supprime tous les messages marqués "deleted" du serveur de messagerie IMAP. Le marqueur "deleted" peut être fixé avec les fonctions `.delete()` ou `.addFlags()`.

Objet retourné

La fonction retourne un objet décrivant le statut IMAP :

Propriété		Type	Description
success		Booléen	Vrai si l'opération est réussie, sinon Faux
statusText		Text	Message du statut retourné par le serveur IMAP, ou dernière erreur retournée dans la pile d'erreurs 4D
errors		Collection	Pile d'erreurs 4D (non retournée si une réponse du serveur IMAP est reçue)
	[].errcode	Nombre	Code d'erreur 4D
	[].message	Text	Description de l'erreur 4D
	[].componentSignature	Text	Signature du composant interne qui a retourné l'erreur

Exemple

```

var $options;$transporter;$boxInfo;$status : Object
var $ids : Collection

$options:=New object
$options.host:="imap.gmail.com"
$options.port:=993
$options.user:="4d@gmail.com"
$options.password:="xxxxx"

// Creation du transporter
$transporter:=IMAP New transporter($options)

// Sélection de la boîte
$boxInfo:=$transporter.selectBox("INBOX")

// Chercher et sélectionner les messages lus dans INBOX
$ids:=$transporter.searchMails("SEEN")
$status:=$transporter.delete($ids)

// Purger tous les messages marqués deleted
$status:=$transporter.expunge()

```

.getBoxInfo()

► Historique

.getBoxInfo({ name : Text }) : Object

Paramètres	Type		Description
name	Text	->	Nom de la boîte de réception
Résultat	Object	<-	Objet boxInfo

Description

La fonction `.getBoxInfo()` retourne un objet `boxInfo` correspondant à la boîte de réception courante ou à la boîte nommée *name*. Cette fonction retourne les mêmes informations que `.selectBox()` mais sans modifier la boîte de réception courante.

Dans le paramètre optionnel *name* passez le nom de la boîte de réception à laquelle vous souhaitez accéder. Le nom doit représenter une hiérarchie claire, de gauche à droite, avec des niveaux séparés par un caractère délimiteur spécifique. Le délimiteur peut être récupéré à l'aide de la fonction `.getDelimiter()`.

Si la boîte de réception nommée *name* n'est pas sélectionnable ou n'existe pas, la fonction génère une erreur et retourne null.

Objet retourné

L'objet `boxInfo` contient les propriétés suivantes :

Propriété	Type	Description
name	Texte	Nom de la boîte de réception
mailCount	number	Nombre de messages contenus dans la boîte de réception
mailRecent	number	Nombre de messages portant le marqueur "récent" (indiquant les nouveaux messages)

Exemple

```

var $transporter : 4D.IMAPTransporter
$transporter:=IMAP New transporter($server)

$info:=$transporter.getBoxInfo("INBOX")
ALERT("INBOX contains "+String($info.mailRecent)+" recent emails.")

```

.getBoxList()

► Historique

.getBoxList({ parameters : Object }) : Collection

Paramètres	Type		Description
parameters	Object	->	Objet de paramètre
Résultat	Collection	<-	Collection d'objets mailbox

Description

La fonction `.getBoxList()` retourne une collection de boîtes de réception décrivant toutes les boîtes de réception disponibles. Cette fonction vous permet de gérer localement les listes de messages situés sur le serveur de messagerie IMAP.

Dans le paramètre optionnel `parameters`, passez un objet contenant des valeurs de filtrage des boîtes de réception retournées. Vous pouvez passer :

Propriété	Type	Description
isSubscribed	Booléen	<ul style="list-style-type: none"> True pour retourner uniquement les boîtes auxquelles vous êtes abonné False pour retourner toutes les boîtes de réception disponibles

Résultat

Chaque objet de la collection retournée contient les propriétés suivantes :

Propriété	Type	Description
[].name	Texte	Nom de la boîte de réception
[].selectable	boolean	Indique si les droits d'accès permettent ou non à la boîte de réception d'être sélectionnée : <ul style="list-style-type: none"> vrai - la boîte de réception peut être sélectionnée faux - la boîte de réception ne peut pas être sélectionnée
[].inferior	boolean	Indique si les droits d'accès permettent ou non la création d'une hiérarchie inférieure dans la boîte de réception : <ul style="list-style-type: none"> vrai - un niveau inférieur peut être créé faux - un niveau inférieur ne peut pas être créé
[].interesting	boolean	Indique si la boîte de réception a été marquée comme "intéressante" par le serveur : <ul style="list-style-type: none"> vrai - la boîte de réception a été marquée comme "intéressante" par le serveur. Par exemple, elle peut contenir de nouveaux messages. faux - la boîte de réception n'a pas été marquée comme "intéressante" par le serveur.

Si le compte ne contient pas de boîtes de réception, une collection vide est retournée.

- Si aucune connexion n'est ouverte, `.getBoxList()` ouvrira une connexion.

- Si la connexion n'a pas été utilisée depuis le délai de connexion (voir `IMAP New transporter`), la fonction `.checkConnection()` est automatiquement appelée.

Exemple

```

var $transporter : 4D.IMAPTransporter
$transporter:=IMAP New transporter($server)

$boxList:=$transporter.getBoxList()

For each($box;$boxList)
  If($box.interesting)
    $split:=Split string($box.name;$transporter.getDelimiter())
    ALERT("New emails are available in the box: "+$split[$split.length-1])
  End if
End for each

```

.getDelimiter()

► Historique

`.getDelimiter()` : Text

Paramètres	Type		Description
Résultat	Text	<-	Caractère de délimitation de hiérarchie

Description

La fonction `.getDelimiter()` retourne le caractère utilisé pour délimiter les niveaux de hiérarchie dans le nom de la boite de réception.

Le délimiteur est un caractère pouvant être utilisé pour :

- créer des boites de réception de niveau inférieur
- rechercher des niveaux inférieurs ou supérieurs dans la hiérarchie de la boite de réception

Résultat

Caractère de délimitation des noms de boites de réception.

- Si aucune connexion n'est ouverte, `.getDelimiter()` ouvrira une connexion.
- Si la connexion n'a pas été utilisée depuis le [délai de connexion](#), la fonction

`.checkConnection()` est automatiquement appelée.

Exemple

```

var $transporter : 4D.IMAPTransporter
$transporter:=IMAP New transporter($server)

$boxList:=$transporter.getBoxList()

For each($box;$boxList)
  If($box.interesting)
    $split:=Split string($box.name;$transporter.getDelimiter())
    ALERT("New emails are available in the box: "+$split[$split.length-1])

```

```
End if  
End for each
```

```
## .getMail()  
  
<details><summary>Historique</summary>
```

Version	Modifications
v18 R4	Ajout

.getMail(*msgNumber*: Integer { ; *options* : Object }) : Object
.getMail(*msgID*: Text { ; *options* : Object }) : Object

Paramètres	Type		Description
<i>msgNumber</i>	Integer	->	Numéro de séquence du message
<i>msgID</i>	Text	->	ID unique du message
<i>options</i>	Object	->	Instructions sur la gestion du message
Résultat	Object	<-	Objet email

Description

La fonction `.getMail()` retourne l'objet `Email` correspondant au *msgNumber* ou *msgID* dans la boite de réception désignée par `IMAP_transporter`. Cette fonction vous permet de gérer localement le contenu de l'email.

Dans le premier paramètre, vous pouvez passer soit :

- *msgNumber*, une valeur *integer* indiquant le numéro de séquence du message à récupérer ou
- *msgID*, une valeur *text* indiquant l'ID unique du message à récupérer.

Le paramètre facultatif *options* vous permet de passer un objet définissant des instructions supplémentaires pour la gestion de votre message. Les propriétés suivantes sont disponibles :

Propriété	Type	Description
<i>updateSeen</i>	boolean	Si Vrai, le message est marqué comme "seen" (lu) dans la boite de réception. Si Faux, le message n'est pas marqué comme "seen". Valeur par défaut : Vrai
<i>withBody</i>	boolean	Passez Vrai pour retourner le corps du message. Si Faux, seul l'en-tête du message est retourné. Valeur par défaut : Vrai

- La fonction génère une erreur et retourne Null si *msgID* désigne un message non existant,
- Si aucune boite de réception n'est sélectionnée avec la fonction `.selectBox()`, une erreur est générée,
- Si aucune connexion n'est ouverte, `.getMail()` ouvrira une connexion avec la dernière boite de réception spécifiée par `.selectBox()`.

Résultat

`.getMail()` retourne un `objet Email` avec les propriétés IMAP supplémentaires suivantes : *id*, *receivedAt* et *size*.

Exemple

Vous souhaitez lire le message avec ID = 1 :

```

var $server : Object
var $info; $mail; $boxInfo : Variant
var $transporter : 4D.IMAPTransporter

$server:=New object
$server.host:="imap.gmail.com" //obligatoire
$server.port:=993
$server.user:="4d@gmail.com"
$server.password:="XXXXXXX"

//creation du transporter
$transporter:=IMAP New transporter($server)

//selection de la mailbox
$boxInfo:=$transporter.selectBox("Inbox")

//récupération de l'objet Email d'ID 1
$mail:=$transporter.getMail(1)

```

.getMails()

► Historique

`.getMails(ids : Collection { ; options : Object }) : Object`
`.getMails(startMsg : Integer ; endMsg : Integer { ; options : Object }) : Object`

Paramètres	Type		Description
ids	Collection	->	Collection d'identifiants de messages
startMsg	Integer	->	Numéro de séquence du premier message
endMsg	Integer	->	Numéro de séquence du dernier message
options	Object	->	Instructions sur la gestion du message
Résultat	Object	<-	Objet contenant : <ul style="list-style-type: none"> une collection d'objets Email et une collection d'identifiants ou de numéros des messages manquants, le cas échéant

Description

La fonction `.getMails()` retourne un objet contenant une collection d'objets `Email`.

Première syntaxe :

`.getMails(ids { ; options }) -> result`

La première syntaxe vous permet de récupérer des messages en fonction de leurs identifiants.

Dans le paramètre `ids`, passez une collection d'identifiants des messages à retourner. Vous pouvez obtenir les ID à l'aide de [.getMail\(\)](#).

Le paramètre optionnel `options` vous permet de définir les parties de messages à retourner. Pour une description détaillée des propriétés disponibles, reportez-vous au tableau Options ci-dessous.

Deuxième syntaxe :

`.getMails(startMsg ; endMsg { ; options }) -> result`

La deuxième syntaxe vous permet de récupérer des messages en fonction d'une plage séquentielle. Les valeurs passées représentent la position des messages dans la boîte de réception.

Dans le paramètre `startMsg` passez une valeur *entier* correspondant au numéro du premier message dans une plage

séquentielle. Si vous passez un nombre négatif (`startMsg <= 0`), le premier message de la boîte de réception sera utilisé comme début de la séquence.

Dans le paramètre `endMsg` passez une valeur *entier long* correspondant au numéro du dernier message à inclure dans une plage séquentielle. Si vous passez un nombre négatif (`endMsg <= 0`), le dernier message de la boîte de réception sera utilisé comme fin de séquence.

Le paramètre optionnel `options` vous permet de définir les parties de messages à retourner.

Options

Propriété	Type	Description
<code>updateSeen</code>	Booléen	Si Vrai, les messages sont marqués comme "seen" (lus) dans la boîte de réception. Si Faux, les messages ne sont pas marqués comme "seen". Valeur par défaut : Vrai
<code>withBody</code>	Booléen	Passez Vrai pour retourner le corps des messages spécifiés. Si Faux, seuls les en-tête des messages sont retournés. Valeur par défaut : Vrai

- Si aucune boîte de réception n'est sélectionnée avec la fonction `.selectBox()`, une erreur est générée.
- S'il n'y a pas de connexion ouverte, `.getMails()` ouvrira une connexion avec la dernière boîte de réception spécifiée à l'aide de `.selectBox()`.

Résultat

`.getMails()` retourne un objet contenant les collections suivantes :

Propriété	Type	Description
<code>liste</code>	Collection	Collection d'objets <code>Email</code> . Si aucun objet Email n'est trouvé, une collection vide est retournée.
<code>notFound</code>	Collection	Collection de : <ul style="list-style-type: none">• première syntaxe - IDs de messages passés antérieurement qui n'existent pas• deuxième syntaxe - numéros de séquence des messages entre <code>startMsg</code> et <code>endMsg</code> qui n'existent pas Une collection vide est retournée si tous les messages sont trouvés.

Exemple

Vous souhaitez récupérer les 20 emails les plus récents sans modifier le statut "seen" :

```

var $server,$boxInfo,$result : Object
var $transporter : 4D.IMAPTransporter

$server:=New object
$server.host:="imap.gmail.com" //obligatoire
$server.port:=993
$server.user:="4d@gmail.com"
$server.password:="XXXXXXX"

//create transporter
$transporter:=IMAP New transporter($server)

//select mailbox
$boxInfo:=$transporter.selectBox("INBOX")

If($boxInfo.mailCount>0)
    // récupérer les en-têtes des 20 derniers messages sans les marquer comme lus
    $result:=$transporter.getMails($boxInfo.mailCount-20;$boxInfo.mailCount;\ 
        New object("withBody";False;"updateSeen";False))
    For each($mail;$result.list)
        //
    End for each
End if

```

.getMIMEAsBlob()

► Historique

.getMIMEAsBlob(*msgNumber* : Integer { ; *updateSeen* : Boolean }) : Blob

.getMIMEAsBlob(*msgID* : Text { ; *updateSeen* : Boolean }) : Blob

Paramètres	Type		Description
<i>msgNumber</i>	Integer	->	Numéro de séquence du message
<i>msgID</i>	Text	->	ID unique du message
<i>updateSeen</i>	Booléen	->	Si Vrai, le message est marqué comme "seen" (lu) dans la boîte de réception. Si Faux, le message demeure inchangé.
Résultat	BLOB	<-	Blob de la chaîne MIME retournée par le serveur mail

Description

La fonction `.getMIMEAsBlob()` retourne un BLOB avec le contenu MIME du message correspondant au *msgNumber* ou *msgID* dans la boîte de réception désignée par le `IMAP_transporter`.

Dans le premier paramètre, vous pouvez passer soit :

- *msgNumber*, une valeur *integer* indiquant le numéro de séquence du message à récupérer ou
- *msgID*, une valeur *text* indiquant l'ID unique du message à récupérer.

Le paramètre optionnel *updateSeen* vous permet d'indiquer si le message est marqué comme "seen" (lu) dans la boîte de réception. Vous pouvez passer :

- Vrai - pour marquer le message comme "seen" (indiquant que le message a été lu)
- Faux - pour ne pas modifier le statut "seen" du message

- La fonction retourne un BLOB vide si *msgNumber* ou *msgID* désigne un message inexistant,
- Si aucune boîte de réception n'est sélectionnée avec la fonction `.selectBox()`, une erreur est générée,
- S'il n'y a pas de connexion ouverte, `.getMIMEAsBlob()` ouvrira une connexion avec la dernière boîte de réception spécifiée à l'aide de `.selectBox()`.

Résultat

.getMIMEAsBlob() retourne un `BLOB` qui peut être archivé dans une base de données ou converti en un objet [Email](#) avec la commande `MAIL Convert from MIME`.

Exemple

```
var $server : Object
var $boxInfo : Variant
var $blob : Blob
var $transporter : 4D.IMAPTransporter

$server:=New object
$server.host:="imap.gmail.com"
$server.port:=993
$server.user:="4d@gmail.com"
$server.password:="XXXXXXXX"

//créer transporteur
$transporter:=IMAP New transporter($server)

//sélectionner la boite de réception
$boxInfo:=$transporter.selectBox("Inbox")

//obtenir BLOB
$blob:=$transporter.getMIMEAsBlob(1)
```

.host

► Historique

.host : Text

Description

La propriété `.host` contient le nom ou l'adresse IP du serveur hôte. Utilisée pour les échanges de courrier (SMTP, POP3, IMAP).

.logFile

► Historique

.logFile : Text

Description

La propriété `.logFile` contient le chemin complet du fichier d'historique qui a été défini (le cas échéant) pour la connexion. Elle peut être relative (au dossier Logs courant) ou absolue.

Contrairement aux fichiers log habituels (activés via la commande `SET DATABASE PARAMETER`), les fichiers log étendus stockent le contenu MIME de tous les emails envoyés et n'ont pas de limite de taille. Pour plus d'informations sur les fichiers log étendus, reportez-vous à :

- Connexions SMTP - [4DSMTPLLog.txt](#)
- Connexions POP3 - [4DPOP3Log.txt](#)
- Connexions IMAP - [4DIMAPLog.txt](#)

.move()

► Historique

.move(*msgsIDs* : Collection ; *destinationBox* : Text) : Object

.move(*allMsgs* : Integer ; *destinationBox* : Text) : Object

Paramètres	Type		Description
msgsIDs	Collection	->	Collection d'IDs uniques de messages (texte)
allMsgs	Integer	->	IMAP all : tous les messages de la boîte de réception sélectionnée
destinationBox	Text	->	Boîte de réception recevant les messages déplacés
Résultat	Object	<-	Statut de l'opération de déplacement

Description

La fonction `.move()` déplace les messages définis dans *msgsIDs* ou *allMsgs* vers la *destinationBox* sur le serveur IMAP.

Vous pouvez passer :

- dans le paramètre *msgsIDs*, une collection contenant les ID uniques des messages spécifiques à déplacer, ou
- dans le paramètre *allMsgs*, la constante `IMAP all` (integer) pour déplacer tous les messages de la boîte de réception sélectionnée.

Le paramètre *destinationBox* vous permet de passer une valeur texte avec le nom de la boîte de réception dans laquelle seront placés les messages déplacés.

Cette fonction est uniquement prise en charge par les serveurs IMAP compatibles avec RFC [8474](#).

Objet retourné

La fonction retourne un objet décrivant le statut IMAP :

Propriété		Type	Description
success		Booléen	Vrai si l'opération est réussie, sinon Faux
statusText		Text	Message du statut retourné par le serveur IMAP, ou dernière erreur retournée dans la pile d'erreurs 4D
errors		Collection	Pile d'erreurs 4D (non retournée si une réponse du serveur IMAP est reçue)
	[].errcode	Nombre	Code d'erreur 4D
	[].message	Text	Description de l'erreur 4D
	[].componentSignature	Text	Signature du composant interne qui a retourné l'erreur

Exemple 1

Pour déplacer une sélection de messages :

```

var $server;$boxInfo;$status : Object
var $mailIds : Collection
var $transporter : 4D.IMAPTransporter

$server:=New object
$server.host:="imap.gmail.com" //Obligatoire
$server.port:=993
$server.user:="4d@gmail.com"
$server.password:="XXXXXXX"

$transporter:=IMAP New transporter($server)

//sélectionner la boite de réception
$boxInfo:=$transporter.selectBox("inbox")

//obtenir la collection d'IDs uniques des messages
$mailIds:=$transporter.searchMails("subject \\"4D new feature:\\"")

// déplacer les messages de la boite de réception courante vers la boite de réception "documents"
$status:=$transporter.move($mailIds;"documents")

```

Exemple 2

Pour déplacer tous les messages de la boîte de réception courante :

```

var $server;$boxInfo;$status : Object
var $transporter : 4D.IMAPTransporter

$server:=New object
$server.host:="imap.gmail.com" //Obligatoire
$server.port:=993
$server.user:="4d@gmail.com"
$server.password:="XXXXXXX"

$transporter:=IMAP New transporter($server)

//sélectionner la boite de réception
$boxInfo:=$transporter.selectBox("inbox")

// déplacer tous les messages de la boite de réception courante vers la boite de réception "documents"
$status:=$transporter.move(IMAP all;"documents")

```

.numToID()

► Historique

.numToID(*startMsg* : Integer ; *endMsg* : Integer) : Collection

Paramètres	Type		Description
<i>startMsg</i>	Integer	->	Numéro de séquence du premier message
<i>endMsg</i>	Integer	->	Numéro de séquence du dernier message
Résultat	Collection	<-	Collection d'identifiants de messages uniques

Description

La fonction `.numToID()` convertit les numéros de séquence en IDs uniques IMAP pour les messages de la plage séquentielle désignée par *startMsg* et *endMsg* dans la boîte de réception courante.

Dans le paramètre `startMsg`, passez une valeur integer correspondant au numéro du premier message dans la plage séquentielle. Si vous passez un nombre négatif (`startMsg <= 0`), le premier message de la boîte de réception sera utilisé comme début de la séquence.

Dans le paramètre `endMsg`, passez une valeur integer correspondant au numéro du dernier message dans la plage séquentielle. Si vous passez un nombre négatif (`endMsg <= 0`), le dernier message de la boîte de réception sera utilisé comme fin de séquence.

Résultat

La fonction retourne une collection de chaînes (IDs uniques).

Exemple

```
var $transporter : 4D.IMAPTransporter
var $server;$boxInfo;$status : Object
var $mailIds : Collection

$server:=New object
$server.host:="imap.gmail.com" //Obligatoire
$server.port:=993
$server.user:="4d@gmail.com"
$server.password:="XXXXXXX"

$transporter:=IMAP New transporter($server)

//sélectionner la boite de réception
$boxInfo:=$transporter.selectBox("inbox")

//obtenir les IDs des 5 derniers messages reçus
$mailIds:=$transporter.numToID(($boxInfo.mailCount-5);$boxInfo.mailCount)

//supprimer les messages de la boite de réception courante
$status:=$transporter.delete($mailIds)
```

.removeFlags()

► Historique

`.removeFlags(msgIDs : Collection ; keywords : Object) : Object`

`.removeFlags(msgIDs : Text ; keywords : Object) : Object`

`.removeFlags(msgIDs : Longint ; keywords : Object) : Object`

Paramètres	Type		Description
msgIDs	Collection	->	Collection de chaînes : IDs uniques des messages (texte) Texte : ID unique d'un message Numérique (IMAP all) : Tous les messages de la boîte sélectionnée
keywords	Object	->	Mots-clés de flags à supprimer
Résultat	Object	<-	Statut de l'opération removeFlags

Description

La fonction `.removeFlags()` supprime des flags (drapeaux) des `msgIDs` pour les `keywords` définis.

Dans le paramètre `msgIDs`, vous pouvez passer soit :

- une *collection* contenant les IDs uniques de messages spécifiques, ou
- l'ID unique (*texte*) d'un seul message ou
- la constante suivante (*entier long*) pour tous les messages de la boîte sélectionnée :

Constante	Valeur	Commentaire
IMAP all	1	Sélectionner tous les messages de la boîte sélectionnée

Le paramètre `keywords` vous permet de passer un objet avec des valeurs de mots-clés pour les flags spécifiques à supprimer des `msgIDs`. Vous pouvez utiliser les mots-clés suivants :

Paramètres	Type	Description
<code>\$draft</code>	Booléen	True pour supprimer le marqueur "draft" du message
<code>\$seen</code>	Booléen	True pour supprimer le marqueur "seen" du message
<code>\$flagged</code>	Booléen	True pour supprimer le marqueur "flagged" du message
<code>\$answered</code>	Booléen	True pour supprimer le marqueur "answered" du message
<code>\$deleted</code>	Booléen	True pour supprimer le marqueur "deleted" du message

Les valeurs à faux sont ignorées.

Objet retourné

La fonction retourne un objet décrivant le statut IMAP :

Propriété		Type	Description
<code>success</code>		Booléen	Vrai si l'opération est réussie, sinon Faux
<code>statusText</code>		Text	Message du statut retourné par le serveur IMAP, ou dernière erreur retournée dans la pile d'erreurs 4D
<code>errors</code>		Collection	Pile d'erreurs 4D (non retournée si une réponse du serveur IMAP est reçue)
	<code>[] .errcode</code>	Nombre	Code d'erreur 4D
	<code>[] .message</code>	Text	Description de l'erreur 4D
	<code>[] .componentSignature</code>	Text	Signature du composant interne qui a retourné l'erreur

Exemple

```
var $options;$transporter;$boxInfo;$status : Object

$options:=New object
$options.host:="imap.gmail.com"
$options.port:=993
$options.user:="4d@gmail.com"
$options.password:="xxxxx"

// Créer le transporteur
$transporter:=IMAP New transporter($options)

// Sélectionner la boîte de réception
$boxInfo:=$transporter.selectBox("INBOX")

// Marquer tous les messages de la boîte de réception comme étant non vus
$flags:=New object
$flags["$seen"]:=True
$status:=$transporter.removeFlags(IMAP all;$flags)
```

.renameBox()

► Historique

.renameBox(*currentName* : Text ; *newName* : Text) : Object

Paramètres	Type		Description
currentName	Text	->	Nom actuel de la boîte de réception
newName	Text	->	Nouveau nom de la boîte de réception
Résultat	Object	<-	Statut de l'opération renaming

Description

La fonction `.renameBox()` renomme une mailbox sur le serveur IMAP. Essayer de renommer une mailbox qui n'existe pas ou de renommer une mailbox avec un nom qui est déjà utilisé génère une erreur.

Dans le paramètre `currentName`, passez le nom de la mailbox à renommer.

Passez le nouveau nom de la mailbox dans le paramètre `newName`.

Objet retourné

La fonction retourne un objet décrivant le statut IMAP :

Propriété		Type	Description
success		Booléen	Vrai si l'opération est réussie, sinon Faux
statusText		Text	Message du statut retourné par le serveur IMAP, ou dernière erreur retournée dans la pile d'erreurs 4D
errors		Collection	Pile d'erreurs 4D (non retournée si une réponse du serveur IMAP est reçue)
	[].errcode	Nombre	Code d'erreur 4D
	[].message	Text	Description de l'erreur 4D
	[].componentSignature	Text	Signature du composant interne qui a retourné l'erreur

Exemple

Pour renommer la mailbox "Invoices" en "Bills":

```
var $pw : text
var $options; $transporter; $status : object

$options:=New object

$pw:=Request("Please enter your password:")

If(OK=1) $options.host:="imap.gmail.com"
$options.user:="test@gmail.com"
$options.password:=$pw

$transporter:=IMAP New transporter($options)

// renommer la boite de réception
$status:=$transporter.renameBox("Invoices"; "Bills")

If ($status.success)
  ALERT("Mailbox renaming successful!")
Else
  ALERT("Error: "+$status.statusText)
End if
End if
```

.port

► Historique

.port : Integer

Description

La propriété `.port` contient le numéro de port utilisé pour les transactions d'emails. Par défaut, si la propriété `port` n'a pas été définie dans l'objet `server` (utilisé pour créer l'objet transporteur avec `SMTP New transporter`, `POP3 New transporter`, `IMAP New transporter`), le port utilisé est :

- SMTP - 587
- POP3 - 995
- IMAP - 993

.searchMails()

► Historique

.searchMails(*searchCriteria* : Text) : Collection

Paramètres	Type		Description
searchCriteria	Text	->	Critère(s) de recherche
Résultat	Collection	<-	Collection de numéros de messages

Description

Cette fonction est basée sur la spécification du [protocole IMAP](#).

La fonction `.searchMails()` recherche les messages qui correspondent aux critères de recherche `searchCriteria` dans la boîte de réception courante. Le paramètre `searchCriteria` contient un ou plusieurs mots-clés de recherche.

`searchCriteria` est un paramètre texte listant un ou plusieurs critères de recherche (voir [Mots-clés de recherche autorisés](#) ci-dessous) associés ou non à des valeurs à rechercher. Un critère de recherche peut être composé d'un ou plusieurs éléments. Par exemple :

```
SearchKey1 = FLAGGED  
SearchKey2 = NOT FLAGGED  
SearchKey3 = FLAGGED DRAFT
```

La correspondance n'est généralement pas sensible à la casse

- Si `searchCriteria` est une chaîne vide, la recherche sera équivalente à un "tout sélectionner".
- Si `searchCriteria` contient plusieurs mots-clés de recherche, le résultat est l'intersection (fonction ET) de tous les messages qui correspondent à ces critères.

```
searchCriteria = FLAGGED FROM "SMITH"
```

... retourne tous les messages comportant le marqueur `¥Flagged` ET envoyés par Smith.

- Vous pouvez utiliser les opérateurs OR ou NOT comme suit :

```
searchCriteria = OR SEEN FLAGGED
```

... retourne tous les messages comportant le marqueur `¥Seen` OU `¥Flagged`

```
searchCriteria = NOT SEEN
```

... retourne tous les messages ne comportant pas le marqueur \$Seen.

```
searchCriteria = HEADER CONTENT-TYPE "MIXED" NOT HEADER CONTENT-TYPE "TEXT" ...
```

... retourne les messages dont l'en-tête content-type contient "Mixed" et ne contient pas "Text".

```
searchCriteria = HEADER CONTENT-TYPE "E" NOT SUBJECT "o" NOT HEADER CONTENT-TYPE "MIXED"
```

... retourne les messages dont l'en-tête content-type contient " e " et dont l'en-tête Subject ne contient pas " o " et dont l'en-tête content-type n'est pas " Mixed ".

A noter que dans les deux derniers exemples, le résultat de la recherche est différent lorsque vous enlevez les parenthèses de la première liste de mots-clés.

- Le paramètre `searchCriteria` peut inclure optionnellement l'instruction [CHARSET]. Cette instruction est composée du mot "CHARSET" suivi d'un jeu de caractères défini [CHARSET] (US ASCII, ISO-8859). Elle indique le codage de caractères utilisé dans la chaîne `searchCriteria`. Par conséquent, vous devez convertir la chaîne `searchCriteria` dans le jeu de caractères spécifié si vous utilisez l'instruction [CHARSET] (voir les commandes `CONVERT FROM TEXT` ou `Convert to text`). Par défaut, 4D encode la chaîne de critères `searchCriteria` en "Quotable Printable" si elle contient des caractères étendus.

```
searchCriteria = CHARSET "ISO-8859" BODY "Help"
```

signifie que le critère de recherche utilise le jeu de caractères iso-8859 et que le serveur devra convertir la chaîne avant la recherche, si nécessaire.

Mots-clés de recherche autorisés

Les mots-clés de recherche peuvent traiter des valeurs des types suivants :

- Valeurs de type date : Les valeurs de type date sont placées dans des chaînes formatées de la manière suivante : `date-day+"-"+date-month+"-"+date-year` où date-day indique la date du jour dans le mois (2 caractères maxi), date-month indique le mois (Jan/Feb/Mar/Apr/May/Jun/Jul/Aug/Sep/Oct/Dec) et date-year indique l'année sur 4 chiffres. Exemple : `searchCriteria = SENTBEFORE 1-Feb-2020` (il n'est généralement pas nécessaire de mettre une date entre guillemets puisqu'elle ne contient pas de caractères spéciaux)
- **Valeurs de type chaîne ** : Les valeurs de type chaîne peuvent contenir tout type de caractère et doivent être placées entre des guillemets. Toutefois, si la chaîne ne contient pas de caractères spéciaux (des espaces par exemple), les guillemets ne sont pas obligatoires. Dans tous les cas, les guillemets vous permettent de vous assurer que la chaîne sera correctement interprétée. Exemple : `searchCriteria = FROM "SMITH"` Les recherches basées sur des chaînes de caractères sont du type "contient" : si le champ d'un message contient au moins la chaîne recherchée, le message est trouvé. La recherche ne tient pas compte de la casse des caractères.
- Noms de champs : Les valeurs de type nom de champ contiennent le nom d'un champ d'en-tête. Exemple : `searchCriteria = HEADER CONTENT-TYPE "MIXED"`
- Marqueurs : Les valeurs de type marqueur (flags) acceptent un ou plusieurs mots-clés (y compris des marqueurs standard) séparés par des espaces. Exemple : `searchCriteria = KEYWORD \Flagged \Draft`
- Ensemble de messages : Les valeurs de ce type désignent un ensemble de messages. Elles contiennent une liste de numéros de messages dans un ordre croissant, de 1 au nombre total de messages dans la boîte aux lettres. Les numéros sont séparés par des virgules ; un deux-points (:) indique un intervalle (inclusif) de numéros. Exemples : `2,4:7,9,12:*` représente les messages `2,4,5,6,7,9,12,13,14,15` pour une mailbox contenant 15 messages. `searchCriteria = 1:5 ANSWERED` recherche, parmi les messages 1 à 5, ceux qui comportent le marqueur \$Answered. `searchCriteria = 2,4 ANSWERED` recherche, parmi les messages 2 et 4, ceux qui comportent le marqueur \$Answered.

Mots-clés de recherche autorisés

ALL: Tous les messages de la boîte de réception.
ANSWERED: Messages comportant le marqueur \$Answered.
UNANSWERED: Messages ne comportant pas le marqueur \$Answered.
DELETED: Messages comportant le marqueur \$Deleted.
UNDELETED: Messages ne comportant pas le marqueur \$Deleted.
DRAFT: Messages comportant le marqueur \$Draft.
UNDRAFT: Messages ne comportant pas le marqueur \$Draft.
FLAGGED: Messages comportant le marqueur \$Flagged.
UNFLAGGED: Messages ne comportant pas le marqueur \$Flagged.
RECENT: Messages comportant le marqueur \$Recent.
OLD: Messages ne comportant pas le marqueur \$Recent.
SEEN: Messages comportant le marqueur \$Seen.
UNSEEN: Messages ne comportant pas le marqueur \$Seen.
NEW: Messages comportant le marqueur \$Recent et pas le marqueur \$Seen. Equivaut à "(RECENT UNSEEN)".
KEYWORD ***flag**: Messages comportant le marqueur spécifié.
UNKEYWORD ***flag**: Messages ne comportant pas le marqueur spécifié.
BEFORE ***date**: Messages dont la date interne est antérieure à la date spécifiée.
ON ***date**: Messages dont la date interne est égale à la date spécifiée.
SINCE ***date**: Messages dont la date interne est égale ou postérieure à la date spécifiée.
SENTBEFORE ***date**: Messages dont l'en-tête Date est antérieur à la date spécifiée.
SENTON ***date**: Messages dont l'en-tête Date est égal à la date spécifiée.
SENTSINCE ***date**: Messages dont l'en-tête Date est égal ou postérieur à la date spécifiée.
TO ***string**: Messages contenant la chaîne spécifiée dans l'en-tête Destinataire.
FROM ***string**: Messages contenant la chaîne spécifiée dans l'en-tête Emetteur.
CC ***string**: Messages contenant la chaîne spécifiée dans l'en-tête CC.
BCC ***string**: Messages contenant la chaîne spécifiée dans l'en-tête BCC.
SUBJECT ***string**: Messages contenant la chaîne spécifiée dans l'en-tête Objet.
BODY ***string**: Messages dont le corps contient la chaîne spécifiée.
TEXT ***string**: Messages contenant la chaîne spécifiée dans l'en-tête ou le corps.
HEADER *field-name* ***string**: Messages dont l'en-tête contient le champ défini et dont ce champ contient la chaîne définie.
UID ***message-UID**: Messages dont le numéro unique correspond à la valeur spécifiée.
LARGER ***n**: Messages dont la taille en octets est supérieure à la taille spécifiée.
SMALLER ***n**: Messages dont la taille en octets est inférieure à la taille spécifiée.
NOT ***search-key**: Messages ne correspondant pas au critère spécifié.
OR *search-key1* *search-key2******: Messages correspondant au premier ou au deuxième critère spécifié.

.selectBox()

► Historique

.selectBox(*name* : Text { ; *state* : Integer }) : Object

Paramètres	Type		Description
<i>name</i>	Text	->	Nom de la boîte de réception
<i>state</i>	Integer	->	Statut de l'accès à la boîte de réception
Résultat	Object	<-	Objet boxInfo

Description

La fonction `.selectBox()` sélectionne la mailbox *name* comme étant la mailbox courante. Cette fonction vous permet de récupérer des informations sur la boîte de réception.

Pour lire les informations à partir d'une mailbox sans changer la mailbox courante, utilisez `.getBoxInfo()`.

Dans le paramètre *name* passez le nom de la mailbox à laquelle vous souhaitez accéder. Le nom doit représenter une hiérarchie claire, de gauche à droite, avec des niveaux séparés par un caractère délimiteur spécifique. Le délimiteur peut

être récupéré à l'aide de la fonction `.getDelimiter()`.

Le paramètre optionnel `state` définit le type d'accès à la mailbox. Les valeurs possibles sont les suivantes :

Constante	Valeur	Commentaire
IMAP read only state	1	La mailbox sélectionnée est accessible avec les priviléges de lecture seule. Les messages contenant un marqueur "recent" (indication de nouveaux messages) restent inchangés.
IMAP read write state	0	La mailbox sélectionnée est accessible avec des priviléges de lecture et d'écriture. Les messages sont considérés "seen" et perdent le marqueur "recent" (indication des nouveaux messages). (Valeur par défaut)

- La fonction génère une erreur et retourne Null si `name` désigne une mailbox inexistante.
- Si aucune connexion n'est ouverte, `.selectBox()` ouvrira une connexion.
- Si la connexion n'a pas été utilisée depuis le délai de connexion (voir `IMAP New transporter`), la fonction `.checkConnection()` est automatiquement appelée.

Objet retourné

L'objet `boxInfo` contient les propriétés suivantes :

Propriété	Type	Description
<code>name</code>	Text	Nom de la boîte de réception
<code>mailCount</code>	number	Nombre de messages contenus dans la boîte de réception
<code>mailRecent</code>	number	Nombre de messages avec le marqueur "recent"

Exemple

```
var $server; $boxinfo : Object
$server:=New object
$server.host:="imap.gmail.com" //Obligatoire
$server.user:="4d@gmail.com"
$server.password:="XXXXXXXX"

var $transporter : 4D.IMAPTransporter
$transporter:=IMAP New transporter($server)
$boxInfo:=$transporter.selectBox("INBOX")
```

.subscribe()

► Historique

`.subscribe(name : Text) : Object`

Paramètres	Type		Description
<code>name</code>	Text	->	Nom de la boîte de réception
Résultat	Object	<-	Statut de l'opération subscribe

Description

La fonction `.subscribe()` permet d'ajouter la mailbox spécifiée à l'ensemble des mailboxes auxquelles vous avez "souscrit" sur le serveur IMAP. Ainsi, vous pouvez contrôler la taille de la liste des boîtes de réception disponibles en souscrivant uniquement à celles que vous souhaitez généralement consulter.

Dans le paramètre `name`, passez le nom de la mailbox à ajouter (subscribe) à la liste de mailboxes auxquelles vous avez "souscrit".

Objet retourné

La fonction retourne un objet décrivant le statut IMAP :

Propriété		Type	Description
success		Booléen	Vrai si l'opération est réussie, sinon Faux
statusText		Text	Message du statut retourné par le serveur IMAP, ou dernière erreur retournée dans la pile d'erreurs 4D
errors		Collection	Pile d'erreurs 4D (non retournée si une réponse du serveur IMAP est reçue)
	[].errcode	Nombre	Code d'erreur 4D
	[].message	Text	Description de l'erreur 4D
	[].componentSignature	Text	Signature du composant interne qui a retourné l'erreur

Exemple

Pour souscrire à la mailbox "Atlas Corp" dans la hiérarchie "Bills" :

```
var $pw; $name : text
var $options; $transporter; $status : object

$options:=New object

$pw:=Request("Please enter your password:")

If(OK=1) $options.host:="imap.gmail.com"
$options.user:="test@gmail.com"
$options.password:=$pw

$transporter:=IMAP New transporter($options)

$name:="Bills"+$transporter.getDelimiter()+"Atlas Corp"
$status:=$transporter.subscribe($name)

If ($status.success)
  ALERT("Mailbox subscription successful!")
Else
  ALERT("Error: "+$status.statusText)
End if
End if
Else
  ALERT("Error: "+$status.statusText)
End if
End if
```

.unsubscribe()

► Historique

.unsubscribe(name : Text) : Object

Paramètres	Type		Description
name	Text	->	Nom de la boîte de réception
Résultat	Object	<-	Statut de l'opération unsubscribe

Description

La fonction `.unsubscribe()` supprime la mailbox spécifiée de l'ensemble des mailboxes auxquelles vous avez "souscrit" sur le serveur IMAP. Elle vous permet de réduire le nombre de mailboxes que vous consultez régulièrement.

Dans le paramètre `name`, passez le nom de la mailbox à supprimer (`unsubscribe`) de la liste de mailboxes auxquelles vous avez "souscrit".

Objet retourné

La fonction retourne un objet décrivant le statut IMAP :

Propriété		Type	Description
success		Booléen	Vrai si l'opération est réussie, sinon Faux
statusText		Text	Message du statut retourné par le serveur IMAP, ou dernière erreur retournée dans la pile d'erreurs 4D
errors		Collection	Pile d'erreurs 4D (non retournée si une réponse du serveur IMAP est reçue)
	[].errcode	Nombre	Code d'erreur 4D
	[].message	Text	Description de l'erreur 4D
	[].componentSignature	Text	Signature du composant interne qui a retourné l'erreur

Exemple

Pour ne plus souscrire à la mailbox "Atlas Corp" dans la hiérarchie "Bills" :

```
var $pw; $name : text
var $options; $transporter; $status : object

$options:=New object

$pw:=Request("Please enter your password:")

If(OK=1) $options.host:="imap.gmail.com"
$options.user:="test@gmail.com"
$options.password:=$pw

$transporter:=IMAP New transporter($options)

$name:="Bills"+$transporter.getDelimiter()+"Atlas Corp"
$status:=$transporter.unsubscribe($name)

If ($status.success)
  ALERT("Mailbox unsubscription successful!")
Else
  ALERT("Error: "+$status.statusText)
End if
End if
Else
  ALERT("Error: "+$status.statusText)
End if
End if
```

.user

► Historique

.user : Text

Description

La propriété `.user` contient le nom d'utilisateur employé pour l'authentification sur le serveur mail.

MailAttachment

Les objets pièces jointes permettent de référencer des fichiers dans un objet [Email](#). Les objets Attachment (pièce jointe) sont créés à l'aide de la commande [MAIL New attachment](#).

Objet Attachment

Les objets Attachment fournissent les propriétés et fonctions suivantes en lecture seule :

.cid : Text	l'ID de la pièce jointe
.disposition : Text	la valeur de l'en-tête <code>Content-Disposition</code>
.getContent() : 4D.Blob	retourne le contenu de l'objet pièce jointe dans un objet <code>4D.Blob</code>
.name : Text	le nom et l'extension de la pièce jointe
.path : Text	le chemin POSIX complet de la pièce jointe, si elle existe
.platformPath : Text	le chemin complet de la pièce jointe exprimé avec la syntaxe de la plate-forme courante
.type : Texte	le <code>content-type</code> du fichier joint

MAIL New attachment

► Historique

```
MAIL New attachment( file : 4D.File { ; name : Text {; cid : Text{ ; type : Text { ; disposition :Text } } } } ) :  
4D.MailAttachment  
MAIL New attachment( zipFile : 4D.ZipFile { ; name : Text {; cid : Text{ ; type : Text { ; disposition :Text } } } } ) :  
4D.MailAttachment  
MAIL New attachment( blob : 4D.Blob { ; name : Text {; cid : Text{ ; type : Text { ; disposition :Text } } } } ) :  
4D.MailAttachment  
MAIL New attachment( path : Text { ; name : Text {; cid : Text{ ; type : Text { ; disposition :Text } } } } ) :  
4D.MailAttachment
```

Paramètres	Type		Description
file	4D.File	->	Fichier joint
zipFile	4D.ZipFile	->	Fichier Zip joint
blob	4D.Blob	->	Blob contenant la pièce jointe
path	Text	->	Chemin de la pièce jointe
name	Text	->	Nom + extension utilisés par le client de messagerie pour désigner la pièce jointe
cid	Text	->	ID de la pièce jointe (messages HTML uniquement) ou " " si aucun cid n'est requis
type	Text	->	Valeur de l'en-tête content-type
disposition	Text	->	Valeur de l'en-tête content-disposition : "inline" ou "attachment"
Résultat	4D.MailAttachment	<-	Objet pièce jointe

Description

La commande `MAIL New attachment` vous permet de créer un objet pièce jointe que vous pouvez associer à un objet [Email](#).

Pour définir l'objet attachment, vous pouvez utiliser :

- un `file`, en passant un objet `4D.File` contenant le fichier joint.
- un `zipfile`, en passant un objet `4D.ZipFile` contenant le fichier joint.
- un `blob`, en passant un objet `4D.Blob` contenant la pièce jointe elle-même.
- un `path`, en passant une valeur de type `text` contenant le chemin d'accès du fichier joint, exprimé avec la syntaxe du système. Vous pouvez passer un nom de chemin complet ou un simple nom de fichier (auquel cas 4D recherchera le fichier dans le même répertoire que le fichier du projet).

Le paramètre facultatif `name` vous permet de passer le nom et l'extension à utiliser par le client de messagerie pour désigner la pièce jointe. Si le paramètre `name` est omis et que :

- vous avez passé un chemin d'accès au fichier, le nom et l'extension du fichier sont utilisés,
- vous avez passé un BLOB, un nom aléatoire sans extension est automatiquement généré.

Le paramètre facultatif `cid` vous permet de passer un ID interne pour la pièce jointe. Cet ID est la valeur de l'en-tête `Content-Id` et sera utilisé dans les messages HTML uniquement. Le cid associe la pièce jointe à une référence définie dans le corps du message à l'aide d'une balise HTML telle que `\`. Cela signifie que le contenu de la pièce jointe (par exemple, une image) doit être affiché dans le message sur le client de messagerie. Le résultat final peut varier en fonction du client de messagerie. Vous pouvez passer une chaîne vide dans `cid` si vous ne souhaitez pas utiliser ce paramètre.

Vous pouvez utiliser le paramètre optionnel `type` pour définir explicitement le `content-type` du fichier joint. Par exemple, vous pouvez passer une chaîne définissant un type MIME ("video/mpeg"). Cette valeur de content-type sera définie pour la pièce jointe, quelle que soit son extension. Pour plus d'informations sur les types MIME, veuillez vous référer à [la page type MIME sur Wikipedia](#).

Par défaut, si le paramètre `type` est omis ou contient une chaîne vide, le `content-type` du fichier joint est basé sur son extension. Les règles suivantes sont appliquées pour les principaux types MIME :

Extension	Content Type
jpg, jpeg	image/jpeg
png	image/png
gif	image/gif
pdf	application/pdf
doc	application/msword
xls	application/vnd.ms-excel
ppt	application/vnd.ms-powerpoint
zip	application/zip
gz	application/gzip
json	application/json
js	application/javascript
ps	application/postscript
xml	application/xml
htm, html	text/html
mp3	audio/mpeg
other	application/octet-stream

Le paramètre facultatif *disposition* vous permet de passer l'en-tête `content-disposition` de la pièce jointe. Vous pouvez passer l'une des constantes suivantes du thème de constantes "Mail" :

Constante	Valeur	Commentaire
mail disposition attachment	"attachment"	Définissez la valeur de l'en-tête Content-disposition sur "attachment", ce qui signifie que le fichier joint doit être fourni sous forme de lien dans le message.
mail disposition inline	"inline"	Définissez la valeur de l'en-tête Content-disposition sur "inline", ce qui signifie que la pièce jointe doit être rendue dans le contenu du message, à l'emplacement du "cid". Le rendu dépend du client de messagerie.

Par défaut, si le paramètre *disposition* est omis :

- si le paramètre *cid* est utilisé, l'en-tête `Content-disposition` est défini sur "inline",
- si le paramètre *cid* n'est pas passé ou est vide, l'en-tête `Content-disposition` est fixé à "attachment".

Exemple 1

Vous souhaitez envoyer un e-mail avec un fichier sélectionné par l'utilisateur comme pièce jointe et une image intégrée dans le corps HTML :

```

$doc:=Select document("", "*"; "Please select a file to attach"; 0)
If (OK=1) //If a document was selected

C_OBJECT($email;$server;$transporter)

$server:=New object
$server.host:="smtp.mail.com"
$server.user:="test_user@mail.com"
$server.password:="p@ssw@rd"
$transporter:=SMTP New transporter($server)

$email:=New object
$email.from:="test_user@mail.com"
$email.to:="test_user@mail.com"
$email.subject:="This is a test message with attachments"

//add a link to download file
$email.attachments:=New collection(MAIL New attachment(Document))
//insert an inline picture (use a cid)
$email.attachments[1]:=MAIL New attachment("c:\\\\Pictures\\\\4D.jpg";"";"4D")

$email.htmlBody:="<html>" + \
"<body>Hello World!" + \
"<img src='cid:4D' >" + \
"</body>" + \
"</head>" + \
"</html>"

$transporter.send($email) //envoyer e-mail

```

End if

Exemple 2

Vous voulez envoyer un e-mail avec une zone 4D Write Pro en pièce jointe :

```

C_BLOB($blob)
WP EXPORT VARIABLE(WPArea;$blob;wk docx)

C_OBJECT($email;$server;$transporter)

$server:=New object
$server.host:="smtp.mail.com"
$server.user:="user@mail.com"
$server.password:="p@ssw@rd"
$transporter:=SMTP New transporter($server)

$email:=New object
$email.from:="user@mail.com"
$email.to:="customer@mail.com"
$email.subject:="New annual report"
$email.textBody:="Please find enclosed our latest annual report."
$email.attachments:=New collection(MAIL New attachment($blob;"Annual report.docx"))

$transporter.send($email)

```

4D.MailAttachment.new()

► Historique

4D.MailAttachment.new(*file* : 4D.File { ; *name* : Text { ; *cid* : Text{ ; *type* : Text { ; *disposition* :Text } } } }) :

```

4D.MailAttachment
4D.MailAttachment.new( zipFile : 4D.ZipFile { ; name : Text {; cid : Text{ ; type : Text { ; disposition :Text } } } } ) :
4D.MailAttachment
4D.MailAttachment.new( blob : 4D.Blob { ; name : Text {; cid : Text{ ; type : Text { ; disposition :Text } } } } ) :
4D.MailAttachment
4D.MailAttachment.new( path : Text { ; name : Text {; cid : Text{ ; type : Text { ; disposition :Text } } } } ) :
4D.MailAttachment

```

Paramètres	Type		Description
file	4D.File	->	Fichier joint
zipFile	4D.ZipFile	->	Fichier Zip joint
blob	4D.Blob	->	Blob contenant la pièce jointe
path	Text	->	Chemin de la pièce jointe
name	Text	->	Nom + extension utilisés par le client de messagerie pour désigner la pièce jointe
cid	Text	->	ID de la pièce jointe (messages HTML uniquement) ou " " si aucun cid n'est requis
type	Text	->	Valeur de l'en-tête content-type
disposition	Text	->	Valeur de l'en-tête content-disposition : "inline" ou "attachment"
Résultat	4D.MailAttachment	<-	Objet pièce jointe

Description

La fonction `4D.MailAttachment.new()` crée et retourne un nouvel objet de type `4D.MailAttachment`. Elle est identique à la commande `MAIL New attachment` (raccourci).

.cid

`.cid` : Text

Description

La propriété `.cid` contient l'ID de la pièce jointe. Cette propriété est utilisée uniquement dans les messages HTML. Si cette propriété est manquante, le fichier est géré comme une simple pièce jointe (lien).

.disposition

`.disposition` : Text

Description

La propriété `.disposition` contient la valeur de l'en-tête `Content-Disposition`. Deux valeurs sont disponibles :

- "inline" : la pièce jointe est rendue dans le contenu du message, à l'emplacement "cid". Le rendu dépend du client de messagerie.
- "attachment" : la pièce jointe est fournie sous forme de lien dans le message.

.getContent()

`.getContent()` : 4D.Blob

Paramètres	Type		Description
Résultat	4D.Blob	<-	Contenu de la pièce jointe

Description

La fonction `.getContent()` retourne le contenu de l'objet pièce jointe dans un objet `4D.Blob`. Vous pouvez utiliser cette fonction avec les objets pièce jointe reçus par la commande `MAIL Convert from MIME`.

.name

`.name` : Text

Description

La propriété `.name` contient le nom et l'extension de la pièce jointe. Par défaut, c'est le nom du fichier, sauf si un autre nom a été indiqué dans la commande `MAIL New attachment`.

.path

`.path` : Text

Description

La propriété `.path` contient le chemin POSIX complet de la pièce jointe, si elle existe.

.platformPath

► Historique

`.platformPath` : Text

Description

La propriété `.platformPath` contient le chemin complet de la pièce jointe exprimé avec la syntaxe de la plate-forme courante.

.type

`.type` : Texte

Description

La propriété `.type` contient le `content-type` du fichier joint. Si ce type n'est pas explicitement passé à la commande `MAIL New attachment`, le `content-type` est fondé sur son extension de fichier.

POP3Transporter

La classe `POP3Transporter` vous permet de récupérer des messages à partir d'un serveur de messagerie POP3.

Objet POP3 Transporter

Les objets Transporter POP3 sont instanciés avec la commande [POP3 New transporter](#). Leurs propriétés et fonctions sont les suivantes :

`.acceptUnsecureConnection : Boolean`

True si 4D est autorisé à établir une connexion non chiffrée

`.authenticationMode : Text`

le mode d'authentification utilisé pour ouvrir la session sur le serveur de mail

`.checkConnection() : Object`

vérifie la connexion à l'aide des informations stockées dans l'objet transporteur

`.connectionTimeOut : Integer`

la durée maximale (en secondes) autorisée avant vérification de la connexion au serveur

`.delete(msgNumber : Integer)`

marque l'email `msgNumber` comme devant être supprimé sur le serveur POP3

`.getBoxInfo() : Object`

`.getMail(msgNumber : Integer) : Object`

retourne l'objet `Email` correspondant au `msgNumber` de la boîte de réception désignée par le `POP3 transporter`

`.getMailInfo(msgNumber : Integer) : Object`

`.getMailInfoList() : Collection`

retourne une collection d'objets `mailInfo` décrivant les messages de la boîte de réception désignés par le `POP3 transporter`

`.getMIMEAsBlob(msgNumber : Integer) : Blob`

retourne un BLOB avec le contenu MIME du message correspondant au `msgNumber` dans la boîte de réception désignée par le `POP3 transporter`

`.host : Text`

contient le nom ou l'adresse IP du serveur hôte

`.logFile : Text`

le chemin complet du fichier d'historique qui a été défini (le cas échéant) pour la connexion

`.port : Integer`

le numéro de port utilisé pour les transactions d'emails

`.undeleteAll()`

supprime tous les marqueurs de suppression définis pour les emails de `POP3 transporter`

`.user : Text`

le nom d'utilisateur employé pour l'authentification sur le serveur mail

POP3 New transporter

► Historique

POP3 New transporter(`server` : Object) : 4D.POP3Transporter

Paramètres	Type		Description
server	object	->	Informations sur le serveur IMAP
Résultat	4D.POP3Transporter	<-	Objet POP3 transporter

Description

La commande `POP3 New transporter` configure une nouvelle connexion POP3 en fonction du paramètre `server` et retourne un nouvel objet `POP3 transporter`. L'objet transporteur retourné sera alors utilisé pour la réception d'emails.

Dans le paramètre `server`, passez un objet contenant les propriétés suivantes :

server	Valeur par défaut (si omise)
<code>.acceptUnsecureConnection : Boolean</code> True si 4D est autorisé à établir une connexion non chiffrée	Faux
<code>.accessTokenOAuth2 : Text</code> <code>.accessTokenOAuth2 : Object</code> Chaîne de texte ou objet token représentant les informations d'autorisation OAuth2. Il est utilisé uniquement avec <code>OAUTH2 authenticationMode</code> . Si <code>accessTokenOAuth2</code> est utilisé mais que <code>authenticationMode</code> est omis, le protocole OAuth 2 est utilisé (si le serveur l'autorise). Non retourné dans l'objet <code>SMTP transporter</code> .	aucune
<code>.authenticationMode : Text</code> le mode d'authentification utilisé pour ouvrir la session sur le serveur de mail	le mode d'authentification le plus sûr pris en charge par le serveur est utilisé
<code>.connectionTimeOut : Integer</code> la durée maximale (en secondes) autorisée avant vérification de la connexion au serveur	30
<code>.host : Text</code> contient le nom ou l'adresse IP du serveur hôte	obligatoire
<code>.logFile : Text</code> le chemin complet du fichier d'historique qui a été défini (le cas échéant) pour la connexion	aucune
<code>.password : Text</code> Mot de passe de l'utilisateur pour l'authentification sur le serveur. Non retourné dans l'objet <code>SMTP transporter</code> .	aucune
<code>.port : Integer</code> le numéro de port utilisé pour les transactions d'emails	995
<code>.user : Text</code> le nom d'utilisateur employé pour l'authentification sur le serveur mail	aucune

Résultat

La fonction retourne un `objet POP3 transporter`. Toutes les propriétés retournées sont en lecture seule.

La connexion POP3 est automatiquement fermée lorsque l'objet transporteur est détruit.

Exemple

```

var $server : Object
$server:=New object
$server.host:="pop.gmail.com" //Obligatoire
$server.port:=995
$server.user:="4d@gmail.com"
$server.password:="XXXXXXXX"
$server.logFile:="LogTest.txt" //log à enregistrer dans le dossier Logs

var $transporter : 4D.POP3Transporter
$transporter:=POP3 New transporter($server)

$status:=$transporter.checkConnection()
If(Not($status.success))
    ALERT("An error occurred receiving the mail: "+$status.statusText)
End if

```

4D.POP3Transporter.new()

4D.POP3Transporter.new(*server* : Object) : 4D.POP3Transporter

Paramètres	Type		Description
<i>server</i>	Object	->	Informations sur le serveur IMAP
Résultat	4D.POP3Transporter	<-	Objet POP3 transporter

Description

La fonction `4D.POP3Transporter.new()` crée et retourne un nouvel objet de type `4D.POP3Transporter`. Elle est identique à la commande [POP3 New transporter](#) (raccourci).

.acceptUnsecureConnection

► Historique

`.acceptUnsecureConnection` : Boolean

Description

La propriété `.acceptUnsecureConnection` contient True si 4D est autorisé à établir une connexion non chiffrée lorsqu'une connexion chiffrée n'est pas possible.

Elle contient False si les connexions non chiffrées ne sont pas autorisées, auquel cas une erreur est renvoyée lorsque la connexion chiffrée n'est pas possible.

Ports sécurisés disponibles :

- SMTP
 - 465: SMTPS
 - 587 ou 25 : SMTP avec mise à niveau STARTTLS si le serveur le prend en charge.
- IMAP
 - 143 : Port IMAP non chiffré
 - 993 : IMAP avec mise à niveau STARTTLS si le serveur le prend en charge
- POP3
 - 110 : Port POP3 non chiffré
 - 995 : POP3 avec mise à niveau STARTTLS si le serveur le prend en charge.

.authenticationMode

► Historique

.authenticationMode : Text

Description

La propriété `.authenticationMode` contient le mode d'authentification utilisé pour ouvrir la session sur le serveur de messagerie.

Par défaut, le mode le plus sécurisé pris en charge par le serveur est utilisé.

Valeurs possibles :

Valeur	Constantes	Commentaire
APOP	<code>POP3 authentication APOP</code>	Authentification à l'aide du protocole APOP (POP3 uniquement)
CRAM-MD5	<code>POP3 authentication CRAM-MD5</code>	Authentification à l'aide du protocole CRAM-MD5
LOGIN	<code>POP3 authentication login</code>	Authentification à l'aide du protocole LOGIN
OAuth2	<code>POP3 authentication OAuth2</code>	Authentification à l'aide du protocole OAuth2
PLAIN	<code>POP3 authentication plain</code>	Authentification à l'aide du protocole PLAIN

.checkConnection()

► Historique

.checkConnection() : Object

Paramètres	Type		Description
Résultat	Object	<-	Statut de la connexion de l'objet transporteur

Description

La fonction `.checkConnection()` vérifie la connexion à l'aide des informations stockées dans l'objet transporteur, recréée la connexion si nécessaire et retourne son statut. Cette fonction vous permet de vérifier que les valeurs fournies par l'utilisateur sont valides et cohérentes.

Objet retourné

La fonction envoie une requête au serveur de mail et retourne un objet décrivant le statut. Cet objet peut avoir les propriétés suivantes :

Propriété	Type	Description
success	boolean	Vrai si la vérification a été effectuée avec succès, sinon Faux
status	number	(SMTP uniquement) Code du statut retourné par le serveur de messagerie (0 en cas de problème non lié au traitement du mail)
statusText	text	Message du statut retourné par le serveur de messagerie, ou dernière erreur retournée dans la pile d'erreurs 4D
errors	collection	Pile d'erreurs 4D (non retournée si une réponse du serveur de messagerie est reçue)
[].errCode	number	Code d'erreur 4D
[].message	text	Description de l'erreur 4D
[].componentSignature	text	Signature du composant interne qui a retourné l'erreur

Exemple

```
var $pw : Text
var $options : Object
$options:=New object

$pw:=Request("Please enter your password:")
if(OK=1)
    $options.host:="pop3.gmail.com"

$options.user:="test@gmail.com"
$options.password:=$pw

$transporter:=POP3 New transporter($options)

$status:=$transporter.checkConnection()
If($status.success)
    ALERT("POP3 connection check successful!")
Else
    ALERT("Error: "+$status.statusText)
End if
End if
```

.connectionTimeOut

► Historique

.connectionTimeOut : Integer

Description

La propriété `.connectionTimeOut` contient la durée maximale (en secondes) autorisée avant vérification de la connexion au serveur. Par défaut, si la propriété n'a pas été définie dans l'objet `server` (utilisé pour créer l'objet transporteur avec `SMTP New transporter`, `POP3 New transporter`, `IMAP New transporter`), la valeur utilisée est 30.

.delete()

► Historique

.delete(*msgNumber* : Integer)

Paramètres	Type		Description
<code>msgNumber</code>	Integer	->	Numéro du message à supprimer

Description

La fonction `.delete()` marque l'email *msgNumber* comme devant être supprimé sur le serveur POP3.

Dans le paramètre *msgNumber*, passez le numéro de l'email à supprimer. Ce numéro est retourné dans la propriété `number` par la fonction `.getMailInfoList()`.

L'exécution de cette méthode ne supprime pas réellement l'email. L'email marqué sera supprimé sur le serveur POP uniquement lorsque l'objet `POP3_transporter` (créé à l'aide de `POP3 New transporter`) sera détruit. Le marqueur pourra également être retiré via la fonction `.undeleteAll()`.

Si la session courante se termine inopinément et que la connexion est fermée (ex : timeout, panne de réseau, etc.), un message d'erreur est généré et les messages marqués pour suppression demeureront sur le serveur POP3.

Exemple

```

$mailInfoList:=$POP3_transporter.getMailInfoList()
For each($mailInfo;$mailInfoList)
    // Marquer votre e-mail comme "à supprimer à la fin de la session"
    $POP3_transporter.delete($mailInfo.number)
End for each
    // Forcer la fermeture de la session pour supprimer les e-mails marqués pour suppression
CONFIRM("Selected messages will be deleted.;";"Delete";"Undo")
If(OK=1) //suppression confirmée
    $POP3_transporter:=Null
Else
    $POP3_transporter.undeleteAll() //supprimer les marqueurs de suppression
End if

```

.getBoxInfo()

► Historique

.getBoxInfo() : Object

Paramètres	Type		Description
Résultat	Object	<-	Objet boxInfo

Description

La fonction `.getBoxInfo()` retourne un objet `boxInfo` correspondant à la boîte de réception désignée par le [POP3 transporter](#). Cette fonction vous permet de récupérer des informations sur la boîte de réception.

L'objet `boxInfo` contient les propriétés suivantes :

Propriété	Type	Description
mailCount	Nombre	Nombre de messages contenus dans la boîte de réception
size	Nombre	Taille du message en octets

Exemple

```

var $server; $boxinfo : Object

$server:=New object
$server.host:="pop.gmail.com" //Obligatoire
$server.port:=995
$server.user:="4d@gmail.com"
$server.password:="XXXXXXXX"

$transporter:=POP3 New transporter($server)

//mailbox info
$boxInfo:=$transporter.getBoxInfo()
ALERT("The mailbox contains "+String($boxInfo.mailCount)+" messages.")

```

.getMail()

► Historique

.getMail(*msgNumber* : Integer) : Object

Paramètres	Type		Description
msgNumber	Integer	->	Numéro du message dans la liste
Résultat	Object	<-	Objet email

Description

La fonction `.getMail()` retourne l'objet `Email` correspondant au `msgNumber` de la boîte de réception désignée par le `POP3 transporter`. Cette fonction vous permet de gérer localement le contenu de l'email.

Passez dans `msgNumber` le numéro du message à récupérer. Ce numéro est retourné dans la propriété `number` par la fonction `.getMailInfoList()`.

La fonction retourne Null si :

- `msgNumber` désigne un message inexistant,
- le message a été marqué pour suppression à l'aide de `.delete()`.

Objet retourné

`.getMail()` retourne un objet `Email`.

Exemple

Vous souhaitez connaître l'expéditeur du premier mail de la boîte de réception :

```
var $server; $transporter : Object
var $mailInfo : Collection
var $sender : Variant

$server:=New object
$server.host:="pop.gmail.com" //Obligatoire
$server.port:=995
$server.user:="4d@gmail.com"
$server.password:="XXXXXXXX"

$transporter:=POP3 New transporter($server)

$mailInfo:=$transporter.getMailInfoList()
$sender:=$transporter.getMail($mailInfo[0].number).from
```

.getMailInfo()

► Historique

`getMailInfo(msgNumber : Integer) : Object`

Paramètres	Type		Description
msgNumber	Integer	->	Numéro du message dans la liste
Résultat	Object	<-	Objet MailInfo

Description

La fonction `.getMailInfo()` retourne un objet `mailInfo` correspondant au `msgNumber` de la boîte de réception désignée par le `POP3 transporter`. Cette fonction vous permet de récupérer des informations sur l'email.

Dans le paramètre `msgNumber`, passez le numéro de l'email à récupérer. Ce numéro est retourné dans la propriété `number` par la fonction `.getMailInfoList()`.

L'objet `mailInfo` retourné contient les propriétés suivantes :

Propriété	Type	Description
size	Nombre	Taille du message en octets
id	Text	ID unique du message

La fonction retourne Null si :

- *msgNumber* désigne un message inexistant,
- le message a été marqué pour suppression à l'aide de `.delete()`.

Exemple

```
var $server; $mailInfo : Object
var $mailNumber : Integer

$server.host:="pop.gmail.com" //Obligatoire
$server.port:=995
$server.user:="4d@gmail.com"
$server.password:="XXXXXXXX"

var $transporter : 4D.POP3Transporter
$transporter:=POP3 New transporter($server)

//message info
$mailInfo:=$transporter.getMailInfo(1) //obtenir le premier e-mail
If($mailInfo #Null)
  ALERT("First mail size is:"+String($mailInfo.size)+" bytes.")
End if
```

.getMailInfoList()

► Historique

.getMailInfoList() : Collection

Paramètres	Type		Description
Résultat	Collection	<-	Collection d'objets <code>mailInfo</code>

Description

La fonction `.getMailInfoList()` retourne une collection d'objets `mailInfo` décrivant les messages de la boîte de réception désignés par le `POP3 transporter`. Cette fonction vous permet de gérer localement les listes de messages situés sur le serveur de messagerie POP3.

Chaque objet `mailInfo` retourné contient les propriété suivantes :

Propriété	Type	Description
[].size	Nombre	Taille du message en octets
[].number	Nombre	Numéro du message
[].id	Text	ID unique du message (utile si vous stockez le message localement)

Si la boîte de réception ne contient pas de message, une collection vide est retournée.

Propriétés number et ID

number est le numéro d'un message de la boîte de réception au moment de la création du `POP3_transporter`. La propriété *number* n'est pas une valeur statique liée à un message spécifique. Elle change d'une session à l'autre en fonction de sa relation à d'autres messages de la boîte de réception au moment de l'ouverture de la session. Les

numéros assignés aux messages sont valides uniquement pendant la durée de vie du `POP3_transporter`. Lorsque le `POP3_transporter` est supprimé, les messages marqués pour suppression seront supprimés. Lorsque l'utilisateur se reconnecte au serveur, les messages courants de la boîte de réception sont renumérotés de 1 à x.

A l'inverse, *id* est un numéro unique assigné à un message lorsqu'il a été reçu par le serveur. Ce numéro est calculé à l'aide de l'heure et de la date de réception du message et sa valeur est assignée par votre serveur POP3. Malheureusement, les serveurs POP3 n'utilisent pas l'*id* comme référence primaire à leurs messages. Durant les sessions POP3, vous aurez besoin d'indiquer le *number* comme référence aux messages du serveur. Les développeurs doivent être vigilants s'ils développent des solutions qui importent des références aux messages dans une base de données tout en laissant le corps du message sur le serveur.

Exemple

Vous souhaitez connaitre le nombre total d'emails de la boîte de réception ainsi que leur taille :

```
var $server : Object
$server:=New object
$server.host:="pop.gmail.com" //Obligatoire
$server.port:=995
$server.user:="4d@gmail.com"
$server.password:="XXXXXXXX"

var $transporter : 4D.POP3Transporter
$transporter:=POP3 New transporter($server)

C_COLLECTION($mailInfo)
C_LONGINT($vNum;$vSize)

$mailInfo:=$transporter.getMailInfoList()
$vNum:=$mailInfo.length
$vSize:=$mailInfo.sum("size")

ALERT("The mailbox contains "+String($vNum)+" message(s) for "+String($vSize)+" bytes.")
```

.getMIMEAsBlob()

► Historique

`.getMIMEAsBlob(msgNumber : Integer) : Blob`

Paramètres	Type		Description
msgNumber	Integer	->	Numéro du message dans la liste
Résultat	Blob	<-	Blob de la chaîne MIME renournée par le serveur mail

Description

La fonction `.getMIMEAsBlob()` retourne un BLOB avec le contenu MIME du message correspondant au *msgNumber* dans la boîte de réception désignée par le `POP3_transporter`.

Dans le paramètre *msgNumber*, passez le numéro de l'email à récupérer. Ce numéro est retourné dans la propriété *number* par la fonction `.getMailInfoList()`.

La fonction retourne un Blob vide si :

- *msgNumber* désigne un message inexistant,
- le message a été marqué pour suppression à l'aide de `.delete()`.

BLOB retourné

`.getMIMEAsBlob()` retourne un `BLOB` qui peut être archivé dans une base de données ou converti en un objet `Email` avec la commande `MAIL Convert from MIME`.

Exemple

Vous souhaitez connaitre le nombre total d'emails de la boîte de réception ainsi que leur taille :

```
var $server : Object
var $mailInfo : Collection
var $blob : Blob
var $transporter : 4D.POP3Transporter

$server:=New object
$server.host:="pop.gmail.com"
$server.port:=995
$server.user:="4d@gmail.com"
$server.password:="XXXXXXXX"

$transporter:=POP3 New transporter($server)

$mailInfo:=$transporter.getMailInfoList()
$blob:=$transporter.getMIMEAsBlob($mailInfo[0].number)
```

.host

► Historique

.host : Text

Description

La propriété `.host` contient le nom ou l'adresse IP du serveur hôte. Utilisée pour les échanges de courrier (SMTP, POP3, IMAP).

.logFile

► Historique

.logFile : Text

Description

La propriété `.logFile` contient le chemin complet du fichier d'historique qui a été défini (le cas échéant) pour la connexion. Elle peut être relative (au dossier Logs courant) ou absolue.

Contrairement aux fichiers log habituels (activés via la commande `SET DATABASE PARAMETER`), les fichiers log étendus stockent le contenu MIME de tous les emails envoyés et n'ont pas de limite de taille. Pour plus d'informations sur les fichiers log étendus, reportez-vous à :

- Connexions SMTP - [4DSMTPLLog.txt](#)
- Connexions POP3 - [4DPOP3Log.txt](#)
- Connexions IMAP - [4DIMAPLog.txt](#)

.port

► Historique

.port : Integer

Description

La propriété `.port` contient le numéro de port utilisé pour les transactions d'emails. Par défaut, si la propriété `port` n'a pas été définie dans l'objet `server` (utilisé pour créer l'objet transporteur avec `SMTP New transporter`, `POP3 New transporter`, `IMAP New transporter`), le port utilisé est :

- SMTP - 587

- POP3 - 995
- IMAP - 993

.undeleteAll()

► Historique

.undeleteAll() | Paramètres | Type | Description | | ----- | --- |::| ----- | | | | Ne requiert aucun paramètre |

Description

La fonction `.undeleteAll()` supprime tous les marqueurs de suppression définis pour les emails de [POP3_transporter](#).

.user

► Historique

.user : Text

Description

La propriété `.user` contient le nom d'utilisateur employé pour l'authentification sur le serveur mail.

Session

Les objets Session sont retournés par la commande `Session` lorsque [les sessions extensibles sont activées dans votre projet](#). L'objet Session est automatiquement créé et maintenu par le serveur web 4D pour contrôler la session d'un client web (par exemple, un navigateur). Cet objet fournit au développeur web une interface avec la session de l'utilisateur, permettant de gérer les priviléges, de stocker des données contextuelles, de partager des informations entre les process et de lancer des processus préemptifs liés à la session.

Pour des informations détaillées sur l'implémentation de la session, veuillez consulter la section [Sessions du serveur web](#).

Sommaire

<code>.clearPrivileges()</code>	supprime tous les priviléges associés à la session
<code>.expirationDate : Text</code>	la date et l'heure d'expiration du cookie de session
<code>.hasPrivilege(privilege : Text) : Boolean</code>	retourne Vrai si le privilège est associé à la session, et Faux sinon
<code>.idleTimeout : Integer</code>	le délai maximal d'inactivité de session (en minutes), au-delà duquel la session est automatiquement fermée par 4D
<code>.isGuest() : Boolean</code>	retourne Vrai si la session est une session Guest (i.e. elle n'a pas de priviléges)
<code>.setPrivileges(privilege : Text)</code> <code>.setPrivileges(privileges : Collection)</code> <code>.setPrivileges(settings : Object)</code>	associe le ou les privilège(s) défini(s) en paramètre à la session
<code>.storage : Object</code>	un objet partagé qui peut être utilisé pour stocker des informations disponibles pour toutes les requêtes du client web
<code>.userName : Text</code>	le nom d'utilisateur associé à la session

Session

► Historique

Session : 4D.Session

Paramètres	Type		Description
Résultat	4D.Session	<-	Objet session

Description

La commande `Session` retourne l'objet `Session` correspondant à la session web utilisateur extensible courante.

Cette commande fonctionne uniquement lorsque les [sessions extensibles sont activées](#). Elle retourne `Null` lorsque ces

sessions sont inactives ou lorsque les anciennes sessions sont utilisées.

Lorsque les sessions extensibles sont activées, l'objet `Session` est disponible depuis n'importe quel process web dans les contextes suivants :

- Méthodes base `On Web Authentication`, `On Web Connection`, et `On REST Authentication`,
- Les fonctions Data Model Class ORDA appelées par des requêtes REST,
- le code traité par les balises 4D dans les pages semi-dynamiques (4DTEXT, 4DHTML, 4DEVAL, 4DSRIPT/, 4DCODE)
- les méthodes projet avec l'attribut "Available through 4D tags and URLs (4DACTION...)" et appelées via les urls 4DACTION/.

Exemple

Vous avez défini la méthode `action_Session` ayant l'attribut "Disponible via Balises HTML et URLs 4D". Vous appelez la méthode en saisissant l'URL suivant dans votre navigateur :

IP:port/4DACTION/action_Session

```
//méthode action_Session
Case of
:(Session#Null)
  If(Session.hasPrivilege("WebAdmin")) //appel de la fonction hasPrivilege
    WEB SEND TEXT("4DACTION --> Session is WebAdmin")
  Else
    WEB SEND TEXT("4DACTION --> Session is not WebAdmin")
  End if
Else
  WEB SEND TEXT("4DACTION --> Session is null")
End case
```

.clearPrivileges()

► Historique

.clearPrivileges() | Paramètres | Type | | Description | | ----- | ---- |::| ----- | | | | Ne requiert aucun paramètre |

Description

La fonction `.clearPrivileges()` supprime tous les privilèges associés à la session. En résultat, la session devient automatiquement une session Guest.

Exemple

```
//Invalider une session
var $isGuest : Boolean

Session.clearPrivileges()
$isGuest:=Session.isGuest() //isGuest est mis à True
```

.expirationDate

► Historique

.expirationDate : Text

Description

La propriété `.expirationDate` contient la date et l'heure d'expiration du cookie de session. La valeur est exprimée sous forme de texte au format ISO 8601 : `YYYY-MM-DDTHH:MM:SS.mmmZ`.

Cette propriété est en lecture seule. Elle est automatiquement recalculée si la valeur de la propriété `.idleTimeout` est modifiée.

Exemple

```
var $expiration : Text  
$expiration:=Session.expirationDate //ex : "2021-11-05T17:10:42Z"
```

.hasPrivilege()

► Historique

`.hasPrivilege(privilege : Text) : Boolean`

Paramètres	Type		Description
privilege	Text	<-	Nom du privilège à vérifier
Résultat	Booléen	<-	Vrai si la session dispose du <i>privilege</i> , sinon Faux

Description

La fonction `.hasPrivilege()` retourne Vrai si le privilège est associé à la session, et Faux sinon.

Exemple

Vous voulez vérifier si le privilège "WebAdmin" est associé à la session :

```
If (Session.hasPrivilege("WebAdmin"))  
    //Accès accordé, ne rien faire  
Else  
    //Afficher une page d'authentification  
  
End if
```

.idleTimeout

► Historique

`.idleTimeout : Integer`

Description

La propriété `.idleTimeout` contient le délai maximal d'inactivité de session (en minutes), au-delà duquel la session est automatiquement fermée par 4D.

Si cette propriété n'est pas définie, sa valeur par défaut est 60 (1h).

Lorsque cette propriété est modifiée, la propriété `.expirationDate` est mise à jour en conséquence.

La valeur ne peut pas être < 60 ; si une valeur inférieure est définie, le timeout est élevé à 60.

Cette propriété est en lecture-écriture.

Exemple

```

If (Session.isGuest())
    // La session d'un invité se ferme après 60 minutes d'inactivité. Session.idleTimeout:=60
Else
    //D'autres sessions se fermeront après 60 minutes d'inactivité. Session.idleTimeout:=120
End if

```

.isGuest()

► Historique

.isGuest() : Boolean

Paramètres	Type		Description
Résultat	Booléen	<-	Vrai s'il s'agit d'une session Guest, sinon Faux

Description

La fonction `.isGuest()` retourne Vrai si la session est une session Guest (i.e. elle n'a pas de priviléges).

Exemple

Dans la méthode base `On Web Connection` :

```

If (Session.isGuest())
    //Faire quelque chose pour l'utilisateur invité
End if

```

.setPrivileges()

► Historique

.setPrivileges(*privilege* : Text)
 .setPrivileges(*privileges* : Collection)
 .setPrivileges(*settings* : Object)

Paramètres	Type		Description
<i>privilege</i>	Text	->	Nom de privilège
<i>privileges</i>	Collection	->	Collection de noms de privilèges
<i>settings</i>	Object	->	Objet contenant une propriété "privileges" (texte ou collection)

Description

La fonction `.setPrivileges()` associe le ou les privilège(s) défini(s) en paramètre à la session.

- Dans le paramètre *privilege*, passez une chaîne contenant un nom de privilège (ou plusieurs noms de privilèges séparés par des virgules).
- Dans le paramètre *privileges*, passez une collection de chaînes contenant des noms de privilèges.
- Dans le paramètre *settings*, passez un objet contenant les propriétés suivantes :

Propriété	Type	Description
privileges	Text ou Collection	<ul style="list-style-type: none"> Chaîne contenant un nom de privilège, ou Collection de chaînes contenant des noms de privilèges
userName	Text	Nom d'utilisateur à associer à la session (optionnel)

Si la propriété `privileges` contient un nom de privilège invalide, il est ignoré.

Dans l'implémentation actuelle, seul le privilège "WebAdmin" est disponible.

Par défaut lorsqu'aucun privilège n'est associé à la session, la session est une [session Guest](#).

La propriété `userName` est accessible au niveau de l'objet `session` (lecture seulement).

Exemple

Dans une méthode d'authentification personnalisée, vous assignez le privilège "WebAdmin" à l'utilisateur :

```
var $userOK : Boolean
...
... //Authentifier l'utilisateur
If ($userOK) //L'utilisateur a été approuvé
    var $info : Object
    $info:=New object()
    $info.privileges:=New collection("WebAdmin")
    Session.setPrivileges($info)
End if
```

.storage

► Historique

`.storage` : Object

Description

La propriété `.storage` contient un objet partagé qui peut être utilisé pour stocker des informations disponibles pour toutes les requêtes du client web.

Lorsqu'un objet `Session` est créé, la propriété `.storage` est vide. Puisqu'il s'agit d'un objet partagé, cette propriété sera disponible dans l'objet `Storage` du serveur.

Tout comme l'objet `Storage` du serveur, la propriété `.storage` est toujours "single" : l'ajout d'un objet partagé ou d'une collection partagée à `.storage` ne crée pas de groupe partagé.

Cette propriété est elle-même en lecture seule mais elle retourne un objet en lecture-écriture.

Exemple

Vous voulez stocker l'adresse IP du client dans la propriété `.storage`. Vous pouvez écrire dans la méthode base `On Web Authentication` :

```
If (Session.storage.clientIP=Null) //premier accès
  Use (Session.storage)
    Session.storage.clientIP:=New shared object("value"; $clientIP)
  End use
End if
```

.userName

► Historique

.userName : Text

Description

La propriété `.userName` contient le nom d'utilisateur associé à la session. Vous pouvez vous en servir pour identifier l'utilisateur dans votre code.

Cette propriété est une chaîne vide par défaut. Elle peut être définie via la propriété `privileges` de la fonction `setPrivileges()`.

Cette propriété est en lecture seule.

Signal

Les signaux sont des outils fournis par le langage 4D pour gérer les interactions et éviter les conflits entre les process dans une application multiprocessus. Les signaux vous permettent de vous assurer qu'un ou plusieurs process attendront la fin d'une tâche spécifique avant de poursuivre leur exécution. Tout process peut attendre et/ou libérer un signal.

Les sémaphores peuvent également être utilisés pour gérer les interactions. Les sémaphores permettent de s'assurer que deux ou plusieurs process ne modifient pas la même ressource (fichier, enregistrement...) au même moment. Seul le process qui a posé le sémaphore peut le retirer.

Objet signal

Un signal est un objet partagé qui doit être passé comme paramètre aux commandes qui appellent ou créent des workers ou des process.

Un objet `4D.Signal` contient les méthodes et propriétés intégrées suivantes :

- `.wait()`
- `.trigger()`
- `.signaled`
- `.description`.

Tout worker/process appelant la méthode `.wait()` suspend son exécution jusqu'à ce que la propriété `.signaled` soit mise à true. Lorsque vous êtes en attente d'un signal, le process appelant n'utilise pas de CPU. Cela peut être très intéressant pour les performances des applications multiprocess. La propriété `.signaled` devient est mise à True lorsqu'un worker/processus appelle la méthode `.trigger()`.

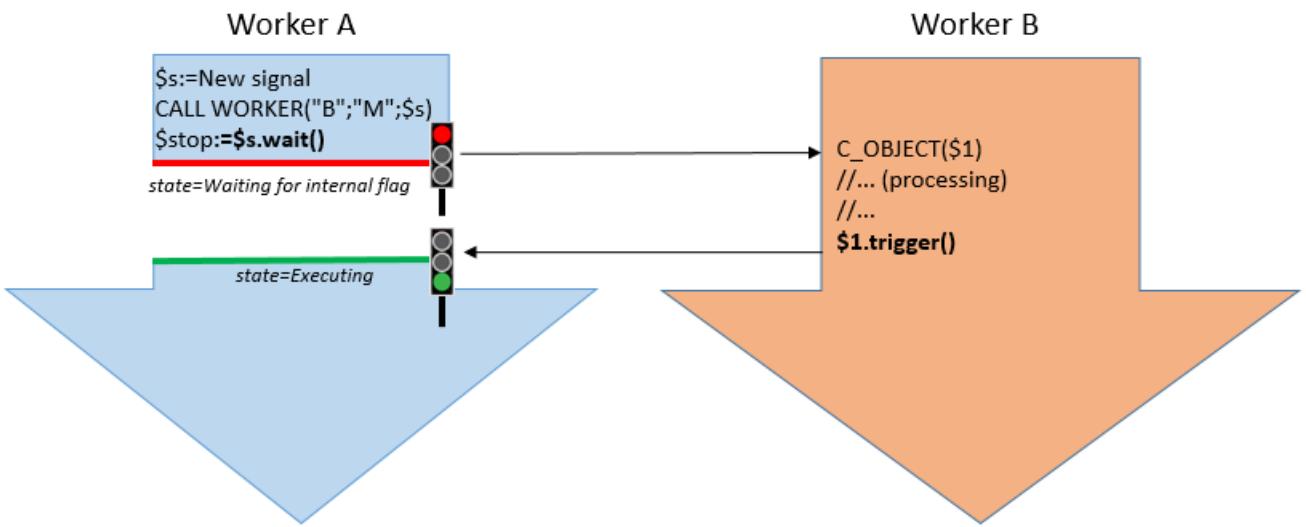
A noter que pour éviter les situations de blocage, la méthode `.wait()` peut également revenir après qu'un délai d'attente défini ait été atteint.

Les objets signal sont créés à l'aide de la commande [New signal](#).

Travailler avec des signaux

Dans 4D, vous créez un nouvel objet signal en appelant la commande [New signal](#). Une fois créé, ce signal doit être passé en paramètre aux commandes `New process` ou `CALL WORKER` afin qu'elles puissent le modifier lorsqu'elles ont terminé la tâche que vous souhaitez attendre.

- `signal.wait()` doit être appelé par le worker/process qui a besoin qu'un autre worker/process termine une tâche pour pouvoir continuer.
- `signal.trigger()` doit être appelé par le worker/process qui a terminé son exécution afin de libérer tous les autres.



Une fois qu'un signal a été libéré par un appel à `signal.trigger()`, il ne peut plus être réutilisé. Si vous souhaitez définir un autre signal, vous devez à nouveau appeler la commande `New signal`.

Etant donné qu'un objet signal est un [objet partagé](#), vous pouvez l'utiliser pour retourner les résultats des workers/process appelés, à condition de ne pas oublier d'écrire les valeurs dans une structure `Use...End use` (voir exemple).

Exemple

```

var $signal : 4D.Signal

// Création d'un signal
$signal:=New signal

// appel du process principal et exécution de la méthode OpenForm
CALL WORKER(1;"OpenForm";$signal)
// autre calcul
...
// Attente de la fin du process
$signaled:=$signal.wait()

// Traitement des résultats
$calc:=$signal.result+...

```

Méthode `OpenForm` :

```

#DECLARE ($signal : 4D.Signal)
var $form : Object
$form:=New object("value";0)

// Ouvrir le form
$win:=Open form window("Information";Movable form dialog box)
DIALOG("Information";$form)
CLOSE WINDOW($win)

// Ajout d'un nouvel attribut à votre objet partagé $signal pour passer votre résultat à l'autre proce
Use($signal)
    $signal.result:=$form.value
End use

// Envoyer le signal au process en attente
$signal.trigger()

```

Sommaire

.description : Text	contient une description personnalisée de l'objet <code>Signal</code>
.signaled : Boolean	contient le statut courant de l'objet <code>Signal</code>
.trigger()	met la propriété <code>signaled</code> de l'objet signal à true
.wait({ <i>timeout</i> : Real }) : Boolean	place le process courant en attente jusqu'à ce que la propriété <code>.signaled</code> de l'objet signal devienne true ou que le <i>timeout</i> optionnel expire

New signal

► Historique

New signal { (*description* : Text) } : 4D.Signal

Paramètres	Type		Description
description	Text	->	Description du signal
Résultat	4D.Signal	<-	Object encapsulant le signal

Description

La commande `New signal` crée un objet `4D.Signal`.

Un signal est un objet partagé qui peut être passé en paramètre depuis un worker ou un process à un autre worker ou process, de manière à ce que :

- le worker/process appelé puisse mettre à jour l'objet signal après qu'un traitement spécifique soit terminé
- le worker/process appelant puisse stopper son exécution et attendre jusqu'à ce que le signal soit mis à jour, sans consommer aucune ressource CPU.

Optionnellement, dans le paramètre *description*, vous pouvez passer un texte personnalisé décrivant le signal. Ce texte peut également être défini après la création du signal.

Comme l'objet signal est un objet partagé, il peut aussi être utilisé pour maintenir des propriétés utilisateur, y compris la propriété `.description`, via l'appel de la structure `Use...End use`.

Valeur renournée

Un nouvel objet `4D.Signal`.

Exemple

Voici un exemple type de worker qui définit un signal :

```

var $signal : 4D.Signal
$signal:=New signal("This is my first signal")

CALL WORKER("myworker";"doSomething";$signal)
$signaled:=$signal.wait(1) //patienter 1 seconde au maximum

If($signaled)
    ALERT("myworker finished the work. Result: "+$signal.myresult)
Else
    ALERT("myworker has not finished in less than 1s")
End if

```

La méthode `doSomething` est par exemple :

```

#DECLARE ($signal : 4D.Signal)
//any processing
/...
Use($signal)
    $signal.myresult:=$processingResult //retourner le résultat
$signal.trigger() // L'opération est terminée

```

.description

► Historique

`.description` : Text

Description

La propriété `.description` contient une description personnalisée de l'objet `Signal`.

`.description` peut être définie à la création de l'objet `signal` ou à tout moment. Notez que comme l'objet `Signal` est un objet partagé, tout accès en mode écriture à la propriété `.description` doit être encadré par les mots-clés `Use...End use`.

Cette propriété est en lecture-écriture.

.signaled

► Historique

`.signaled` : Boolean

Description

La propriété `.signaled` contient le statut courant de l'objet `Signal`. Lorsque le signal est créé, `.signaled` est False. Elle devient True lorsque la fonction `.trigger()` est appelée sur l'objet.

Cette propriété est en lecture seule.

.trigger()

► Historique

`.trigger()` | Paramètres | Type | | Description | | ----- | --- |::| ----- | | | | Ne requiert aucun paramètre |

Description

La fonction `.trigger()` met la propriété `signaled` de l'objet `signal` à true et réveille tous les workers ou process attendant ce signal.

Si le signal est déjà dans l'état signaled (i.e., la propriété `.signaled` est déjà true), la fonction ne fait rien.

.wait()

► Historique

`.wait({ timeout : Real }) : Boolean`

Paramètres	Type		Description
timeout	Real	->	Délai d'attente maximum du signal en secondes
Résultat	Boolean	<-	Etat de la propriété <code>.signaled</code>

Description

La fonction `.wait()` place le process courant en attente jusqu'à ce que la propriété `.signaled` de l'objet signal devienne true ou que le *timeout* optionnel expire.

Pour prévenir tout code bloquant, vous pouvez passer un temps d'attente maximum en secondes dans le paramètre *timeout* (les décimales sont acceptées).

Attention : L'appel de `.wait()` sans *timeout* dans le process principal de 4D n'est pas recommandé car il pourrait geler l'ensemble de l'application 4D.

Si le signal est déjà dans l'état signaled (i.e., la propriété `.signaled` est déjà true), la fonction retourne immédiatement, sans attendre.

La fonction retourne la valeur de la propriété `.signaled`. Evaluer cette valeur permet de savoir si la fonction a retourné à cause de l'appel de `.trigger()` (`.signaled` est true) ou si le *timeout* a expiré (`.signaled` est false).

L'état d'un process qui attend un signal est `En attente d'un marqueur interne`.

SMTPTransporter

La classe `SMTPTransporter` vous permet de configurer des connexions SMTP et d'envoyer des emails par le biais d'objets *SMTP transporter*.

Objet SMTP Transporter

Les objets SMTP Transporter sont instanciés avec la commande [SMTP New transporter](#). Leurs propriétés et fonctions sont les suivantes :

<code>.acceptUnsecureConnection : Boolean</code>	True si 4D est autorisé à établir une connexion non chiffrée
<code>.authenticationMode : Text</code>	le mode d'authentification utilisé pour ouvrir la session sur le serveur de mail
<code>.bodyCharset : Text</code>	contient le charset et l'encodage utilisés pour le corps du mail
<code>.checkConnection() : Object</code>	vérifie la connexion à l'aide des informations stockées dans l'objet transporteur
<code>.connectionTimeOut : Integer</code>	la durée maximale (en secondes) autorisée avant vérification de la connexion au serveur
<code>.headerCharset : Text</code>	le charset et l'encodage utilisés pour l'en-tête de l'email
<code>.host : Text</code>	contient le nom ou l'adresse IP du serveur hôte
<code>.keepAlive : Boolean</code>	True si la connexion SMTP doit rester active jusqu'à la destruction de l'objet <code>transporter</code>
<code>.logFile : Text</code>	le chemin complet du fichier d'historique qui a été défini (le cas échéant) pour la connexion
<code>.port : Integer</code>	le numéro de port utilisé pour les transactions d'emails
<code>.send(mail : Object) : Object</code>	envoie l'objet <code>mail</code> object au serveur SMTP défini dans l'objet <code>transporter</code> et retourne un objet statut
<code>.sendTimeOut : Integer</code>	le temps d'attente maximal (en secondes) d'un appel à <code>.send()</code> avant le timeout (l'expiration du délai)
<code>.user : Text</code>	le nom d'utilisateur employé pour l'authentification sur le serveur mail

SMTP New transporter

► Historique

[SMTP New transporter\(server : Object \) : 4D.SMTPTransporter](#)

Paramètres	Type		Description
server	Object	->	Informations sur le serveur IMAP
Résultat	4D.SMTPTransporter	<-	Objet SMTP Transporter

Description

La commande `SMTP New transporter` configure une nouvelle connexion SMTP en fonction du paramètre `server` et retourne un nouvel objet `SMTP transporter`. L'objet transporteur retourné sera alors utilisé pour l'envoi d'emails.

Cette commande n'ouvre pas de connexion au serveur SMTP. La connexion SMTP est réellement ouverte lorsque la fonction `.send()` est exécutée.

La connexion SMTP est automatiquement fermée : * lorsque l'objet transporteur est détruit si la propriété `keepAlive` est mise à true (par défaut), * après chaque exécution de la fonction `.send()` si la propriété `keepAlive` est mise à false.

Dans le paramètre `server`, passez un objet contenant les propriétés suivantes :

<code>server</code>	Valeur par défaut (si omise)
<code>.acceptUnsecureConnection : Boolean</code> True si 4D est autorisé à établir une connexion non chiffrée	Faux
<code>.accessTokenOAuth2 : Text</code> <code>.accessTokenOAuth2 : Object</code> Chaîne de texte ou objet token représentant les informations d'autorisation OAuth2. Il est utilisé uniquement avec <code>OAUTHP2 authenticationMode</code> . Si <code>accessTokenOAuth2</code> est utilisé mais que <code>authenticationMode</code> est omis, le protocole OAuth 2 est utilisé (si le serveur l'autorise). Non retourné en objet <code>SMTP transporter</code> .	aucune
<code>.authenticationMode : Text</code> le mode d'authentification utilisé pour ouvrir la session sur le serveur de mail	le mode d'authentification le plus sûr pris en charge par le serveur est utilisé
<code>.bodyCharset : Text</code> contient le charset et l'encodage utilisés pour le corps du mail	<code>mail mode UTF8 (US-ASCII_UTF8_QP)</code>
<code>.connectionTimeOut : Integer</code> la durée maximale (en secondes) autorisée avant vérification de la connexion au serveur	30
<code>.headerCharset : Text</code> le charset et l'encodage utilisés pour l'en-tête de l'email	<code>mail mode UTF8 (US-ASCII_UTF8_QP)</code>
<code>.host : Text</code> contient le nom ou l'adresse IP du serveur hôte	<i>obligatoire</i>
<code>.keepAlive : Boolean</code> True si la connexion SMTP doit rester active jusqu'à la destruction de l'objet <code>transporter</code>	Vrai
<code>.logFile : Text</code> le chemin complet du fichier d'historique qui a été défini (le cas échéant) pour la connexion	aucune
<code>password : Text</code> Mot de passe de l'utilisateur pour l'authentification sur le serveur. Non retourné en objet <code>SMTP transporter</code> .	aucune
<code>.port : Integer</code> le numéro de port utilisé pour les transactions d'emails	587
<code>.sendTimeOut : Integer</code> le temps d'attente maximal (en secondes) d'un appel à <code>.send()</code> avant le timeout (l'expiration du délai)	100
<code>.user : Text</code> le nom d'utilisateur employé pour l'authentification sur le serveur mail	aucune

Résultat

La fonction retourne un `objet SMTP transporter`. Toutes les propriétés retournées sont en lecture seule.

Exemple

```
$server:=New object
$server.host:="smtp.gmail.com" //Obligatoire
$server.port:=465
$server.user:="4D@gmail.com"
$server.password:="XXXX"
$server.logFile:="LogTest.txt" //Log étendu à sauvegarder dans le dossier Logs

var $transporter : 4D.SMTPTransporter
$transporter:=SMTP New transporter($server)

$email:=New object
$email.subject:="my first mail "
$email.from:="4d@gmail.com"
$email.to:="4d@4d.com;test@4d.com"
$email.textBody:="Hello World"
$email.htmlBody:="<h1>Hello World</h1><h4>'Neque porro quisquam est qui dolorem ipsum quia dolor sit am
<p>There are many variations of passages of Lorem Ipsum available."\
+"The generated Lorem Ipsum is therefore always free from repetition, injected humour, or non-character

$status:=$transporter.send($email)
If(Not($status.success))
    ALERT("An error occurred sending the mail: "+$status.message)
End if
```

4D.SMTPTransporter.new()

4D.SMTPTransporter.new(server : Object) : 4D.SMTPTransporter

Paramètres	Type		Description
server	Object	->	Informations sur le serveur IMAP
Résultat	4D.SMTPTransporter	<-	Objet SMTP Transporter

Description

La fonction `4D.SMTPTransporter.new()` crée et retourne un nouvel objet de type `4D.SMTPTransporter`. Elle est identique à la commande `SMTP New transporter` (raccourci).

.acceptUnsecureConnection

► Historique

.acceptUnsecureConnection : Boolean

Description

La propriété `.acceptUnsecureConnection` contient True si 4D est autorisé à établir une connexion non chiffrée lorsqu'une connexion chiffrée n'est pas possible.

Elle contient False si les connexions non chiffrées ne sont pas autorisées, auquel cas une erreur est retournée lorsque la connexion chiffrée n'est pas possible.

Ports sécurisés disponibles :

- SMTP
 - 465: SMTPTS
 - 587 ou 25 : SMTP avec mise à niveau STARTTLS si le serveur le prend en charge.

- IMAP
 - 143 : Port IMAP non chiffré
 - 993 : IMAP avec mise à niveau STARTTLS si le serveur le prend en charge
- POP3
 - 110 : Port POP3 non chiffré
 - 995 : POP3 avec mise à niveau STARTTLS si le serveur le prend en charge.

.authenticationMode

► Historique

.authenticationMode : Text

Description

La propriété `.authenticationMode` contient le mode d'authentification utilisé pour ouvrir la session sur le serveur de messagerie.

Par défaut, le mode le plus sécurisé pris en charge par le serveur est utilisé.

Valeurs possibles :

Valeur	Constantes	Commentaire
CRAM-MD5	<code>SMTP authentication CRAM MD5</code>	Authentification à l'aide du protocole CRAM-MD5
LOGIN	<code>SMTP authentication login</code>	Authentification à l'aide du protocole LOGIN
OAuth2	<code>SMTP authentication OAuth2</code>	Authentification à l'aide du protocole OAuth2
PLAIN	<code>SMTP authentication plain</code>	Authentification à l'aide du protocole PLAIN

.bodyCharset

► Historique

.bodyCharset : Text

Description

La propriété `.bodyCharset` contient le charset et l'encodage utilisés pour le corps du mail.

- subject,
- attachment filename(s),
- email name.

Valeurs possibles :

Constante	Valeur	Commentaire
mail mode ISO2022JP	US-ASCII_ISO-2022-JP_UTF8_QP	<ul style="list-style-type: none"> • <i>headerCharset</i> : US-ASCII si possible, japonais (ISO-2022-JP) & Quoted-printable si possible, sinon UTF-8 & Quoted-printable • <i>bodyCharset</i> : US-ASCII si possible, japonais (ISO-2022-JP) et 7 bits si possible, sinon UTF-8 et Quoted-printable
mail mode ISO88591	ISO-8859-1	<ul style="list-style-type: none"> • <i>headerCharset</i> : ISO-8859-1 & Quoted-printable • <i>bodyCharset</i> : ISO-8859-1 et 8 bits
mail mode UTF8	US-ASCII_UTF8_QP	<i>headerCharset</i> & <i>bodyCharset</i> : US-ASCII si possible, sinon UTF-8 & Quoted-printable (valeur par défaut)
mail mode UTF8 in base64	US-ASCII_UTF8_B64	<i>headerCharset</i> & <i>bodyCharset</i> : US-ASCII si possible, sinon UTF-8 & base64

.checkConnection()

► Historique

.checkConnection() : Object

Paramètres	Type		Description
Résultat	Object	<-	Statut de la connexion de l'objet transporteur

Description

La fonction `.checkConnection()` vérifie la connexion à l'aide des informations stockées dans l'objet transporteur, recréée la connexion si nécessaire et retourne son statut. Cette fonction vous permet de vérifier que les valeurs fournies par l'utilisateur sont valides et cohérentes.

Objet retourné

La fonction envoie une requête au serveur de mail et retourne un objet décrivant le statut. Cet objet peut avoir les propriétés suivantes :

Propriété		Type	Description
success		boolean	Vrai si la vérification a été effectuée avec succès, sinon Faux
status		number	(SMTP uniquement) Code du statut retourné par le serveur de messagerie (0 en cas de problème non lié au traitement du mail)
statusText		text	Message du statut retourné par le serveur de messagerie, ou dernière erreur retournée dans la pile d'erreurs 4D
errors		collection	Pile d'erreurs 4D (non retournée si une réponse du serveur de messagerie est reçue)
	[].errCode	number	Code d'erreur 4D
	[].message	text	Description de l'erreur 4D
	[].componentSignature	text	Signature du composant interne qui a retourné l'erreur

Pour une description des codes de statut SMTP, veuillez vous reporter à [cette page](#).

Exemple

```

var $pw : Text
var $options : Object
var $transporter : 4D.SMTPTransporter
$options:=New object

$pw:=Request("Please enter your password:")
$options.host:="smtp.gmail.com"

$options.user:="test@gmail.com"
$options.password:=$pw

$transporter:=SMTP New transporter($options)

$status:=$transporter.checkConnection()
If($status.success=True)
    ALERT("SMTP connection check successful!")
Else
    ALERT("Error # "+String($status.status)+", "+$status.statusText)
End if

```

.connectionTimeOut

► Historique

.connectionTimeOut : Integer

Description

La propriété `.connectionTimeOut` contient la durée maximale (en secondes) autorisée avant vérification de la connexion au serveur. Par défaut, si la propriété n'a pas été définie dans l'objet server (utilisé pour créer l'objet transporteur avec `SMTP New transporter`, `POP3 New transporter`, `IMAP New transporter`), la valeur utilisée est 30.

.headerCharset

► Historique

.headerCharset : Text

Description

La propriété `.headerCharset` contient le charset et l'encodage utilisés pour l'en-tête de l'email. L'en-tête comprend les éléments suivants de l'e-mail :

- subject,
- attachment filename(s),
- email name.

Valeurs possibles :

Constante	Valeur	Commentaire
mail mode ISO2022JP	US-ASCII_ISO-2022-JP_UTF8_QP	<ul style="list-style-type: none"> • <i>headerCharset</i> : US-ASCII si possible, japonais (ISO-2022-JP) & Quoted-printable si possible, sinon UTF-8 & Quoted-printable • <i>bodyCharset</i> : US-ASCII si possible, japonais (ISO-2022-JP) et 7 bits si possible, sinon UTF-8 et Quoted-printable
mail mode ISO88591	ISO-8859-1	<ul style="list-style-type: none"> • <i>headerCharset</i> : ISO-8859-1 & Quoted-printable • <i>bodyCharset</i> : ISO-8859-1 et 8 bits
mail mode UTF8	US-ASCII_UTF8_QP	<i>headerCharset</i> & <i>bodyCharset</i> : US-ASCII si possible, sinon UTF-8 & Quoted-printable (valeur par défaut)
mail mode UTF8 in base64	US-ASCII_UTF8_B64	<i>headerCharset</i> & <i>bodyCharset</i> : US-ASCII si possible, sinon UTF-8 & base64

.host

► Historique

.host : Text

Description

La propriété `.host` contient le nom ou l'adresse IP du serveur hôte. Utilisée pour les échanges de courrier (SMTP, POP3, IMAP).

.keepAlive

► Historique

.keepAlive : Boolean

Description

La propriété `.keepAlive` contient True si la connexion SMTP doit rester active jusqu'à la destruction de l'objet `transporter`, et False sinon. Par défaut, si la propriété `keepAlive` n'a pas été définie dans l'objet `server` qui permet de créer l'objet `transporter` via la commande `SMTP New transporter`), elle est mise à True.

La connexion SMTP est automatiquement fermée :

- lorsque l'objet `transporter` est détruit si la propriété `.keepAlive` est mise à vrai,
- après chaque fonction `.send()` exécutée si la propriété `.keepAlive` est mise à faux.

.logFile

► Historique

.logFile : Text

Description

La propriété `.logFile` contient le chemin complet du fichier d'historique qui a été défini (le cas échéant) pour la connexion. Elle peut être relative (au dossier Logs courant) ou absolue.

Contrairement aux fichiers log habituels (activés via la commande `SET DATABASE PARAMETER`), les fichiers log étendus stockent le contenu MIME de tous les emails envoyés et n'ont pas de limite de taille. Pour plus d'informations sur les fichiers log étendus, reportez-vous à :

- Connexions SMTP - [4DSMTPLog.txt](#)
- Connexions POP3 - [4DPOP3Log.txt](#)
- Connexions IMAP - [4DIMAPLog.txt](#)

.port

► Historique

.port : Integer

Description

La propriété `.port` contient le numéro de port utilisé pour les transactions d'emails. Par défaut, si la propriété `port` n'a pas été définie dans l'objet `server` (utilisé pour créer l'objet transporteur avec `SMTP New transporter`, `POP3 New transporter`, `IMAP New transporter`), le port utilisé est :

- SMTP - 587
- POP3 - 995
- IMAP - 993

.send()

► Historique

.send(*mail* : Object) : Object

Paramètres	Type		Description
mail	Object	->	Email à envoyer
Résultat	Object	<-	Statut SMTP

Description

La fonction `.send()` envoie l'objet `mail object` au serveur SMTP défini dans l'objet `transporter` et retourne un objet statut.

L'objet `transporter` doit avoir déjà été créé à l'aide de la commande `SMTP New transporter`.

La fonction établit la connexion SMTP si cette dernière n'est pas déjà active. Si la propriété `.keepAlive` de l'objet `transporter` est à `false`, la connexion SMTP est automatiquement fermée après l'exécution de la commande `.send()`. Pour plus d'informations, voir la description de la commande `SMTP New transporter`.

Dans `mail`, passez un objet `Email` valide à envoyer. Les propriétés origine (la provenance de l'Email) et destination (un ou plusieurs destinataires) de l'Email doivent être incluses, les autres propriétés sont optionnelles.

Objet retourné

La fonction retourne un objet décrivant le statut SMTP de l'opération. Cet objet peut avoir les propriétés suivantes :

Propriété	Type	Description
success	boolean	Vrai si l'envoi a été effectué avec succès, sinon Faux
status	number	Code du statut retourné par le serveur SMTP (0 si problème non lié au traitement de l'email)
statusText	Texte	Message de statut retourné par le serveur SMTP

En cas de problème non lié au traitement SMTP (par exemple une propriété obligatoire qui est manquante dans l'objet `mail`), 4D génère une erreur que vous pouvez intercepter à l'aide d'une méthode installée via la commande `ON ERR CALL`. Utilisez la commande `GET LAST ERROR STACK` pour obtenir des informations sur l'erreur.

Dans ce cas, l'objet erreur qui en résulte contient les valeurs suivantes :

Propriété	Valeur
success	Faux
status	0
statusText	"Failed to send email"

.sendTimeOut

► Historique

.sendTimeOut : Integer

Description

La propriété `.sendTimeOut` contient le temps d'attente maximal (en secondes) d'un appel à `.send()` avant le timeout (l'expiration du délai). Par défaut, si la propriété `.sendTimeOut` n'a pas été définie dans l'objet `server`, la valeur 100 est utilisée.

.user

► Historique

.user : Text

Description

La propriété `.user` contient le nom d'utilisateur employé pour l'authentification sur le serveur mail.

SystemWorker

Les System workers permettent au code 4D d'appeler n'importe quel process externe (une commande shell, PHP, etc.) sur la même machine. Les System workers sont appelés de manière asynchrone. En utilisant des callbacks, 4D permet de communiquer dans les deux sens.

La classe `SystemWorker` est disponible dans le class store `4D`.

Exemple

```
// Exemple Windows pour avoir accès aux informations d'ipconfig
var $myWinWorker : 4D.SystemWorker
var $ipConfig : Text
$myWinWorker:= 4D.SystemWorker.new("ipconfig")
$ipConfig:=$myWinWorker.wait(1).response //timeout 1 seconde

// Exemple macOS pour modifier les permissions d'un fichier sous macOS
// chmod est la commande macOS utilisée pour modifier l'accès aux fichiers
var $myMacWorker : 4D.SystemWorker
$myMacWorker:= 4D.SystemWorker.new("chmod x /folder/myfile.sh")
```

Sommaire

`4D.SystemWorker.new (commandLine : Text { ; options : Object }) : 4D.SystemWorker`

crée et renvoie un objet `4D.SystemWorker` qui exécutera la `commandLine` que vous avez passée en paramètre pour lancer un process externe

`.closeInput()`

ferme le flux d'entrée (`stdin`) du process externe

`.commandLine : Text`

contient la ligne de commande passée en paramètre à la fonction `new()`

`.currentDirectory : 4D.Folder`

contient le répertoire de travail dans lequel le process externe est exécuté

`.dataType : Text`

contient le type du contenu du corps de la réponse

`.encoding : Text`

contient l'encodage du contenu du corps de la réponse

`.errors : Collection`

contient une collection d'erreurs 4D dans le cas d'erreur(s) d'exécution

`.exitCode : Integer`

contient le code de sortie renvoyé par le process externe

`.hideWindow : Boolean`

peut être utilisée pour cacher la fenêtre de la console DOS ou la fenêtre de l'exécutable lancé (Windows uniquement)

`.pid : Integer`

contient l'identifiant unique du process externe au niveau du système

`.postMessage(message : Text)`

`.postMessage(messageBLOB : Blob)`

permet d'écrire dans le flux d'entrée (`stdin`) du process externe

`.response : Text`

`.response : Blob`

contient la concaténation de toutes les données renvoyées une fois la demande terminée

`.responseError : Text`

contient la concaténation de toutes les erreurs renvoyées, une fois la requête terminée

`.terminate()`

force le `SystemWorker` à terminer son exécution

`.terminated : Boolean`

contient true si le process externe est terminé

`.timeout : Integer`

contient la durée en secondes avant que le process externe ne soit tué s'il est toujours en vie

`.wait({timeout : Real}) : 4D.SystemWorker`

attend la fin de l'exécution du `SystemWorker` ou du `timeout` spécifié

4D.SystemWorker.new()

► Historique

4D.SystemWorker.new (*commandLine* : Text { ; *options* : Object }) : 4D.SystemWorker

Paramètres	Type		Description
<i>commandLine</i>	Text	->	Ligne de commande à exécuter
<i>options</i>	Object	->	Paramètres du worker
<i>result</i>	4D.SystemWorker	<-	Nouveau System worker asynchrone ou null si le process n'a pas démarré

Description

La fonction `4D.SystemWorker.new()` crée et renvoie un objet `4D.SystemWorker` qui exécutera la *commandLine* que vous avez passée en paramètre pour lancer un process externe.

L'objet system worker retourné peut être utilisé pour envoyer des messages au worker et obtenir les résultats du worker.

Si une erreur se produit pendant la création de l'objet proxy, la fonction renvoie un objet `null` et une erreur est levée.

Dans le paramètre *commandLine* , passez le chemin complet du fichier de l'application à exécuter (syntaxe posix), ainsi que les arguments requis, si nécessaire. Si vous ne passez que le nom de l'application, 4D utilisera la variable d'environnement `PATH` pour localiser l'exécutable.

Attention : Cette fonction ne peut lancer que des applications exécutables ; elle ne peut pas exécuter les instructions qui font partie du shell (interpréteur de commandes). Par exemple, sous Windows, il n'est pas possible d'utiliser cette commande pour exécuter l'instruction `dir` .

Objet *options*

Dans le paramètre *options* , passez un objet qui peut contenir les propriétés suivantes :

Propriété	Type	Par défaut	Description
onResponse	Formula	indéfini	Callback pour les messages du system worker. Ce callback est appelé une fois que la réponse complète est reçue. Il reçoit deux objets en paramètres (voir ci-dessous)
onData	Formula	indéfini	Callback pour les données du system worker. Ce callback est appelé chaque fois que le system worker reçoit des données. Il reçoit deux objets en paramètres (voir ci-dessous)
onDataError	Formula	indéfini	Callback pour les erreurs du process externe (<code>stderr</code> du process externe). Il reçoit deux objets en paramètres (voir ci-dessous)
onError	Formula	indéfini	Callback pour les erreurs d'exécution, renvoyées par le system worker en cas de conditions d'exécution inhabituelles (erreurs système). Il reçoit deux objets en paramètres (voir ci-dessous)
onTerminate	Formula	indéfini	Callback lorsque le process externe est terminé. Il reçoit deux objets en paramètres (voir ci-dessous)
timeout	Nombre	indéfini	Délai en secondes avant que le process soit tué s'il est toujours actif
dataType	Text	"text"	Type de contenu du corps de la réponse. Valeurs possibles : "text" (par défaut), "blob".
encoding	Text	"UTF-8"	Seulement si <code>dataType="text"</code> . Encodage du contenu du corps de la réponse. Pour la liste des valeurs disponibles, voir la description de la commande CONVERT FROM TEXT
variables	Object		Définit des variables d'environnement personnalisées pour le system worker. Syntaxe : <code>variables.key=value</code> , où <code>key</code> est le nom de la variable et <code>value</code> sa valeur. Les valeurs sont converties en chaînes de caractères lorsque cela est possible. La valeur ne peut pas contenir un '='. S'il n'est pas défini, le system worker hérite de l'environnement 4D.
currentDirectory	Folder		Répertoire de travail dans lequel le process est exécuté
hideWindow	Booléen	true	(Windows) Masquer la fenêtre de l'application (si possible) ou la console Windows

Toutes les fonctions de callback reçoivent deux paramètres objet. Leur contenu dépend du callback :

Paramètres	Type	onResponse	onData	onDataError	onError	onTerminate
\$param1	Object	SystemWorker	SystemWorker	SystemWorker	SystemWorker	SystemWorker
\$param2.type	Text	"response"	"data"	"error"	"error"	"termination"
\$param2.data	Text ou Blob		données reçues	données d'erreur		

Voici la séquence des appels de callbacks :

1. `onData` et `onDataError` sont exécutés une ou plusieurs fois
2. s'il est appelé, `onError` est exécuté une fois (arrête le traitement du system worker)
3. si aucune erreur ne s'est produite, `onResponse` est exécuté une fois
4. `onTerminate` est toujours exécuté

Valeur retournée

La fonction renvoie un objet system worker sur lequel vous pouvez appeler les fonctions et les propriétés de la classe SystemWorker.

Exemples sous Windows

1. Pour ouvrir le Bloc-notes et ouvrir un document spécifique :

```
var $sw : 4D.SystemWorker
var $options : Object
$options:=New object
$options.hideWindow:= False

$sw:=4D.SystemWorker.new ("C:\\WINDOWS\\notepad.exe C:\\Docs\\new folder\\res.txt";$options)
```

2. Exécuter npm install dans la console :

```
var $folder : 4D.Folder
var $options : Object
var $worker : 4D.SystemWorker

$folder:=Folder(fk database folder)
$options:=New object
$options.currentDirectory:=$folder
$options.hideWindow:=False

$worker:=4D.SystemWorker.new("cmd /c npm install";$options)
```

3. Pour lancer l'application Microsoft® Word® et ouvrir un document spécifique :

```
$mydoc:="C:\\Program Files\\Microsoft Office\\Office15\\WINWORD.EXE C:\\Tempo\\output.txt"
var $sw : 4D.SystemWorker
$sw:=4D.SystemWorker.new($mydoc)
```

4. Pour lancer une commande avec le répertoire courant et poster un message :

```
var $param : Object
var $sys : 4D.SystemWorker

$param:=New object
$param.currentDirectory:=Folder(fk database folder)
$sys:=4D.SystemWorker.new("git commit -F -";$param)
$sys.postMessage("This is a postMessage")
$sys.closeInput()
```

5. Pour permettre à l'utilisateur d'ouvrir un document externe sous Windows :

```
$docname:=Select document(" ; \"*.*\" ; \"Choisissez le fichier à ouvrir\";0)
If(OK=1)
    var $sw : 4D.SystemWorker
    $sw:=4D.SystemWorker.new("cmd.exe /C start \"\" \"$docname\"\"")
End if
```

Exemples sous macOS

1. Modifier un fichier texte (cat est la commande macOS utilisée pour modifier les fichiers). Dans cet exemple, le chemin d'accès complet de la commande est transmis :

```

var $sw : 4D.SystemWorker
$sw:=4D.SystemWorker.new("/bin/cat /folder/myfile.txt")
$sw.wait() /exécution synchrone

```

2. Pour lancer une application "graphique" indépendante, il est préférable d'utiliser la commande système `open` (dans ce cas, le code a le même effet qu'un double-clic sur l'application) :

```

var $sw : 4D.SystemWorker
$sw:=4D.SystemWorker.new ("open /Applications/Calculator.app")

```

3. Pour obtenir le contenu du dossier "Users" (ls -l est l'équivalent sous macOS de la commande dir sous DOS).

```

var $systemworker : 4D.SystemWorker
var $output : Text
var $errors : Collection

$systemworker:=4D.SystemWorker.new("/bin/ls -l /Users ")
$output:=$systemworker.response
$errors:=$systemworker.errors

```

4. Même commande que ci-dessus, mais en utilisant un exemple de classe utilisateur "Params" pour montrer comment gérer les fonctions de callback :

```

var $systemworker : 4D.SystemWorker
$systemworker:=4D.SystemWorker.new("/bin/ls -l /Users ";cs.Params.new())

// "Params" class

Class constructor
This.dataType:="text"
This.data:=""
This.dataError=""

Function onResponse($systemWorker : Object)
This._createFile("onResponse"; $systemWorker.response)

Function onData($systemWorker : Object; $info : Object)
This.data+=$info.data
This._createFile("onData";this.data)

Function onDataError($systemWorker : Object; $info : Object)
This.dataError+=$info.data
This._createFile("onDataError";this.dataError)

Function onTerminate($systemWorker : Object)
var $textBody : Text
$textBody:="Response: "+$systemWorker.response
$textBody+="ResponseError: "+$systemWorker.responseError
This._createFile("onTerminate"; $textBody)

Function _createFile($title : Text; $textBody : Text)
TEXT TO DOCUMENT(Get 4D folder(Current resources folder)+$title+".txt"; $textBody)

```

.closeInput()

► Historique

.closeInput() | Paramètres | Type | | Description | | ----- | ---- |::| ----- | | | | Ne requiert aucun paramètre |

Description

La fonction `.closeInput()` ferme le flux d'entrée (`stdin`) du process externe.

Lorsque l'exécutable attend que toutes les données soient reçues par `postMessage()`, `.closeInput()` est utile pour indiquer à l'exécutable que l'envoi des données est terminé et qu'il peut continuer.

Exemple

```
// Créer quelques données à gzipper
var $input;$output : Blob
var $gzip : Text
TEXT TO BLOB("Hello, World !" ;$input)
$gzip:="\"C:\\Program Files (x86)\\GnuWin32\\bin\\\\\\gzip.exe\" "

// Créer un system worker asynchrone
var $worker : 4D.SystemWorker
$worker:= 4D.SystemWorker.new($gzip;New object("dataType" ; "blob"))

// Envoyer le fichier compressé sur stdin.
$worker.postMessage($input)
// Notez que nous appelons closeInput() pour indiquer que nous avons terminé.
// gzip (et la plupart des programmes attendant des données de stdin) attendra plus de données jusqu'à ce que
$worker.closeInput()
$worker.wait()

$output:=$worker.response
```

.commandLine

.commandLine : Text

Description

La propriété `.commandLine` contient la ligne de commande passée en paramètre à la fonction `new()`.

Cette propriété est en lecture seule.

.currentDirectory

.currentDirectory : 4D.Folder

Description

La propriété `.currentDirectory` contient le répertoire de travail dans lequel le process externe est exécuté.

.dataType

.dataType : Text

Description

La propriété `.dataType` contient le type du contenu du corps de la réponse. Possible values : "text" or "blob".

Cette propriété est en lecture seule.

.encoding

`.encoding` : Text

Description

La propriété `.encoding` contient l'encodage du contenu du corps de la réponse. Cette propriété est uniquement disponible si le `dataType` est "text".

Cette propriété est en lecture seule.

.errors

`.errors` : Collection

Description

La propriété `.errors` contient une collection d'erreurs 4D dans le cas d'erreur(s) d'exécution.

Chaque élément de la collection est un objet avec les propriétés suivantes :

Propriété	Type	Description
<code>[]errorCode</code>	number	Code d'erreur 4D
<code>[]message</code>	Texte	Description de l'erreur 4D
<code>[].componentSignature</code>	Texte	Signature du composant interne qui a retourné l'erreur

Si aucune erreur ne s'est produite, `.errors` contient une collection vide.

.exitCode

`.exitCode` : Integer

Description

La propriété `.exitCode` contient le code de sortie renvoyé par le process externe. Si le process ne s'est pas terminé normalement, `exitCode` est *undefined*.

Cette propriété est en lecture seule.

.hideWindow

`.hideWindow` : Boolean

Description

La propriété `.hideWindow` peut être utilisée pour cacher la fenêtre de la console DOS ou la fenêtre de l'exécutable lancé (Windows uniquement).

Cette propriété est en lecture-écriture.

.pid

`.pid` : Integer

Description

La propriété `.pid` contient l'identifiant unique du process externe au niveau du système.

Cette propriété est en lecture seule.

`.postMessage()`

`.postMessage(message : Text)`
`.postMessage(messageBLOB : Blob)`

Paramètres	Type		Description
message	Text	->	Texte à écrire dans le flux d'entrée (<code>stdin</code>) du process externe
messageBLOB	Blob	->	Octets écrits dans le flux d'entrée

Description

La fonction `.postMessage()` permet d'écrire dans le flux d'entrée (`stdin`) du process externe. Dans le paramètre `message`, passez le texte à écrire dans `stdin`.

La fonction `.postMessage()` accepte également une valeur de type Blob dans `messageBLOB`, de sorte que vous pouvez poster des données binaires dans `stdin`.

Vous pouvez utiliser la propriété `.dataType` de l'objet `options` pour que le corps de réponse renvoie des valeurs Blob.

`.response`

`.response : Text`
`.response : Blob`

Description

La propriété `.response` contient la concaténation de toutes les données renvoyées une fois la demande terminée, c'est-à-dire le message complet reçu à partir de l'output du process.

Le type du message est défini selon l'attribut `dataType`.

Cette propriété est en lecture seule.

`.responseError`

`.responseError : Text`

Description

La propriété `.responseError` contient la concaténation de toutes les erreurs retournées, une fois la requête terminée.

`.terminate()`

`.terminate()` | Paramètres | Type | | Description | | ----- | ---- | ::| ----- | | | | | Ne requiert aucun paramètre |

Description

La fonction `.terminate()` force le `SystemWorker` à terminer son exécution.

Cette fonction envoie l'instruction de terminer et de redonner le contrôle au script en cours d'exécution.

.terminated

.terminated : Boolean

Description

La propriété `.terminated` contient true si le process externe est terminé.

Cette propriété est en lecture seule.

.timeout

.timeout : Integer

Description

La propriété `.timeout` contient la durée en secondes avant que le process externe ne soit tué s'il est toujours en vie.

Cette propriété est en lecture seule.

.wait()

► Historique

`.wait({timeout : Real}) : 4D.SystemWorker`

Paramètres	Type		Description
timeout	Réel	->	Temps d'attente (en secondes)
Résultat	4D.SystemWorker	<-	Objet SystemWorker

Description

La fonction `.wait()` attend la fin de l'exécution du `SystemWorker` ou du `timeout` spécifié.

Dans `timeout`, passez une valeur en secondes. Le script `SystemWorker` attendra le process externe pendant la durée définie dans le paramètre `timeout`. Si vous omettez le paramètre `timeout`, l'exécution du script attendra indéfiniment.

En fait, `.wait()` attend la fin du traitement de la formule `onTerminate`, sauf si le `timeout` est atteint. Si le `timeout` est atteint, le `SystemWorker` n'est pas tué.

Pendant une exécution `.wait()`, les fonctions de callback sont exécutées, en particulier les callbacks provenant d'autres événements ou d'autres instances de `SystemWorker`. Vous pouvez sortir d'un `.wait()` en appelant `terminate()` à partir d'un callback.

Cette fonction renvoie l'objet `SystemWorker`.

Cette fonction n'est pas nécessaire si vous avez créé le `SystemWorker` à partir d'un process worker 4D.

WebServer

La classe `WebServer` vous permet de démarrer et de contrôler un serveur web pour l'application principale (hôte) ainsi que pour chaque composant (voir la présentation de l'[objet Web Server](#)). Cette classe est disponible depuis le "class store" de `4D`.

Objet Serveur Web

Les objets Web server sont installés à l'aide de la commande `WEB Server`.

Leurs propriétés et fonctions sont les suivantes :

Sommaire

<code>.accessKeyDefined : Boolean</code> vrai si une access key est définie dans les settings du serveur Web
<code>.certificateFolder : Text</code> dossier contenant les fichiers des certificats
<code>.characterSet : Number</code> <code>.characterSet : Text</code>
<code>.cipherSuite : Text</code>
<code>.CORSEnabled : Boolean</code>
<code>.CORSSettings : Collection</code>
<code>.debugLog : Number</code>
<code>.defaultHomepage : Text</code>
<code>.HSTSEnabled : Boolean</code>
<code>.HSTSMaxAge : Number</code>
<code>.HTTPCompressionLevel : Number</code>
<code>.HTTPCompressionThreshold : Number</code>
<code>.HTTPEnabled : Boolean</code>

.HTTPPort : Number	Numéro de port IP d'écoute pour HTTPS
.HTTPTrace : Boolean	
.HTTPSEnabled : Boolean	
.HTTPSPort : Number	
.inactiveProcessTimeout : Number	Durée de vie (en minutes) des sessions legacy inactives
.inactiveSessionTimeout : Number	
.IPAddressToListen : Text	
.isRunning : Boolean	Statut démarré du serveur Web
.keepSession : Boolean	Vrai si les sessions évolutives (scalable sessions) sont activées dans le serveur Web, sinon Faux
.logRecording : Number	
.maxConcurrentProcesses : Number	
.maxRequestSize : Number	
.maxSessions : Number	Nombre maximum de sessions legacy simultanées
.minTLSVersion : Number	
.name : Text	
.openSSLVersion : Text	Version de la librairie OpenSSL utilisée
.perfectForwardSecrecy : Boolean	Disponibilité du PFS sur le serveur
.rootFolder : Text	
.scalableSession : Boolean	Vrai si les sessions évolutives (scalable sessions) sont activées dans le serveur Web, sinon Faux

```

.sessionCookieDomain : Text
| | .sessionCookieName : Text
| | .sessionCookiePath : Text
Champ "path" du cookie de session| | .sessionCookieSameSite : Text
| | .sessionIPAddressValidation : Boolean
Validation de l'adresse IP des cookies de session| | .start() : Object
.start( settings : Object ) : Object
démarre le serveur Web server auquel elle est appliquée| | .stop()
arrête le serveur Web auquel elle est appliquée|

```

WEB Server

► Historique

WEB Server : 4D.WebServer

WEB Server(option : Integer) : 4D.WebServer

Paramètres	Type		Description
option	Integer	->	Serveur Web à référencer (défaut si omis = Web server database)
Résultat	4D.WebServer	<-	Objet Serveur Web

La commande `WEB Server` retourne l'objet Web server par défaut ou l'objet Web server désigné par le paramètre *option*.

Le serveur Web démarre avec les paramètres par défaut définis dans le fichier de settings du projet ou (base hôte uniquement) à l'aide de la commande `WEB SET OPTION`. Cependant, à l'aide du paramètre *settings*, vous pouvez définir des paramètres personnalisés pour la session du serveur Web.

Constante	Valeur	Commentaire
Web server database	1	Le serveur Web de la base courante (par défaut si omis)
Web server host database	2	Le serveur Web de la base hôte du composant
Web server receiving request	3	Le serveur Web ayant reçu la requête (serveur Web cible)

L'objet Web server retourné contient les valeurs courantes des propriétés du serveur Web.

Exemple

L'objet Web server retourné contient les valeurs courantes des propriétés du serveur Web.

```

// Méthode d'un composant
var $hostWS : 4D.WebServer
$hostWS:=WEB Server(Web server host database)
If($hostWS.isRunning)
  ...
End if

```

WEB Server list

► Historique

WEB Server list : Collection

Paramètres	Type		Description
Résultat	Collection	<-	Collection des objets Web server disponibles

La fonction `.stop()` arrête le serveur Web auquel elle est appliquée.

La commande `WEB Server list` retourne une collection de tous les objets Web server disponibles dans l'application 4D.

- un serveur Web pour la base de données hôte (serveur Web par défaut)
- un serveur Web pour chaque composant.

Une application 4D peut contenir de 1 à N serveurs Web :

L'objet serveur Web par défaut est automatiquement chargé par 4D au démarrage. En revanche, chaque composant serveur Web que vous souhaitez utiliser doit être instancié à l'aide de la commande `WEB Server`.

La commande `WEB Server list` retourne tous les serveurs Web disponibles, qu'ils soient en cours d'exécution ou non.

Exemple

Vous pouvez utiliser la propriété `.name` de l'objet Web server pour identifier le projet ou le composant auquel chaque objet Web server de la liste appartient.

```
var $wSList : Collection
var $vRun : Integer

$wSList:=WEB Server list
$vRun:=$wSList.countValues(True;"isRunning")
ALERT(String($vRun)+" web server(s) running on "+String($wSList.length)+" available.")
```

.accessKeyDefined

`.accessKeyDefined` : Boolean

La propriété `.accessKeyDefined` contient vrai si une access key est définie dans les settings du serveur Web. Cette propriété est utilisée par le serveur web WebAdmin pour valider la configuration de sécurité de l'interface d'administration.

.certificateFolder

`.certificateFolder` : Text

Chemin du dossier contenant les fichiers des certificats. Chemin d'accès complet au format POSIX utilisant des filesystems. Lorsque cette propriété est utilisée dans le paramètre `settings` de la fonction `.start()`, il peut s'agir d'un objet `Folder`.

.characterSet

`.characterSet` : Number

`.characterSet` : Text

Jeu de caractères que le serveur Web doit utiliser pour communiquer avec les navigateurs se connectant à l'application. La valeur par défaut dépend de la langue du système d'exploitation. Peut être un numéro MIBEnum ou un nom (chaîne), identifiants [définis par l'IANA](#). Voici la liste des identifiants correspondant aux jeux de caractères pris en charge par le serveur Web de 4D :

- 4 = ISO-8859-1

- 12 = ISO-8859-9
- 13 = ISO-8859-10
- 17 = Shift-JIS
- 2024 = Windows-31J
- 2026 = Big5
- 38 = euc-kr
- 106 = UTF-8
- 2250 = Windows-1250
- 2251 = Windows-1251
- 2253 = Windows-1253
- 2255 = Windows-1255
- 2256 = Windows-1256

.cipherSuite

.cipherSuite : Text

Liste de chiffrement utilisée pour le protocole sécurisé. Définit la priorité des algorithmes de chiffrement implémentés par le serveur Web de 4D. Peut être une séquence de chaînes séparées par des deux-points (par exemple "ECDHE-RSA-AES128 -..."). Voir la [page des chiffrements](#) sur le site OpenSSL.

.CORSEnabled

.CORSEnabled : Boolean

Statut du service CORS (*Cross-origin resource sharing*) pour le serveur Web. Pour des raisons de sécurité, les requêtes "cross-domain" sont interdites par défaut au niveau du navigateur. Pour des raisons de sécurité, les requêtes "cross-domain" sont interdites par défaut au niveau du navigateur. Lorsqu'il est activé (True), les appels XHR (par exemple les requêtes REST) à partir de pages Web hors du domaine peuvent être autorisés dans votre application (vous devez définir la liste des adresses autorisées dans la liste des domaines CORS, voir `CORSSettings` ci-dessous). Lorsqu'il est désactivé (False, par défaut), toutes les requêtes entre sites (cross site) envoyées avec CORS sont ignorées.

Par défaut : False (désactivé)

Par défaut : False (désactivé)

.CORSSettings

.CORSSettings : Collection

Liste d'hôtes et de méthodes autorisées pour le service CORS (voir la propriété `CORSEnabled`). Chaque objet doit contenir une propriété host et, optionnellement, une propriété methods :

- host (texte, obligatoire) : nom de domaine ou adresse IP à partir duquel les pages externes sont autorisées à envoyer des requêtes de données au serveur via CORS. Plusieurs attributs de domaine peuvent être ajoutés pour créer une liste blanche. Si `host` n'est pas présent ou vide, l'objet est ignoré. Plusieurs syntaxes sont supportées :
 - 192.168.5.17:8081
 - 192.168.5.17
 - 192.168.*
 - 192.168.*:8081
 - <http://192.168.5.17:8081>
 - http://*.myDomain.com
 - <http://myProject.myDomain.com>
 - *.myDomain.com
 - myProject.myDomain.com
 - *
- methods (texte, facultatif) : méthode(s) HTTP acceptée(s) pour l'hôte CORS correspondant. Séparez chaque

méthode par un ";" (ex : "post;get"). Si *methods* est vide, null ou non défini, toutes les méthodes sont activées.

.debugLog

.debugLog : Number

Statut du fichier de log des requêtes HTTP (HTTPDebugLog_nn.txt, stocké dans le dossier "Logs" de l'application -- nn est le numéro de fichier).

- 0 = désactivé
- 1 = activé sans les parties du corps (la taille du corps est fournie dans ce cas)
- 3 = activé avec les parties du corps en réponse uniquement
- 5 = activé avec des parties du corps sur requête uniquement
- 7 = activé avec des parties du corps en réponse et requête

.defaultHomepage

.defaultHomepage : Text

Nom de la page home par défaut ou "" pour ne pas envoyer de page home personnalisée.

.HSTSEnabled

.HSTSEnabled : Boolean

Statut du HTTP Strict Transport Security (HSTS). HSTS permet au serveur Web de déclarer que les navigateurs doivent interagir avec uniquement via des connexions HTTPS sécurisées. Les navigateurs enregistreront les informations HSTS la première fois qu'ils recevront une réponse du serveur Web, puis toutes les futures requêtes HTTP seront automatiquement transformées en requêtes HTTPS. La durée de stockage de ces informations par le navigateur est indiquée avec la propriété `HSTSMaxAge`. HSTS nécessite l'activation de HTTPS sur le serveur. HTTP doit également être activé pour permettre des connexions client initiales.

.HSTSMaxAge

.HSTSMaxAge : Number

Durée maximum (en secondes) pendant laquelle HSTS est actif pour chaque nouvelle connexion cliente. Ces informations sont stockées côté client pendant la durée spécifiée.

Valeur par défaut : 63072000 (2 ans).

.HTTPCompressionLevel

.HTTPCompressionLevel : Number

Niveau de compression pour tous les échanges HTTP compressés du serveur (requêtes clientes ou réponses du serveur). Ce sélecteur vous permet d'optimiser les échanges en priorisant soit la vitesse d'exécution (moins de compression), soit la quantité de compression (moins de vitesse).

Valeurs possibles :

- 1 à 9 (où 1 correspond à la compression la plus rapide et 9 la plus élevée).
- -1 = définir un compromis entre la vitesse et le taux de compression.

Valeurs possibles :

.HTTPCompressionThreshold

.HTTPCompressionThreshold : Number

Limite de la taille des requêtes (en octets) au-dessous de laquelle les échanges ne doivent pas être compressés. Ce paramètre est utile pour éviter de perdre du temps machine en compressant les petits échanges.

Seuil de compression par défaut = 1024 octets

.HTTPEnabled

.HTTPEnabled : Boolean

Statut du protocole HTTP.

.HTTPPort

.HTTPPort : Number

Statut du protocole HTTPS.

Numéro de port IP d'écoute pour HTTP.

.HTTPTrace

.HTTPTrace : Boolean

Activation de `HTTP TRACE`. Pour des raisons de sécurité, le serveur Web rejette par défaut les requêtes `HTTP TRACE` avec une erreur 405. Lorsque le `HTTP TRACE` est activé, le serveur Web répond aux requêtes `HTTP TRACE` avec la ligne, l'en-tête et le corps de la requête.

.HTTPSEnabled

.HTTPSEnabled : Boolean *Propriété en lecture seulement.* Statut démarré du serveur Web.

.HTTPSPort

.HTTPSPort : Number Numéro de port IP d'écoute pour HTTPS.

Numéro de port IP d'écoute pour HTTPS.

.inactiveProcessTimeout

.inactiveProcessTimeout : Number

Cette propriété n'est pas retournée [en mode sessions évolutives](#).

Cette propriété n'est pas retournée [en mode sessions évolutives](#). Durée de vie (en minutes) des sessions legacy inactives. À la fin de cette période, le cookie de session expire et n'est plus envoyé par le client HTTP.

Par défaut = 480 minutes

.inactiveSessionTimeout

.inactiveSessionTimeout : Number

Cette propriété n'est pas retournée [en mode sessions évolutives](#).

Durée de vie (en minutes) des process de session legacy inactifs. À la fin de cette période, le cookie de session expire et n'est plus envoyé par le client HTTP.

Par défaut = 480 minutes

.IPAddressToListen

.IPAddressToListen : Text

Adresse IP sur laquelle le serveur Web recevra les requêtes HTTP. Par défaut, aucune adresse spécifique n'est définie. Les formats de chaîne IPv6 et IPv4 sont pris en charge.

.isRunning

.isRunning : Boolean

Propriété en lecture seulement.

Vrai si les sessions legacy sont activées dans le serveur Web, sinon Faux.

.keepSession

.keepSession : Boolean

Vrai si les sessions évolutives (scalable sessions) sont activées dans le serveur Web, sinon Faux.

Voir aussi:

[.scalableSession](#)

.logRecording

.logRecording : Number

Mode d'enregistrement du log des requêtes (logweb.txt).

- 0 = Ne pas enregistrer (par défaut)
- 1 = Enregistrer au format CLF
- 2 = Enregistrer au format DLF
- 3 = Enregistrer au format ELF
- 4 = Enregistrer au format WLF

.maxConcurrentProcesses

.maxConcurrentProcesses : Number

Nombre maximum de process web simultanés accepté par le serveur Web. Lorsque ce nombre (moins un) est atteint, 4D ne crée aucun autre process et retourne le statut HTTP 503 - Service Unavailable to all new requests.

Valeurs possibles : 500000 - 2147483648

Par défaut = 80

.maxRequestSize

.maxRequestSize : Number

Taille maximum (en octets) des requêtes HTTP entrantes (POST) que le serveur Web est autorisé à traiter. Passer la valeur maximale (2147483647) signifie qu'en pratique, aucune limite n'est définie. Cette limite est utilisée pour éviter la saturation du serveur Web en raison de requêtes entrantes trop volumineuses. Si une requête atteint cette limite, le serveur Web la rejette.

Valeurs possibles : 500000 - 2147483647

.maxSessions

.maxSessions : Number

Cette propriété n'est pas retournée en mode sessions évolutives.

Nombre maximum de sessions legacy simultanées. Lorsque vous atteignez la limite, la session la plus ancienne est fermée (et la méthode base `On Web Legacy Close Session` est appelée) si le serveur Web doit en créer une nouvelle. Le nombre de sessions legacy simultanées ne peut pas dépasser le nombre total de processus Web (propriété `maxConcurrentProcesses`, 100 par défaut)

.minTLSVersion

.minTLSVersion : Number

Version minimum de TLS acceptée pour les connexions. Les tentatives de connexion de clients prenant en charge uniquement les versions inférieures au minimum seront rejetées.

Valeurs possibles :

- 1 = TLSv1_0
- 2 = TLSv1_1
- 3 = TLSv1_2 (par défaut)
- 4 = TLSv1_3

En cas de modification, le serveur doit être redémarré pour utiliser la nouvelle valeur.

.name

.name : Text

Propriété en lecture seulement.

Propriété en lecture seulement. Disponibilité du PFS sur le serveur.

.openSSLVersion

.openSSLVersion : Text

Propriété en lecture seulement.

Propriété en lecture seulement. Version de la librairie OpenSSL utilisée.

.perfectForwardSecrecy

.perfectForwardSecrecy : Boolean

Propriété en lecture seulement.

La fonction `.start()` fonction démarre le serveur Web server auquel elle est appliquée, en utilisant les propriétés définies dans le paramètre optionnel `settings`.

.rootFolder

.rootFolder : Text

Chemin du dossier racine du serveur Web. Chemin d'accès complet au format POSIX utilisant des filesystems. Peut être passé comme objet `Folder` dans le paramètre `settings`.

.scalableSession

.scalableSession : Boolean

Vrai si les sessions évolutives (scalable sessions) sont activées dans le serveur Web, sinon Faux.

Voir aussi:

[.keepSession](#)

.sessionCookieDomain

.sessionCookieDomain : Text

Champ "domain" du cookie de session. Utilisé pour contrôler la portée des cookies de session. Par exemple, si vous définissez la valeur "/4DACTION" pour ce sélecteur, le client enverra un cookie uniquement pour les requêtes dynamiques commençant par 4DACTION, et non pour les images, les pages statiques, etc.

.sessionCookieName

.sessionCookieName : Text

Nom du cookie utilisé pour stocker l'ID de session.

Propriété en lecture seulement.

.sessionCookiePath

.sessionCookiePath : Text

Champ "path" du cookie de session. Utilisé pour contrôler la portée des cookies de session. Par exemple, si vous définissez la valeur "/4DACTION" pour ce sélecteur, le client enverra un cookie uniquement pour les requêtes dynamiques commençant par 4DACTION, et non pour les images, les pages statiques, etc.

.sessionCookieSameSite

► Historique

.sessionCookieSameSite : Text

Valeur "SameSite" du cookie de session. Valeurs possibles (avec constantes):

Constante	Valeur	Description
Web SameSite Strict	"Strict"	<i>Valeur par défaut</i> - Les cookies sont envoyés uniquement dans un contexte interne (first-party)
Web SameSite Lax	"Lax"	Les cookies sont également envoyés aux sous-requêtes intersites mais uniquement lorsque l'internaute navigue vers le site d'origine (i.e. en suivant un lien).
Web SameSite None	"None"	Les cookies sont envoyés dans tous les contextes, i.e. en réponse aux requêtes internes (first-party) et aux requêtes cross-origin.

Tous les paramètres des [objets Web Server](#) peuvent être personnalisés, hormis les propriétés en lecture seule ([.isRunning](#), [.name](#), [.openSSLVersion](#), [.perfectForwardSecrecy](#), et [.sessionCookieName](#)).

.sessionIPAddressValidation

.sessionIPAddressValidation : Boolean

Cette propriété n'est pas utilisée dans le [mode de sessions évolutives](#) (il n'existe pas de validation d'adresse IP).

Validation de l'adresse IP des cookies de session. Pour des raisons de sécurité, le serveur Web vérifie par défaut l'adresse IP de chaque requête contenant un cookie de session et la rejette si cette adresse ne correspond pas à l'adresse IP utilisée pour créer le cookie. Dans certaines applications spécifiques, vous souhaiterez peut-être désactiver cette validation et accepter les cookies de session, même lorsque leurs adresses IP ne correspondent pas. Par exemple, lorsque les appareils mobiles basculent entre les réseaux Wifi et 3G/4G, leur adresse IP change. Dans ce cas, vous pouvez permettre aux clients de continuer à utiliser leurs sessions Web même lorsque les adresses IP changent (ce paramétrage abaisse le niveau de sécurité de votre application).

.start()

► Historique

.start() : Object

.start(*settings* : Object) : Object

Paramètres	Type		Description
settings	Object	->	Paramètres du serveur web au démarrage
Résultat	Object	<-	État du démarrage du serveur web

le serveur Web par défaut. Pour désigner le serveur Web à retourner, vous pouvez passer une des constantes suivantes dans le paramètre *option* :

Le serveur Web démarre avec les paramètres par défaut définis dans le fichier de settings du projet ou (base hôte uniquement) à l'aide de la commande `WEB SET OPTION`. Cependant, à l'aide du paramètre *settings*, vous pouvez définir des paramètres personnalisés pour la session du serveur Web.

Tous les paramètres des [objets Web Server](#) peuvent être personnalisés, hormis les propriétés en lecture seule (`.isRunning`, `.name`, `.openSSLVersion`, `.perfectForwardSecrecy`, et `.sessionCookieName`).

Pour arrêter le serveur Web de la base :

Objet retourné

La fonction retourne un objet décrivant le statut démarré du serveur Web. Cet objet peut avoir les propriétés suivantes :

Propriété		Type	Description
success		Boolean	Vrai si le serveur web a été correctement démarré, sinon Faux
errors		Collection	Pile d'erreurs 4D (non retournée si le serveur web a démarré avec succès)
	[].errCode	Number	Code d'erreur 4D
	[].message	Text	Description de l'erreur 4D
	[].componentSignature	Text	Signature du composant interne qui a retourné l'erreur

Si le serveur Web a déjà été lancé, une erreur est générée.

Exemple

```

var $settings;$result : Object
var $webServer : 4D.WebServer

$settings:=New object("HTTPPort";8080;"defaultHomepage";"myAdminHomepage.html")

$webServer:=WEB Server
$result:=$webServer.start($settings)
If($result.success)
//...
End if

```

.stop()

► Historique

.stop()

Paramètres	Type	Description	-----	----	-----		Ne requiert aucun paramètre

La fonction `.stop()` arrête le serveur Web auquel elle est appliquée.

Si le serveur Web était lancé, toutes les connexions Web et tous les process Web sont fermés une fois que les requêtes actuellement traitées sont terminées. Si le serveur Web n'était pas démarré, la fonction ne fait rien.

Cette fonction réinitialise les paramètres Web personnalisés définis pour la session à l'aide du paramètre `settings` de la fonction `.start()`, le cas échéant.

Exemple

Pour arrêter le serveur Web de la base :

```

var $webServer : 4D.WebServer

$webServer:=WEB Server(Web server database)
$webServer.stop()

```

ZIPArchive

Une archive ZIP 4D est un objet `File` ou `Folder` contenant un ou plusieurs fichiers ou dossiers, qui sont compressés afin d'être plus petits que leur taille d'origine. Ces archives sont créées avec une extension ".zip" et peuvent être utilisées pour économiser de l'espace sur le disque ou transférer des fichiers sur des supports de taille limitée (par exemple, l'email ou le réseau).

- Créez une archive ZIP 4D avec la commande [ZIP Create archive](#).
- Les instances `ZIPFile` et `ZIPFolder` de 4D sont disponibles via la propriété `root` (`ZIPFolder`) de l'objet retourné par la commande [ZIP Read archive](#).

Exemple

Pour récupérer et visualiser le contenu d'un objet ZIP file :

```
var $path; $archive : 4D.File
var $zipFile : 4D.ZipFile
var $zipFolder : 4D.ZipFolder
var $txt : Text

$path:=Folder(fk desktop folder).file("MyDocs/Archive.zip")
$archive:=ZIP Read archive($path)
$zipFolder:=$archive.root // stocker le dossier principal du zip
$zipFile:=$zipFolder.files()[0] //lire le premier fichier zippé

If($zipFile.extension=".txt")
    $txt:=$zipFile.getText()
End if
```

Sommaire

.root : 4D.ZipFolder Un dossier virtuel permettant d'accéder au contenu de l'archive ZIP

ZIP Create archive

► Historique

`ZIP Create archive (fileToZip : 4D.File ; destinationFile : 4D.File) : Object`

`ZIP Create archive (folderToZip : 4D.Folder ; destinationFile : 4D.File { ; options : Integer }) : Object`

`ZIP Create archive (zipStructure : Object ; destinationFile : 4D.File) : Object`

Paramètres	Type		Description
fileToZip	4D.File	->	Objet fichier ou dossier à compresser
folderToZip	4D.Folder	->	Objet fichier ou dossier à compresser
zipStructure	Object	->	Objet fichier ou dossier à compresser
destinationFile	4D.File	->	Fichier de destination de l'archive
options	Integer	->	Si <code>folderToZip</code> utilisé : <code>ZIP Without enclosing folder</code>
Résultat	Object	<-	Objet statut

Description

La commande `ZIP Create archive` crée un objet archive ZIP compressé et retourne le statut de l'opération.

Vous pouvez passer un objet `4D.File`, `4D.Folder`, ou une structure Zip en tant que premier paramètre :

- `fileToZip` : passez simplement un `4D.File` à compresser.
- `folderToZip` : passez un `4D.Folder` à compresser. Dans ce cas, le paramètre `options` vous permet de compresser uniquement le contenu du dossier (c'est-à-dire d'exclure le dossier parent). Par défaut, l'archive `ZIP Create archive` compressera le dossier et son contenu, de sorte que l'opération de décompression recrée un dossier. Si vous souhaitez que l'opération de décompression ne restaure que le contenu du dossier, passez la constante `ZIP Without enclosing folder` dans le paramètre `options`.
- `zipStructure` : passez un objet décrivant l'objet ZIP archive. Les propriétés suivantes sont disponibles pour définir la structure :

Propriété	Type	Description												
compression	Integer	<ul style="list-style-type: none"> • <code>ZIP Compression standard</code> : Réduire la compression (par défaut) • <code>ZIP Compression LZMA</code> : compression LZMA • <code>ZIP Compression XZ</code> : compression XZ • <code>ZIP Compression none</code> : Pas de compression 												
level	Integer	<p>Niveau de compression. Valeurs possibles : 1 à 10. Une valeur plus faible produira un fichier plus volumineux, tandis qu'une valeur plus élevée produira un fichier plus petit. Le niveau de compression a toutefois un impact sur les performances. Valeur par défaut (si omis) :</p> <ul style="list-style-type: none"> • <code>ZIP Compression standard</code> : 6 • <code>ZIP Compression LZMA</code> : 4 • <code>ZIP Compression XZ</code> : 4 												
encryption	Integer	<p>Le chiffrement à utiliser si un mot de passe est défini :</p> <ul style="list-style-type: none"> • <code>ZIP Encryption AES128</code> : chiffrement AES à l'aide d'une clé 128 octets. • <code>ZIP Encryption AES192</code> : chiffrement AES à l'aide d'une clé 192 octets. • <code>ZIP Encryption AES256</code> : chiffrement AES à l'aide d'une clé 256 octets (par défaut si un mot de passe est défini). • <code>ZIP Encryption none</code> : les données ne sont pas chiffrées (par défaut si aucun mot de passe n'est défini) 												
password	Text	Un mot de passe à définir si le chiffrement est requis.												
Historique	Collection	<ul style="list-style-type: none"> • une collection d'objets <code>4D.File</code> ou <code>4D.Folder</code> ou • une collection d'objets dont les propriétés sont les suivantes : <table border="1"> <thead> <tr> <th>Propriété</th><th>Type</th><th>Description</th></tr> </thead> <tbody> <tr> <td>source</td><td><code>4D.File</code> ou <code>4D.Folder</code></td><td>File ou Folder</td></tr> <tr> <td>destination</td><td>Text</td><td>(facultatif) - Indiquer un chemin de fichier relatif pour modifier l'organisation du contenu de l'archive</td></tr> <tr> <td>option</td><td>number</td><td>(facultatif) - <code>ZIP Ignore invisible files</code> ou 0 pour compresser tout le fichier</td></tr> </tbody> </table>	Propriété	Type	Description	source	<code>4D.File</code> ou <code>4D.Folder</code>	File ou Folder	destination	Text	(facultatif) - Indiquer un chemin de fichier relatif pour modifier l'organisation du contenu de l'archive	option	number	(facultatif) - <code>ZIP Ignore invisible files</code> ou 0 pour compresser tout le fichier
Propriété	Type	Description												
source	<code>4D.File</code> ou <code>4D.Folder</code>	File ou Folder												
destination	Text	(facultatif) - Indiquer un chemin de fichier relatif pour modifier l'organisation du contenu de l'archive												
option	number	(facultatif) - <code>ZIP Ignore invisible files</code> ou 0 pour compresser tout le fichier												
callback	4D.Function	Une formule de rétro-appel qui recevra la progression de la compression (0 à 100) dans \$1.												

Dans le paramètre `destinationFile`, passez un objet `4D.File` décrivant l'archive ZIP à créer (nom, emplacement, etc.). Il est conseillé d'utiliser l'extension ".zip" si vous souhaitez que l'archive ZIP soit traitée automatiquement par un logiciel.

Une fois que l'archive est créée, vous pouvez utiliser la commande [ZIP Read archive](#) pour y accéder.

Objet statut

L'objet statut retourné contient les propriétés suivantes :

Propriété	Type	Description
statusText	Text	Message d'erreur (le cas échéant) : <ul style="list-style-type: none">Impossible d'ouvrir l'archive ZIPImpossible de créer une archive ZIPLe mot de passe est requis pour le chiffrement
status	Integer	Code d'état
success	Boolean	Vrai si l'archive a été créée avec succès, sinon faux

Exemple 1

Pour compresser un `4D.File` :

```
var $file; $destination : 4D.File
var $status : Object

$destination:=Folder(fk desktop folder).file("MyDocs/file.zip")
$file:=Folder(fk desktop folder).file("MyDocs/text.txt")

$status:=ZIP Create archive($file;$destination)
```

Exemple 2

Pour compresser un `4D.Folder` sans le dossier lui-même :

```
var $folder : 4D.Folder
var $destination : 4D.File
var $status : Object

$destination:=Folder(fk desktop folder).file("MyDocs/Images.zip")
$folder:=Folder(fk desktop folder).folder("MyDocs/Images")

$status:=ZIP Create archive($folder;$destination;ZIP Without enclosing folder)
```

Exemple 3

Pour compresser une structure d'archive ZIP avec un mot de passe et une barre de progression :

```

var $destination : 4D.File
var $zip;$status : Object
var progID : Integer

$destination:=Folder(fk desktop folder).file("MyDocs/Archive.zip")

$zip:=New object
$zip.files:=Folder(fk desktop folder).folder("MyDocs/Resources").folders()
$zip.password:="password"
$zip.callback:=Formula(myFormulaCompressingMethod($1))

progID:=Progress New //nous utilisons le composant 4D Progress

$status:=ZIP Create archive($zip;$destination)

Progress QUIT(progID)

```

myFormulaCompressingMethod :

```

var $1 : Integer
Progress SET PROGRESS(progID;Num($1/100))

```

Exemple 4

Vous souhaitez passer une collection de dossiers et de fichiers à compresser à l'objet `zipStructure` :

```

var $destination : 4D.File
var $zip;$err : Object
$zip:=New object
$zip.files:=New collection
$zip.files.push(New object("source";Folder(fk desktop folder).file("Tests/text.txt")))
$zip.files.push(New object("source";Folder(fk desktop folder).file("Tests/text2.txt")))
$zip.files.push(New object("source";Folder(fk desktop folder).file("Images/image.png")))

$destination:=Folder(fk desktop folder).file("file.zip")
$err:=ZIP Create archive($zip;$destination)

```

Exemple 5

Vous souhaitez utiliser un autre algorithme de compression à un niveau de compression élevé :

```

var $destination : 4D.File
var $zip; $err : Object

$zip:=New object
$zip.files:=New collection
$zip.files.push(Folder(fk desktop folder).folder("images"))
$zip.compression:=ZIP Compression LZMA
$zip.level:=7 //4 par défaut

$destination:=Folder(fk desktop folder).file("images.zip")
$err:=ZIP Create archive($zip; $destination)

```

ZIP Read archive

► Historique

ZIP Read archive (`zipFile` : 4D.File { ; `password` : Text }) : 4D.ZipArchive

Paramètres	Type		Description
zipFile	4D.File	->	Fichier archive ZIP
password	Text	->	Mot de passe de l'archive ZIP, le cas échéant
Résultat	4D.ZipArchive	<-	Objet archive

Description

La commande `ZIP Read archive` récupère le contenu du `zipFile` et le retourne sous forme d'objet `4D.ZipArchive`.

Cette commande ne décomprime pas l'archive ZIP, elle fournit seulement un aperçu de son contenu. Pour extraire le contenu d'une archive, vous devez utiliser des méthodes telles que `file.copyTo()` ou `folder.copyTo()`.

Passez un objet `4D.File` référençant l'archive ZIP compressée dans le paramètre `zipFile`. Le fichier d'archive cible est ouvert jusqu'à la fin de l'exécution de la commande `ZIP Read archive` et jusqu'à ce que tous les contenus/références soient extrait(e)s/publié(e)s ; il est ensuite fermé automatiquement.

Si le `zipFile` est protégé par un mot de passe, vous devez utiliser le paramètre `password` pour fournir un mot de passe. Si un mot de passe est requis mais qu'il n'est pas passé lorsque vous tentez de lire le contenu de l'archive, une erreur est générée.

Objet archive

L'objet `4D.ZipArchive` retourné contient une seule propriété, `root`, dont la valeur est un objet `4D.ZipFolder`. Ce dossier décrit le contenu de l'archive ZIP.

Exemple

Pour récupérer et visualiser le contenu d'un objet ZIP file :

```
var $archive : 4D.ZipArchive
var $path : 4D.File

$path:=Folder(fk desktop folder).file("MyDocs/Archive.zip")
$archive:=ZIP Read archive($path)
```

Pour récupérer la liste des fichiers et dossiers de l'archive :

```
$folders:=$archive.root.folders()
$files:=$archive.root.files()
```

Pour lire le contenu d'un fichier sans l'extraire du dossier root :

```
If($files[$i].extension=".txt")
  $txt:=$files[$i].getText()
Else
  $blob:=$files[$i].getContent()
End if
```

Pour extraire à partir du dossier root :

```
//extract a file  
$folderResult:=$files[$i].copyTo(Folder(fk desktop folder).folder("MyDocs"))  
  
//extract all files  
$folderResult:=$archive.root.copyTo(Folder(fk desktop folder).folder("MyDocs"))
```

.root

.root : 4D.ZipFolder

Description

La propriété `.root` contient Un dossier virtuel permettant d'accéder au contenu de l'archive ZIP.

Le dossier `root` et son contenu sont manipulés à l'aide des fonctions et propriétés des classes [ZipFile](#) et [ZipFolder](#).

Cette propriété est en lecture seule.

ZIPFile

Les propriétés et fonctions suivantes de la classe [File](#) sont disponibles pour les objets `ZIPFile` :

API disponibles de File API des ZIPFile	Commentaire
<code>.copyTo(destinationFolder : 4D.Folder { ; newName : Text } { ; overwrite : Integer }) : 4D.File</code>	
<code>.creationDate : Date</code>	
<code>.creationTime : Time</code>	
<code>.exists : Boolean</code>	
<code>.extension : Text</code>	
<code>.fullName : Text</code>	
<code>.getContent() : 4D.Blob</code>	
<code>.getIcon({ size : Integer }) : Picture</code>	
<code>.getText({ charSetName : Text { ; breakMode : Integer } }) : Text</code> <code>.getText({ charSetNum : Integer { ; breakMode : Integer } }) : Text</code>	
<code>.hidden : Boolean</code>	
<code>.isAlias : Boolean</code>	
<code>.isFile : Boolean</code>	
<code>.isFolder : Boolean</code>	
<code>.isWritable : Boolean</code>	Toujours Faux avec ZIP archive
<code>.modificationDate : Date</code>	
<code>.modificationTime : Time</code>	
<code>.name : Text</code>	
<code>.original : 4D.File</code> <code>.original : 4D.Folder</code>	
<code>.parent : 4D.Folder</code>	
<code>.path : Text</code>	Retourne un chemin relatif à l'archive
<code>.platformPath : Text</code>	

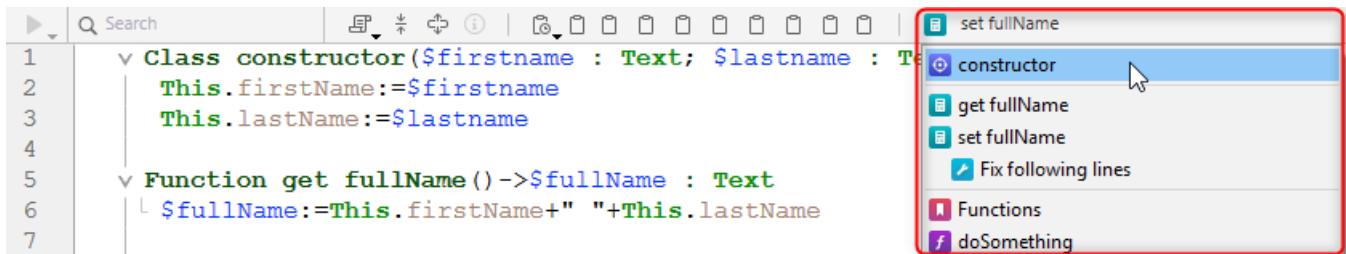
ZIPFolder

Les propriétés et fonctions suivantes de la classe [Folder](#) sont disponibles pour les objets [ZIPFolder](#) :

API disponibles de Folder API des ZIPFolder	Commentaire
<code>.copyTo(destinationFolder : 4D.Folder { ; newName : Text } { ; overwrite : Integer }) : 4D Folder</code>	
<code>.creationDate : Date</code>	La date du dossier <code>racine</code> peut être différente de celle du dossier de l'archive.
<code>.creationTime : Time</code>	L'heure du dossier <code>racine</code> peut être différente de celle du dossier de l'archive.
<code>.exists : Boolean</code>	
<code>.extension : Text</code>	
<code>.file(path : Text) : 4D.File</code>	
<code>.files({ options : Integer }) : Collection</code>	
<code>.folder(path : Text) : 4D.Folder</code>	
<code>.folders({ options : Integer }) : Collection</code>	
<code>.fullName : Text</code>	
<code>.getIcon({ size : Integer }) : Picture</code>	
<code>.hidden : Boolean</code>	
<code>.isAlias : Boolean</code>	
<code>.isFile : Boolean</code>	
<code>.isFolder : Boolean</code>	
<code>.isPackage : Boolean</code>	
<code>.modificationDate : Date</code>	La date du dossier <code>racine</code> peut être différente de celle du dossier de l'archive.
<code>.modificationTime : Time</code>	L'heure du dossier <code>racine</code> peut être différente de celle du dossier de l'archive.
<code>.name : Text</code>	
<code>.original : 4D.Folder</code>	
<code>.parent : 4D.Folder</code>	Le dossier <code>racine</code> virtuel de l'archive n'a pas de parent. Cependant, les dossiers de l'archive peuvent avoir un parent autre que la racine.
<code>.path : Text</code>	Retourne un chemin relatif à l'archive
<code>.platformPath : Text</code>	

Navigation dropdown

La liste déroulante de navigation vous aide à organiser votre code et à naviguer plus facilement dans vos classes et méthodes :



Certaines balises sont ajoutées automatiquement ; vous pouvez compléter la liste déroulante à l'aide de [marqueurs](#).

Navigation dans le code

Cliquez sur un élément de la liste déroulante pour accéder à sa première ligne dans le code. Vous pouvez également naviguer avec les touches fléchées et appuyer sur Entrée.

Balisage automatique

Les constructeurs, les déclarations de méthodes, les fonctions et les attributs calculés sont automatiquement balisés et ajoutés à la liste déroulante.

Lorsqu'il n'y a pas de balise dans la classe/méthode, l'outil affiche "No tag".

Les éléments suivants sont ajoutés automatiquement :

Icône	Élément
∅	Pas de balise
🎯	Class constructor ou déclaration de méthode
⌚	Attribut calculé (get, set, orderBy et query)
f	Nom de la fonction de classe

Balisage manuel

En ajoutant des marqueurs dans votre code, vous pouvez ajouter les balises suivantes à la liste déroulante :

Icône	Élément
🔖	MARK : balise
📝	TODO : balise
🔧	FIXME : balise

Vous les déclarez en ajoutant des commentaires tels que :

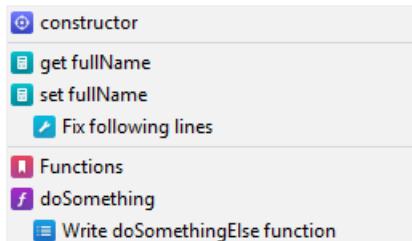
```
// FIXME : Corrige les éléments suivants
```

Les déclarations ne sont pas sensibles à la casse ; écrire `fixme` : a le même effet.

L'ajout d'un trait d'union après la balise `MARK:` trace une ligne de séparation dans l'éditeur de code et dans le menu déroulant. Ainsi, cette saisie :

```
// FIXME: Fix following lines  
  
This.firstName:=Substring($fullName; 1; $p-1)  
This.lastName:=Substring($fullName; $p+1)  
  
//MARK:- Functions []  
  
Function doSomething  
  
// TODO: Write doSomethingElse function
```

Se traduit par ceci :



Tous les marqueurs situés à l'intérieur des fonctions sont indentés dans la liste déroulante, à l'exception des balises `MARK:` situées à la fin des fonctions et non suivies d'instructions. Celles-ci apparaîtront au premier niveau.

Ordre d'affichage

Les balises sont affichées dans leur ordre d'apparition à l'intérieur de la méthode/classe.

Pour afficher les balises d'une méthode ou d'une classe par ordre alphabétique, effectuez l'une des opérations suivantes :

- Faites un clic droit sur l'outil déroulant
- maintenez la touche Cmd sous macOS ou Alt sous Windows, et cliquez sur l'outil de liste déroulante

Les balises à l'intérieur des fonctions se déplacent avec leurs éléments parents.

Principes de base

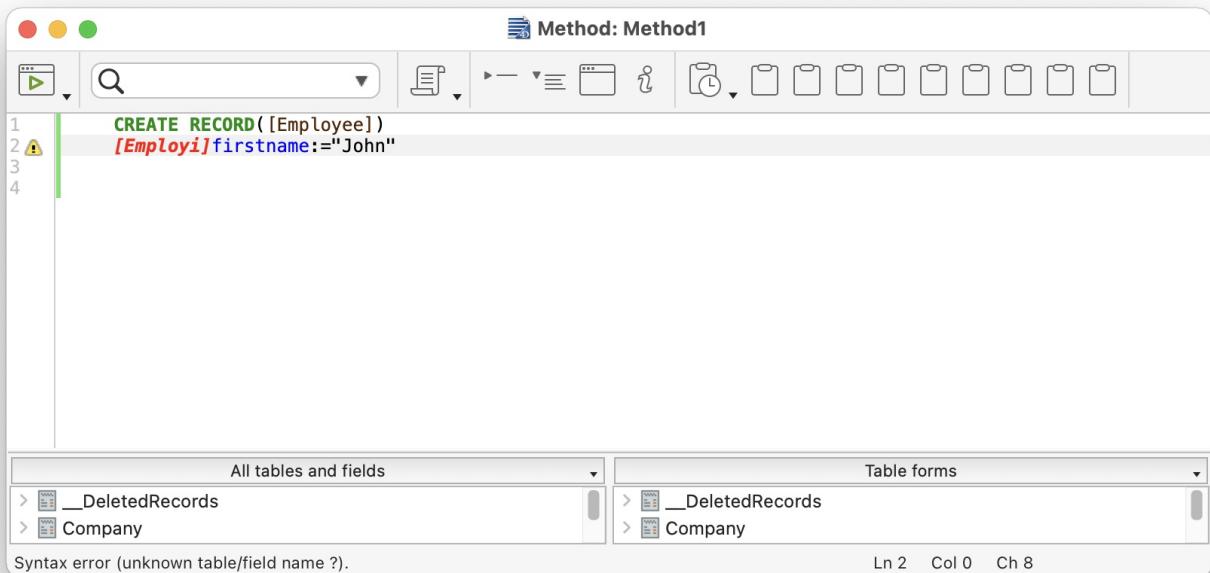
Les erreurs sont courantes. Il serait inhabituel d'écrire un nombre important de lignes de code sans générer aucune erreur. À l'inverse, traiter et/ou corriger des erreurs est également normal !

L'environnement de développement 4D fournit plusieurs outils de débogage pour tous les types d'erreurs.

Types d'erreurs

Erreurs de typage

Les erreurs de frappe sont détectées par l'éditeur de méthode. Elles sont affichées en rouge et des informations complémentaires sont fournies en bas de la fenêtre. Voici une erreur de frappe :



The screenshot shows the 4D Method editor window titled "Method: Method1". The code area contains the following lines:

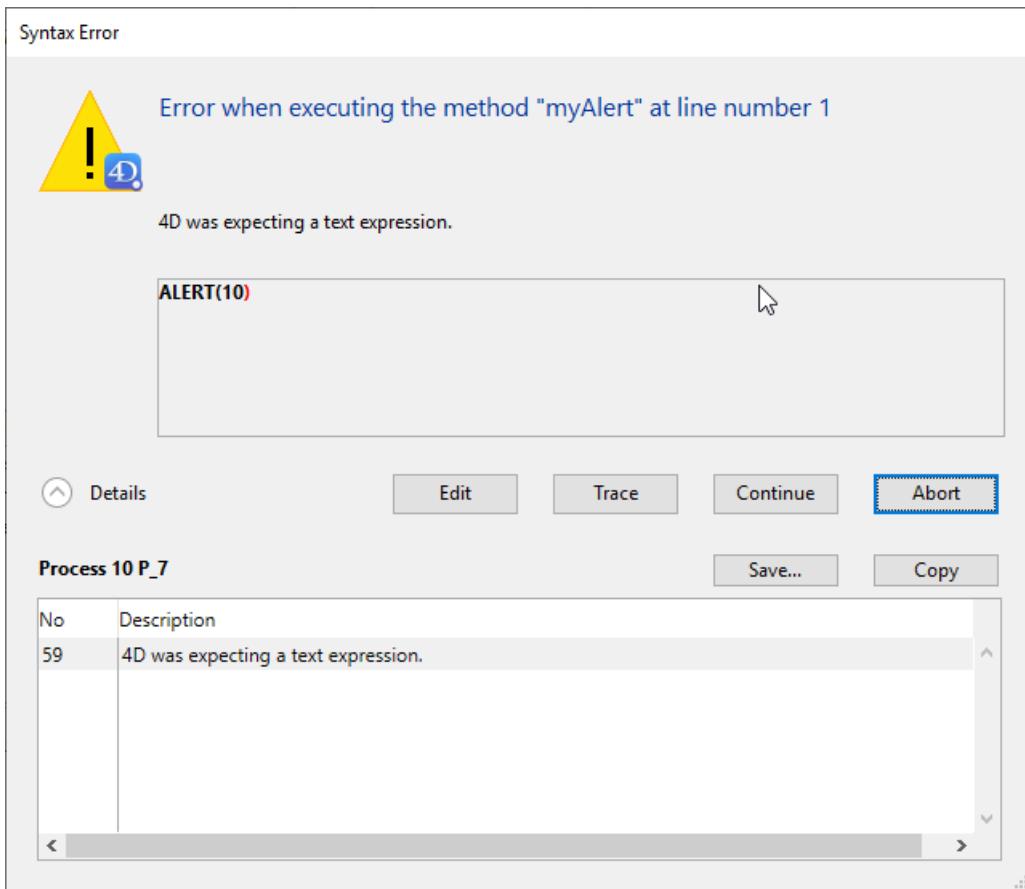
```
1 CREATE RECORD([Employee])
2 [Employi]firstname := "John"
```

Line 2 has a yellow warning icon next to it. The status bar at the bottom left shows "Syntax error (unknown table/field name ?)." and the status bar at the bottom right shows "Ln 2 Col 0 Ch 8".

Ces erreurs de frappe provoquent généralement des erreurs de syntaxe (dans l'image ci-dessus, le nom de la table est inconnu). Vous obtenez la description de l'erreur lorsque vous validez la ligne de code. Lorsque cela se produit, corrigez l'erreur de frappe et tapez Entrée pour valider la correction.

Erreurs de syntaxe

Certaines erreurs ne peuvent être détectées que lorsque vous exécutez la méthode. La [fenêtre d'erreur de syntaxe](#) apparaît lorsqu'une erreur se produit pendant l'exécution du code. Par exemple :



Agrandissez la zone Détails pour afficher la dernière erreur et son numéro.

Erreurs dans l'Environnement

Il peut arriver que la mémoire soit insuffisante pour créer une BLOB. Ou, lorsque vous accédez à un document sur le disque, il se peut que ce document n'existe pas ou qu'il soit déjà ouvert par une autre application. Ces erreurs dans l'Environnement ne sont pas directement dues à votre code ou à la façon dont vous l'avez écrit. La plupart du temps, ces erreurs sont faciles à traiter avec une [méthode de capture d'erreur](#) installée à l'aide de la commande `ON ERR CALL`.

Erreurs de conception ou de logique

Ce sont généralement les types d'erreurs les plus difficiles à trouver. À l'exception des erreurs de frappe, tous les types d'erreurs énumérés ci-dessus sont, dans une certaine mesure, couverts par l'expression "erreur de conception ou de logique". Utilisez le [débogueur](#) pour les détecter. Par exemple :

- Une *erreur de syntaxe* peut se produire lorsque vous essayez d'utiliser une variable qui n'est pas encore initialisée.
- Une *erreur au niveau de l'Environnement* peut se produire lorsque vous essayez d'ouvrir un document, parce que le nom de ce document est reçu par une sous-routine qui n'a pas obtenu la bonne valeur comme paramètre.

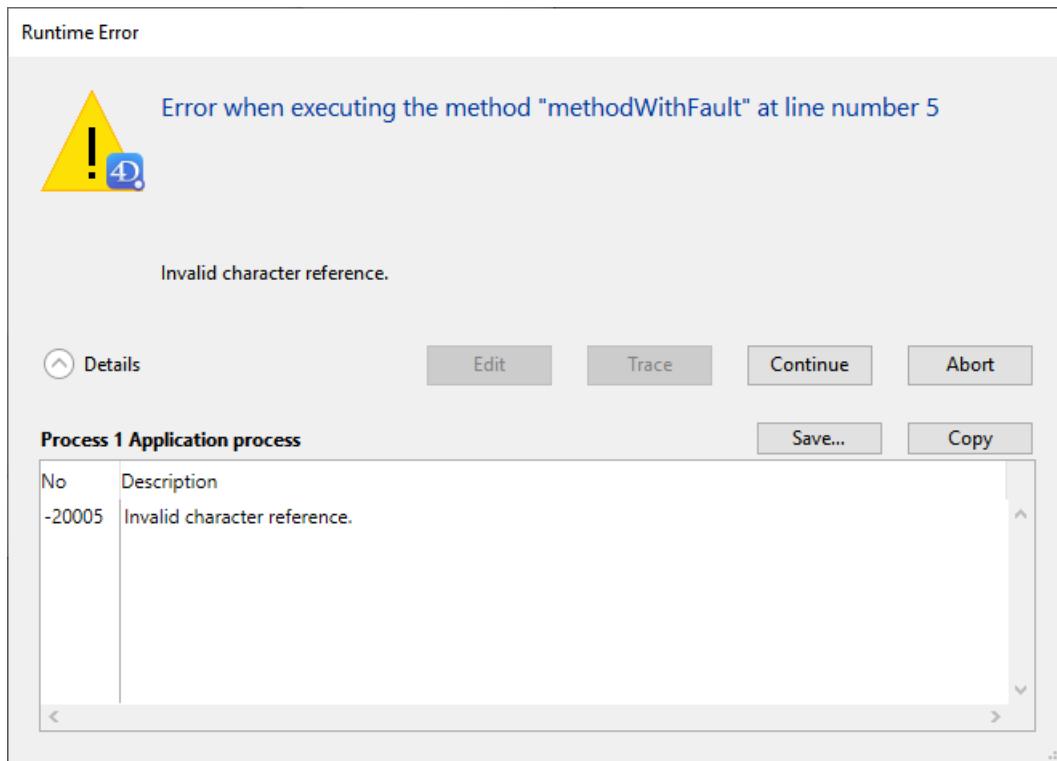
Les erreurs de conception ou de logique comprennent également des situations telles que :

- Un enregistrement n'est pas correctement mis à jour parce que, en appelant `SAVE RECORD`, vous avez oublié de tester d'abord si l'enregistrement était verrouillé ou non.
- Une méthode ne fait pas exactement ce que vous attendez, car la présence d'un paramètre facultatif n'est pas testée.

Parfois, le morceau de code qui affiche l'erreur peut être différent du code qui est en fait à l'origine du problème.

Erreurs d'exécution

En mode Application, vous pouvez obtenir des erreurs que vous ne voyez pas en mode interprété. Voici un exemple :

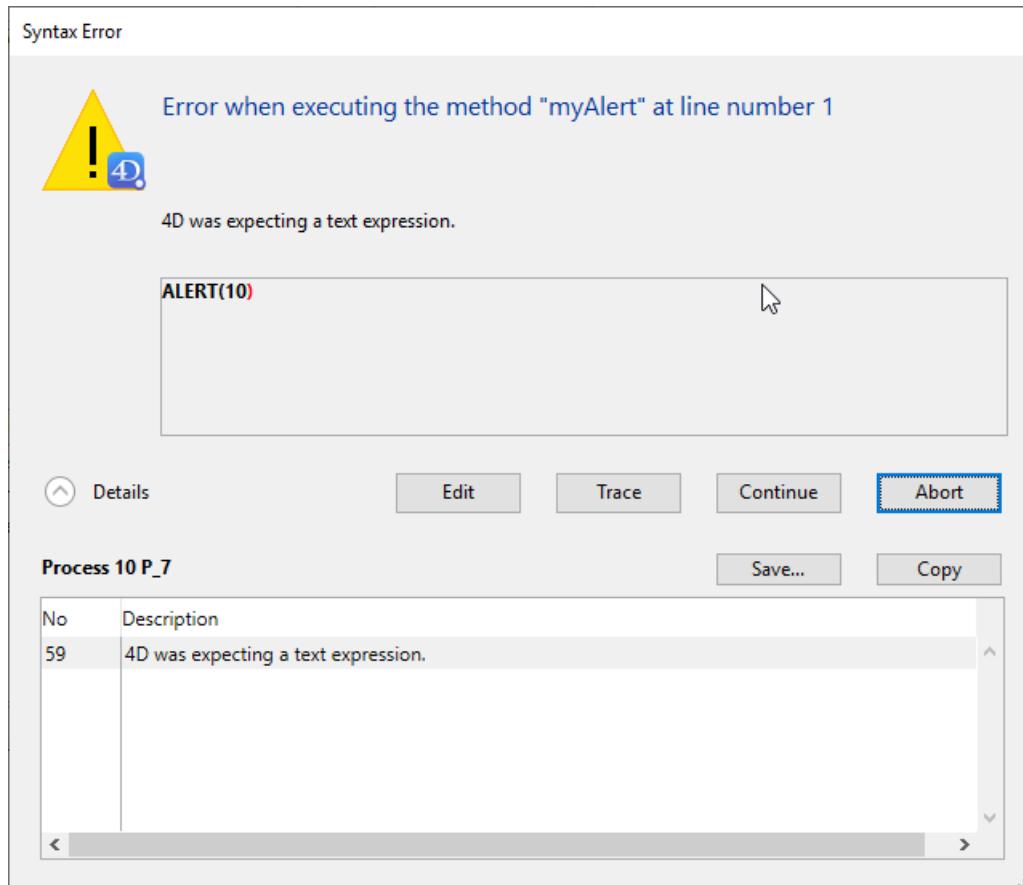


Pour trouver rapidement l'origine du problème, rouvrez la version interprétée du fichier de structure, ouvrez la méthode et allez à la ligne correspondante.

Fenêtre d'erreur de syntaxe

La fenêtre d'erreur de syntaxe apparaît automatiquement lorsque l'exécution d'une méthode est interrompue. Cela peut se produire lorsque :

- une erreur empêche la poursuite de l'exécution du code
- la méthode produit une fausse assertion (voir la commande `ASSERT`)



La zone de texte supérieure affiche un message décrivant l'erreur. La zone de texte inférieure montre la ligne qui était en cours d'exécution lorsque l'erreur s'est produite ; la zone où l'erreur s'est produite est mise en évidence. La section Détails développée contient la "pile" des erreurs liées au processus.

La fenêtre d'erreur de syntaxe propose plusieurs options :

- Modifier : arrête l'exécution de toutes les méthodes. 4D passe à l'environnement Conception et la méthode comportant l'erreur s'ouvre dans l'éditeur de méthode, ce qui vous permet de la corriger. Utilisez cette option lorsque vous reconnaissiez immédiatement l'erreur et que vous pouvez la corriger sans autre forme de procès.
- Trace : Accède au mode Trace/Debugger. La fenêtre du [débogueur](#) s'affiche. Si la ligne en cours n'a été exécutée que partiellement, vous devrez peut-être cliquer plusieurs fois sur le bouton Trace.
- Continuer : L'exécution se poursuit. La ligne contenant l'erreur peut être partiellement exécutée, selon l'endroit où se trouve l'erreur. Continuez avec prudence : l'erreur peut empêcher le reste de votre méthode de s'exécuter correctement. Nous vous recommandons de cliquer sur Continuer uniquement si l'erreur se trouve dans un appel trivial (tel que `SET WINDOW TITLE`) qui n'empêche pas d'exécuter et de tester le reste de votre code.

Astuce : pour ignorer une erreur qui se répète (par exemple, dans les boucles), vous pouvez transformer le bouton Continuer en bouton Ignorer. Maintenez la touche Alt (Windows) ou Option (macOS) enfoncée et cliquez sur le bouton Continuer la première fois qu'il apparaît. L'étiquette du bouton devient Ignorer si la boîte de dialogue est appelée à nouveau pour la même erreur.

- Abandonner : Stoppe l'exécution de la méthode et revient à l'état antérieur au début de l'exécution de la méthode :
 - Si une méthode formulaire ou méthode objet s'exécute en réponse à un événement, elle est arrêtée et vous revenez au formulaire.
 - Si la méthode est exécutée à partir de l'environnement de l'application, vous revenez à cet environnement.
- Copier : Copie les informations de débogage dans le presse-papiers. L'info décrit l'environnement interne de l'erreur (numéro, composant interne, etc.). Elles sont formatées sous forme de texte tabulé.
- Enregistrer... : Enregistre le contenu de la fenêtre d'erreur syntaxique et de la chaîne d'appel dans un fichier `.txt`.

Débogueur

Une erreur courante des débutants en matière de détection d'erreurs est de cliquer sur `Abandon` dans la fenêtre d'erreur de syntaxe, de retourner à l'éditeur de méthode et d'essayer de comprendre comment les erreurs sont détectées. Attention à ne pas faire cela ! Vous économiserez beaucoup de temps et d'énergie en utilisant toujours le Débogueur.

Le débogueur vous permet d'avancer lentement dans les méthodes. Il affiche toutes les informations dont vous avez besoin pour comprendre pourquoi une erreur s'est produite. Une fois que vous avez ces informations, vous savez comment corriger l'erreur.

Une autre raison d'utiliser le débogueur est le développement du code. Il peut arriver que vous écriviez un algorithme plus complexe que d'habitude. Malgré tous les sentiments d'accomplissement, vous ne pouvez pas être totalement sûr que votre codage est 100% correct. Au lieu de l'exécuter "à l'aveugle", vous pouvez utiliser la commande `TRACE` au début de votre code, puis l'exécuter pas à pas pour garder un œil sur ce qui se passe.

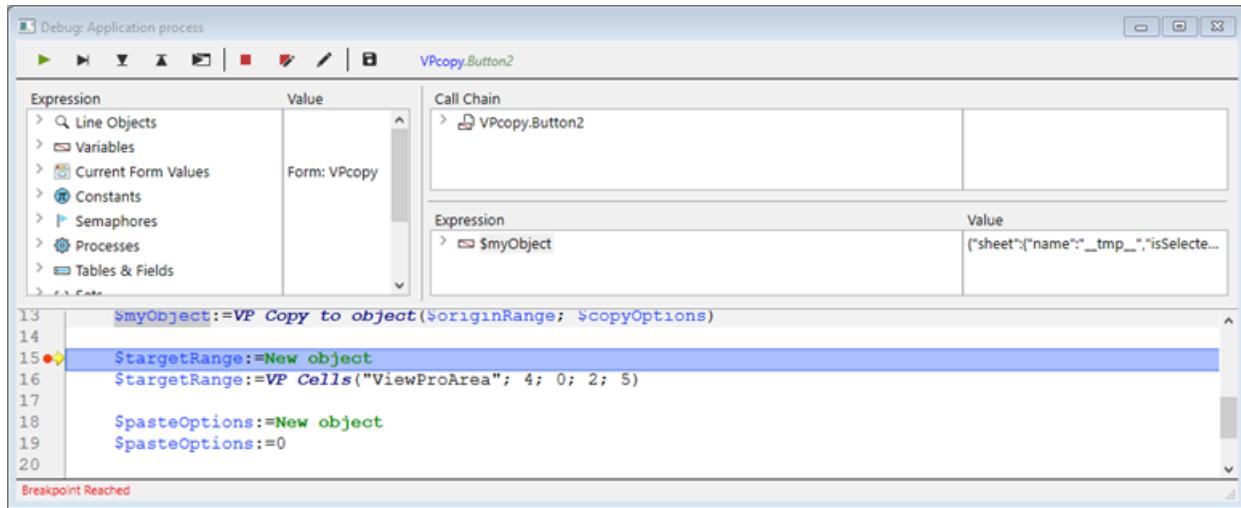
Ruptures/sauts

Dans le processus de débogage, vous pouvez avoir besoin de sauter le traçage de certaines parties du code jusqu'à une certaine ligne. Ou bien, vous pouvez vouloir tracer le code lorsqu'une expression donnée a une certaine valeur (par exemple `"$myVar > 1000"`), ou chaque fois qu'une commande 4D spécifique est appelée.

Ces besoins sont couverts par les points d'arrêt et les fonctions de capture de commande. Ils peuvent être configurés à partir de l'éditeur de méthode, du débogueur ou de l'explorateur de temps d'exécution.

Débogueur

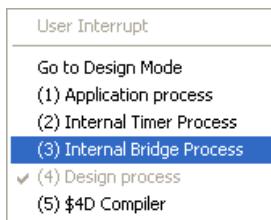
Le débogueur est utile lorsque vous devez repérer des erreurs ou surveiller l'exécution de méthodes. It allows you to step through your code slowly and examine the information. Ce processus est appelé le "traçage".



Appeler le débogueur

Il existe plusieurs façons d'afficher le débogueur :

- En cliquant sur le bouton Trace dans la fenêtre des erreurs de syntaxe
- En utilisant la commande `TRACE`
- En cliquant sur le bouton Debug dans la fenêtre d'exécution de la méthode ou en sélectionnant le bouton Run and debug... dans l'éditeur de méthode
- En utilisant Alt+Shift+Clic droit (Windows) ou Ctrl+Option+Cmd+Clic (macOS) pendant l'exécution d'une méthode, puis en sélectionnant le processus à suivre dans le menu contextuel :



- Cliquez sur le bouton Trace lorsqu'un processus est sélectionné dans la page Process de l'Explorateur d'exécution.
- Ajout d'un point d'arrêt dans la fenêtre de l'éditeur de méthode ou dans les pages Break et Catch de l'explorateur d'exécution.

Lorsqu'il est appelé, la fenêtre du débogueur fournit le nom de la méthode ou de la fonction de classe que vous êtes en train de tracer, ainsi que l'action qui a provoqué l'apparition initiale de la fenêtre du débogueur. Par exemple, dans la fenêtre du débogueur ci-dessus :

- `Clients_BuildLogo` est la méthode en cours de traçage
- La fenêtre du débogueur s'est affichée parce qu'elle a détecté un appel à la commande `C_PICTURE`, qui faisait partie des commandes à identifier

L'affichage d'une nouvelle fenêtre de débogage utilise la même configuration que la dernière fenêtre affichée dans la même session. Si vous exécutez plusieurs processus utilisateur, vous pouvez les tracer indépendamment et avoir une fenêtre de débogage ouverte pour chaque processus.

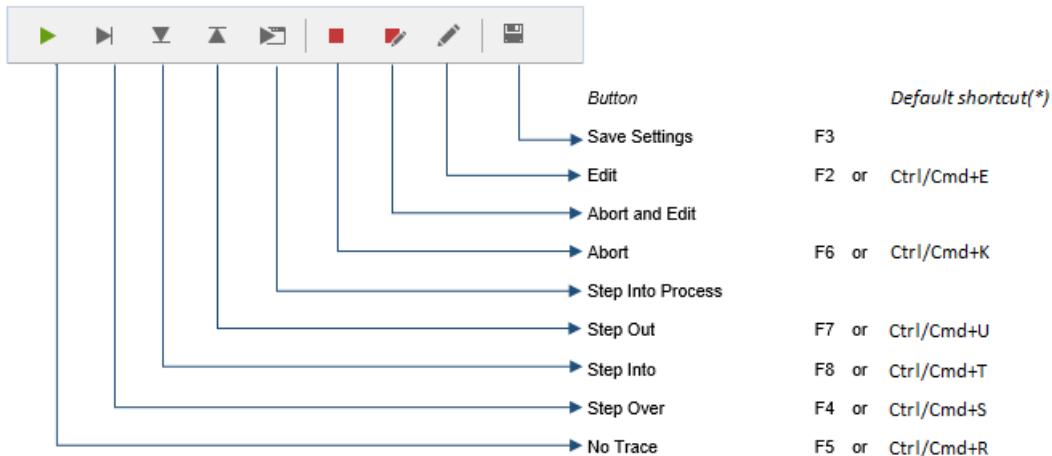
La fenêtre du débogueur est généralement affichée sur la machine où le code est exécuté. With a single-user application, it is always displayed on the machine running the application. With a client/server application, it is displayed:

- on the remote 4D for code running locally
- on the server machine for code running on the server (for example, a method with the execute on server option).

If the server is running headless, no debugger window can be displayed on the server, you need to use the remote debugger. See [Debugging from remote machines](#).

Tool bar Buttons

The debugger's tool bar includes several buttons, associated with default shortcuts:



Default shortcuts can be customized in the Shortcuts Page of the Preferences dialog box.

Bouton 'Reprendre exécution'

Arrêt du mode Trace et reprise du cours normal de l'exécution de la méthode.

La combinaison Maj+F5 ou Maj+clic sur le bouton Reprendre exécution provoque la reprise de l'exécution avec désactivation de tous les appels à TRACE suivants dans le process courant.

Bouton 'Exécuter pas à pas'

La ligne courante de la méthode (indiquée par la flèche jaune — cette flèche s'appelle le compteur de programme) est exécuté et le débogueur passe à la ligne suivante.

Le bouton Exécuter pas à pas ne passe pas dans les sous-routines et les fonctions. Il reste au niveau de la méthode que vous êtes en train de tracer. Si vous souhaitez également tracer les appels aux sous-routines et aux fonctions, utilisez le bouton Pas à pas détaillé.

Dans le débogage distant, lors de l'exécution de la méthode sur le serveur, la méthode parente est appelée après l'exécution de la dernière ligne de méthode enfant. Si la méthode parente est exécutée du côté distant, le bouton agit de la même manière que le bouton Reprendre exécution.

Bouton 'Exécuter pas à pas détaillé'

Lors de l'exécution d'une ligne qui appelle une autre méthode (sous-routine ou fonction), ce bouton provoque l'affichage de la méthode appelée dans la fenêtre du débogueur, et permet au développeur de passer pas à pas dans cette méthode.

La nouvelle méthode devient la méthode courante (en haut) dans la sous-fenêtre Fenêtre de chaîne d'appel de la fenêtre du débogueur.

Lors de l'exécution d'une ligne qui n'appelle pas une autre méthode, ce bouton se comporte comme le bouton Exécuter pas à pas.

Bouton 'Exécuter et sortir'

La méthode s'arrête et vous retournez là où vous étiez avant son exécution :

- Si vous tracez une méthode formulaire ou une méthode objet s'exécutant en réponse à un événement, elle s'arrête et vous retournez au formulaire.
- Si vous tracez une méthode s'exécutant à partir du mode Application, vous retournez à ce mode.

Bouton 'Stopper et éditer'

La méthode s'arrête comme lorsque vous cliquez sur Stopper exécution. The method that is executing when you click the Abort and Edit button opens in the Method Editor.

Conseil : Utilisez ce bouton lorsque vous connaissez les modifications à apporter à votre code, et le moment où elles doivent être effectuées pour pouvoir poursuivre le test de vos méthodes. Une fois vos modifications effectuées, ré-exécutez la méthode.

Modifier

La méthode s'arrête comme lorsque vous cliquez sur Stopper exécution. The method that is executing at the time you click the Edit button opens in the Method Editor.

If you use this button to modify a method, the modifications are only effective the next time it executes.

Tip: Use this button when you know which changes are required in your code and when they don't interfere with the rest of the code to be executed or traced.

Bouton 'Enregistrer paramètres'

Ce bouton permet de sauvegarder la configuration courante de la fenêtre du débogueur (taille et position de la fenêtre, emplacement des lignes de division et contenu de la zone d'évaluation des expressions). Elle sera alors utilisée par défaut à chaque ouverture de la base. This includes:

- the size and position of the window
- the position of the division lines and the contents of the area that evaluates the expressions

Ces paramétrages sont stockés dans le projet.

Cette action n'est pas disponible en mode débogueur distant (voir [Débogage depuis des machines distantes](#)).

Fenêtre d'expression

The Watch pane is displayed in the top left corner of the Debugger window, below the Execution Control Tool Bar. Voici un exemple :

Expression	Value
▶ 🔎 Line Objects	
◀ 📂 Variables	
▶ 📂 Interprocess	
◀ 📂 Process	
▶ 📂 Document	""
▶ 📂 Error	0
▶ 📂 FldDelimit	9
▶ 📂 OK	0
▶ 📂 RecDelimit	13
▶ 📂 Local	
▶ 📂 Parameters	
▶ 📂 Self	Nil
▶ 📂 Current Form Values	
▶ 🚗 Constants	
▶ 🚧 Semaphores	
▶ 🏢 Processes	
▶ 📈 Tables & Fields	
▶ 📁 Sets	
▶ 📄 Named Selections	
▶ 🌐 Information	
▶ 🌐 Web	

This pane is not available in remote debugging mode.

The Watch Pane displays useful general information about the system, the 4D environment, and the execution environment.

The Expression column displays the names of the objects and expressions. The Value column displays their current corresponding values. Clicking on any value on the right side of the pane allows you to modify the value of the object, if this is permitted for that object.

At any point, you can drag and drop themes, theme sublists (if any), and theme items to the [Custom Watch Pane](#).

Expression list

Line Objects

This theme lets you keep track of the values of the objects or expressions:

- used in the line of code to be executed (the one marked with the program counter—the yellow arrow in the [Source Code Pane](#)),
- used in the previous line of code

Since the previous line of code is the one that was just executed before, this theme therefore shows the objects or expressions of the current line before and after that the line was executed. Let's say you execute the following method:

```
TRACE
$a:=1
$b:=a+1
$c:=a+b
```

1. A Debugger window opens with the program counter set to the line with `a:=1`. At this point the Line Objects theme displays:

\$a	Indéfini

The `$a` variable is not yet initialized, but it is displayed because it is used in the line to be executed.

2. You click the Step Over button. The program counter is now set to the line `b:=a+1`. At this point, the theme displays:

\$a	1
\$b	Indéfini

The value of the `$a` variable is now 1. The `$b` variable is not yet initialized, but it is displayed because it is used in the line to be executed.

3. You click the Step Over button again. The program counter is now set on the line with `c:=a+b`. At this point the Line Objects theme displays:

\$c	Indéfini
\$a	1
\$b	2

The value of the `$b` variable is now 2. The `$c` variable is not yet initialized, but it is displayed because it is used in the line to be executed.

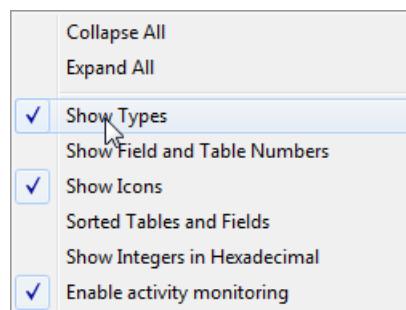
Variables

This theme is composed of the following subthemes:

Subtheme	Description	Can the values be modified?
Interprocess	List of interprocess variables being used at this point	Oui
Process	List of process variables used by the current process	Oui
Local	List of local variables used by the method being traced	Oui
Paramètres	List of parameters received by the method	Oui
Self	Pointer to the current object, when tracing an Object Method	Non

Arrays, like other variables, appear in the Interprocess, Process, and Local subthemes, depending on their scope. The debugger displays the first 100 elements. Inside the Value column, you can modify the values of array elements, but not the size of the arrays.

To display the variable types and their internal names, right click and check the Show Types option in the context menu:



Here's the result:

 Variable2 ->\$form.4B9.42 : Text  vRecNum ->vRecNum : Text	"" "2 of 2"
--	----------------

Current Form Values

This theme contains the name of each dynamic object included in the current form, as well as the value of its associated variable:

	Current Form Values	Form: debugger
	bCancel	0
	bDelete	0
	Button3	1
	bValidate	0
	FirstName	"Tony"
	ID	"2"
>	List Box1	0 elements
>	List Box1	Listbox sub objects

Some objects, such as list box arrays, can be presented as two distinct objects, the variable of the object itself and its data source.

Constantes

Like the Constants page of the Explorer window, this theme displays predefined constants provided by 4D. The expressions from this theme cannot be modified.

Semaphores

This theme lists the local semaphores currently being set. For each semaphore, the Value column provides the name of the process that sets the semaphore. The expressions from this theme cannot be modified. Global semaphores are not displayed.

Process

This theme lists the processes started since the beginning of the working session. The value column displays the time used and the current state for each process (i.e., Executing, Paused, and so on). The expressions from this theme cannot be modified.

Tables et champs

This theme lists the tables and fields in the 4D database. For each Table item, the Value column displays the size of the current selection for the current process as well as the number of locked records.

For each Field item, the Value column displays the value of the field for the current record (except picture and BLOB). You can modify the field values but not the tables' information.

Ensembles

This theme lists the sets defined in the current process (the one you're currently tracing) and the interprocess sets. For each set, the Value column displays the number of records and the table name. The expressions from this theme cannot be modified.

Sélections temporaires

This theme lists the named selections that are defined in the current process (the one you're currently tracing); it also lists the interprocess named selections. For each named selection, the Value column displays the number of records and the table name. The expressions from this theme cannot be modified.

Information

This theme contains general information regarding database operation, such as the current default table (if one exists), physical, virtual, free and used memory space, query destination, etc.

Web

This theme displays information regarding the main Web server of the application (only available if the Web server is active):

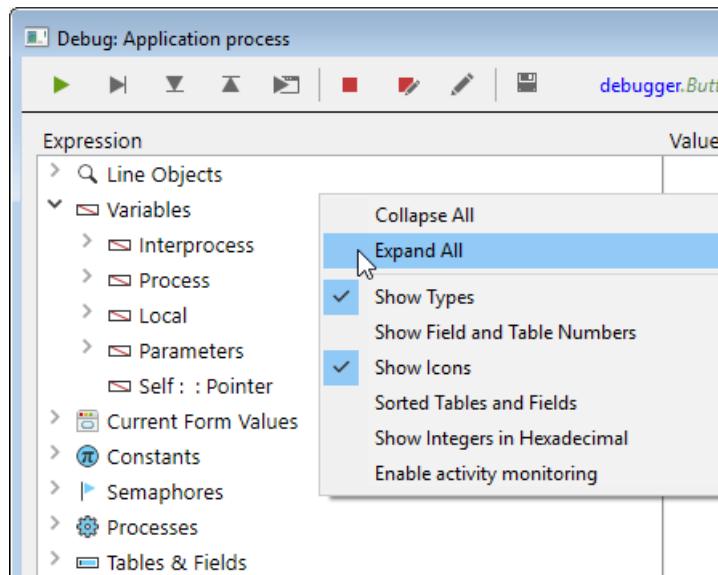
- Web File To Send: name of Web file waiting to be sent (if any)
- Web Cache Usage: number of pages present in Web cache as well as its use percentage

- Web Server Elapsed Time: duration of Web server use in hours:minutes:seconds format
- Web Hits Count: total number of HTTP requests received since Web server launch, as well as the instantaneous number of requests per second
- Number of active Web processes: number of active Web processes, all Web processes together

The expressions contained within this theme cannot be modified.

Menu contextuel de la fenêtre d'expression

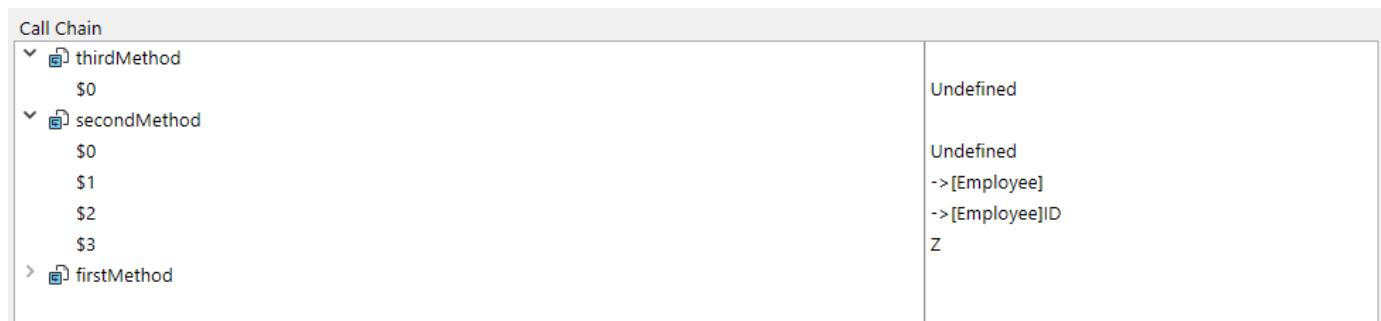
Le menu contextuel de la fenêtre d'expression vous propose des options supplémentaires.



- Contracter : Contracte tous les niveaux de la liste hiérarchique des expressions.
- Déployer : Déploie tous les niveaux de la liste hiérarchique des expressions.
- Montrer les types : Lorsque vous sélectionnez cette option, le type de l'objet s'affiche (lorsque cela est pertinent).
- Montrer le numéro de champ et de table : Affiche le numéro des tables ou de champs. Utile si vous travaillez avec des numéros de table ou de champs, ou avec des pointeurs utilisant les commandes Table ou Champ .
- Montrer les icônes : Chaque objet est précédé d'une icône qui indique son type. Vous pouvez désactiver cette option pour accélérer l'affichage, ou tout simplement parce que l'option Montrer les types vous convient.
- Tables et champs triés : Cette option force les tables et les champs à s'afficher par ordre alphabétique (dans leurs listes respectives).
- Afficher les entiers en hexadécimal : Les nombres s'affichent en notation décimale. Sélectionnez cette option pour les afficher en hexadécimal. Note : Pour exprimer une valeur numérique en hexadécimal, saisissez 0x (zéro + "x") puis les caractères hexadécimaux.
- Activer le suivi d'activité : Active le suivi d'activité (contrôle avancé de l'activité interne de l'application) et affiche les informations collectées dans des thèmes supplémentaires : Séquenceur, Web et Réseau.

Call Chain Pane

A method may call other methods or class functions, which may call other methods or functions. The Call Chain pane lets you keep track of that hierarchy.



Each main level item is the name of a method or class function. The top item is the one you are currently tracing, the

next main level item is the name of the caller (the method or function that called the one you are currently tracing), the next one is the caller's caller, and so on.

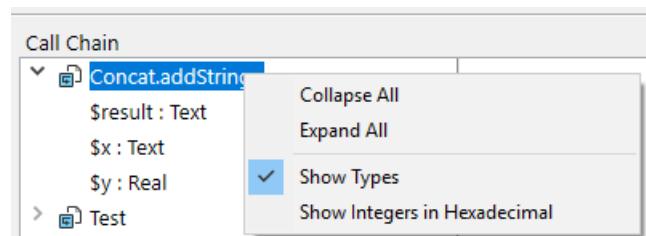
In the image above:

- `thirdMethod` has not received any parameter
- `$0` is currently undefined, as the method did not assign any value to `$0` (because it has not executed this assignment yet or because the method is a subroutine and not a function)
- `secondMethod` has received three parameters from `firstMethod` :
 - `$1` is a pointer to the `[Employee]` table
 - `$2` is a pointer to the `ID` field in the `[Employee]` table
 - `$3` is an alphanumeric parameter whose value is "Z"

You can double-click the name of any method to display its contents in the [Source Code Pane](#).

Clicking the icon next to a method or function name expands or collapses the parameters and the result (if any). Values appear on the right side of the pane. Clicking on any value on the right side allows you to change the value of any parameter or function result.

To display the parameter type, check the Show types option in the contextual menu:



After you deploy the list of parameters, you can drag and drop parameters and function results to the [Custom Watch Pane](#).

You can also use the [Get call chain](#) command to retrieve the call chain programmatically.

Custom Watch Pane

The Custom Watch Pane is useful for evaluating expressions. It is similar to the [Watch Pane](#), except here you decide which expressions are displayed. Any type of expression can be evaluated:

- field
- variable
- pointer
- calculation
- 4D command
- method
- and anything else that returns a value

Expression	Value
\$Text	"Hello, World!"
\$CalcResult	3
\$pField	->[Employee]ID
\$myBlob	10 Ko

You can evaluate any expression that can be shown in text form. This does not cover picture and BLOB fields or variables. To display BLOB contents, you can use BLOB commands, such as [BLOB to text](#).

Handling expressions

There are several ways to add expressions to the list:

- Drag and drop an object or expression from the Watch Pane or the Call Chain Pane

- Select an expression in the [Source Code pane](#) and press **ctrl+D** (Windows) or **cmd+D** (macOS)
- Double-click somewhere in the empty space of the Custom Watch Pane (adds an expression with a placeholder name that you can edit)

You can enter any formula that returns a result.

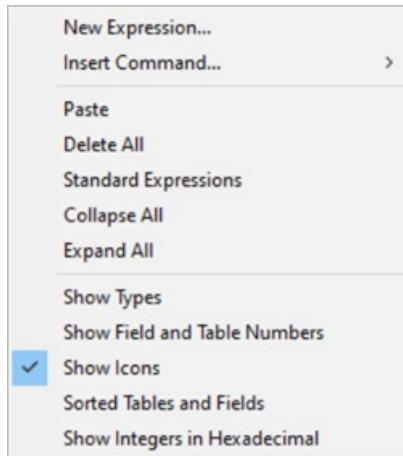
To edit an expression, click on it to select it, then click again or press **Enter** on your keyboard.

To delete an expression, click on it to select it, then press **Backspace** or **Delete** on your keyboard.

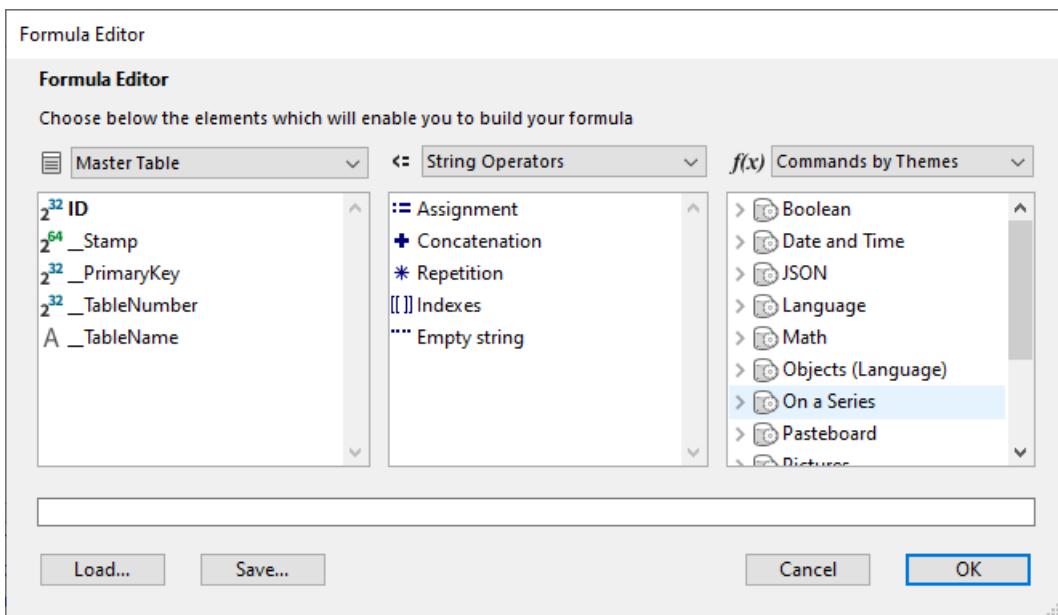
Warning: Be careful when you evaluate a 4D expression modifying the value of one of the System Variables (for instance, the **OK** variable) because the execution of the rest of the method may be altered.

Menu contextuel de la fenêtre d'expression

The Custom Watch Pane's context menu gives you access the 4D formula editor and other options:



New Expression : This inserts a new expression and displays the 4D Formula Editor.



For more information on the Formula Editor, see the [4D Design Reference manual](#).

- Insert Command: Shortcut for inserting a 4D command as a new expression.
- Delete All: Removes all expressions from the Custom Watch Pane.
- Standard Expressions: Copies the Watch Pane's list of expressions.

This option is not available in remote debugging mode (see [Debugging from Remote Machines](#)).

- Collapse All/Expand All: Collapses or Expands all the hierarchical lists.

- Show Types: Displays the type of each item in the list (when appropriate).
- Show Field and Table Numbers: Displays the number of each table or field of the Fields. Useful if you work with tables, field numbers or pointers using the commands such as `Table` or `Field`.
- Show Icons: Displays an icon denoting the type of each item.
- Sorted Tables and Fields: Displays the table and fields in alphabetical order.
- Show Integers in Hexadecimal: Displays numbers using hexadecimal notation. To enter a numeric value in hexadecimal, type `0x` (zero + "x"), followed by the hexadecimal digits.

Source Code Pane

The Source Code Pane shows the source code of the method or function currently being traced.

This area also allows you to add or remove [break points](#).

Tool tip

Hover your pointer over any expression to display a tool tip that indicates:

- the declared type of the expression
- the current value of the expression

```

1 // $1 contains the primary key of the manager
2 // $2 is a pointer to the resulting array
3
4 (●) QUERY([Employee];[Employee]ID;=$1) // finds the employee whose primary key w
5 APPEND TO ARRAY($2->,[Employee]firstname+" "+[Employee]lastname) // and adds
6
7 (●) QUERY([Employee];[Employee]managerID;=;[Employee]ID) // now finds all direct
8
9   v If (Records in selection([Employee])>0) // if there are some
10
11     C_LONGINT($i)
12     ARRAY LONGINT($IDS;0)
13     SELECTION TO ARRAY([Employee]ID;$IDS)

```

A tooltip is displayed over the expression `Records in selection([Employee])`, showing the value `Long Integer = 1`.

This also works with selections:

```

1 // $1 contains the primary key of the manager
2 // $2 is a pointer to the resulting array
3
4 QUERY([Employee];[Employee]ID;=$1) // finds the employee whose primary key w
5 APPEND TO ARRAY($2->,[Employee]firstname+" "+[Employee]lastname) // and adds
6
7 (●) QUERY([Employee];[Employee]managerID;=;[Employee]ID) // now finds all direct
8
9   v If (Records in selection([Employee])>0) // if there are some
10
11     C_LONGINT($i)
12     ARRAY LONGINT($IDS;0)
13     SELECTION TO ARRAY([Employee]ID;$IDS)

```

A tooltip is displayed over the expression `Records in selection([Employee])`, showing the value `Long Integer = 1`.

Adding expressions to the Custom Watch Pane

You can copy any selected expression from the Source Code Pane to the [Custom Watch Pane](#).

1. In the Source code pane, select the expression to evaluate
2. Do one of the following:
 - Drag and drop the selected text to the Expression area of the Custom Watch Pane
 - Press **Ctrl+D** (Windows) or **Cmd+D** (macOS)
 - Right-click the selected text > Copy to Expression Pane

Program Counter

The yellow arrow in the left margin of the Source Code pane is called the program counter. It marks the next line to be executed.

By default, the program counter line (also called the running line) is highlighted in the debugger. You can customize the highlight color in the [Methods page of the Preferences](#).

Moving the program counter

For debugging purposes, you can move the program counter for the method at the top of the call chain (the method currently executing). To do so, click and drag the yellow arrow to another line.

This only tells the debugger to pursue tracing or executing from a different point. It does not execute lines or cancel their execution. All current settings, fields, variables, etc. are not impacted.

Par exemple :

```
// ...
If(This condition)
    DO_SOMETHING
Else
    DO_SOMETHING_ELSE
End if
// ...
```

Say the program counter is set to the line `If (This condition)`. When you click the Step over button, the program counter moves directly to the `DO_SOMETHING_ELSE` line. To examine the results of the `DO_SOMETHING` line, you can move the program counter to that line and execute it.

Menu contextuel de la fenêtre d'évaluation des méthodes

Le menu contextuel de la Fenêtre d'évaluation des méthodes donne accès à plusieurs fonctions utiles en phase d'exécution des méthodes en mode Trace :

Debug: P_5

Get Reporting Emps

Expression	Value
> Line Objects	
> Variables	
> Current Form Values	
> Constants	
> Semaphores	
> Processes	
> Tables & Fields	
> Sets	
> Named Selections	
> Information	
> Web	

Call Chain
> Get Reporting Emps

Expression	Value
\$result	Undefined
Form	null
\$2	->\$reportingEmps (Get All Reporting Emps)
\$1	1
Local	

```

1 // $1 contains the primary key of the manager
2 // $2 is a pointer to the resulting array
3
4 QUERY([Employee]ID;=[Employee]primarykey) // finds the employee whose primary key is $1
5 APPEND TO A
6
7 QUERY([Employee]ID;=[Employee]primarykey) // finds all direct reports of $1
8
9 If (Records>0) // if there are some
10
11 C_LONGINT()
12 ARRAY LONG
13 SELECTION
14
15 For ($i;1;Size of array($IDS))
16   Get Reporting Emps ($IDS{$i};$2) // now get all reporting employees of $1
17 End for
18

```

Breakpoint Reached

Context menu for line 7 (QUERY([Employee]ID;=[Employee]primarykey)):

- Goto Definition...
- Search References...
- Copy
- Copy to Expression Pane
- Run to Cursor
- Set Next Statement
- Toggle Breakpoint
- Edit Breakpoint...

- Aller à définition : permet d'accéder à la définition de l'objet sélectionné. Cette commande est disponible avec les objets suivants :
 - méthode projet : affiche le contenu de la méthode dans une nouvelle fenêtre de l'éditeur de méthodes
 - Champ : affiche les propriétés du champ dans l'inspecteur de la fenêtre de structure
 - table : affiche les propriétés de la table dans l'inspecteur de la fenêtre de structure
 - formulaire : affiche le formulaire dans l'éditeur de formulaires
 - variable (locale, process, interprocess ou paramètre \$n) : affiche la ligne de déclaration de la variable dans la méthode courante ou parmi les méthodes compilateur
- Chercher les références (cette fonction est également accessible depuis l'éditeur de méthodes) : rechercher tous les objets du projet (méthodes et formulaires) dans lesquels l'élément courant de la méthode est référencé. L'élément courant est l'élément sélectionné ou l'élément dans lequel se trouve le curseur. Il peut s'agir d'un nom de champ, de variable, de commande, d'une chaîne, etc. Le résultat de la recherche est affiché dans une nouvelle fenêtre de résultat standard.
- Copier : copie standard de l'expression sélectionnée dans le conteneur de données.
- Copier dans la fenêtre d'expression : copie l'expression sélectionnée dans la Fenêtre d'évaluation.
- Exécuter jusqu'au curseur : provoque l'exécution des instructions situées entre le compteur de programme (flèche jaune) et la ligne sélectionnée de la méthode (dans laquelle se trouve le curseur).
- Fixer prochaine instruction : déplace le compteur de programme jusqu'à la ligne sélectionnée sans l'exécuter et sans exécuter les lignes intermédiaires. La ligne désignée ne sera exécutée que si l'utilisateur clique sur l'un des boutons d'exécution.
- Permuter point d'arrêt (également accessible depuis l'éditeur de méthodes) : Elle permet alternativement d'insérer ou de supprimer le point d'arrêt correspondant la ligne sélectionnée. Cette fonction modifie le point d'arrêt de façon permanente : par exemple, un point d'arrêt supprimé dans le débogueur n'apparaît plus dans la méthode d'origine.
- **Modifier point d'arrêt...** (également accessible depuis l'éditeur de méthodes) : Elle permet d'afficher la boîte de

dialogue de définition des Propriétés du point d'arrêt. Cette fonction modifie le point d'arrêt de façon permanente.

Find Next/Previous

Specific shortcuts allow you to find strings identical to the one selected:

- To search for the next identical strings, press Ctrl+E (Windows) or Cmd+E (macOS)
- To search for the previous identical strings, press Ctrl+Shift+E (Windows) or Cmd+Shift+E (macOS)

The search is carried out only if you select at least one character in the Source code pane.

Raccourcis

Cette section répertorie tous les raccourcis disponibles dans la fenêtre du débogueur.

La barre d'outils comporte également des [raccourcis](#).

Fenêtre d'évaluation & Sous-fenêtre d'évaluation

- Un double-clic sur un article de la fenêtre d'expression copie cet article dans la fenêtre d'évaluation
- Un double-clic dans la sous-fenêtre d'évaluation crée une nouvelle expression

Source Code Pane

- Un clic dans la marge gauche place ou supprime un point d'arrêt.
- Alt+Majuscule+clic (Windows) ou Option+Majuscule+clic (macOS) pose un point d'arrêt provisoire.
- Alt+clic (Windows) ou Option+clic (macOS) affiche la fenêtre des propriétés du point d'arrêt pour un point d'arrêt nouveau ou existant.
- Une expression ou un objet sélectionné(e) peut être copié(e) dans la Fenêtre d'évaluation par glisser-déposer.
- Ctrl+D (Windows) ou Commande+D (macOS) sur un texte sélectionné le copie dans la Fenêtre d'évaluation.
- Ctrl+E (Windows) ou Commande+E (macOS) identifie les chaînes suivantes qui sont identiques à la chaîne sélectionnée.
- Ctrl+Majuscule+E (Windows) ou Commande+Majuscule+E (macOS) identifie les chaînes précédentes qui sont identiques à la chaîne sélectionnée.

Toutes les fenêtres

- Ctrl + +/- (Windows) ou Commande + +/- (macOS) augmente ou réduit la taille de la police pour une meilleure lisibilité. La taille de la police modifiée s'applique également à l'Editeur de méthodes et est stockée dans les Préférences.
- Ctrl+(Windows) ou Commande+ (macOS) force la réactualisation de la Fenêtre d'expression.
- Lorsqu'aucun objet n'est sélectionné dans les fenêtres, en appuyant sur Entrée, vous avancez d'une ligne.
- Lorsque la valeur d'un élément est sélectionnée, utilisez les touches directionnelles pour naviguer dans la liste.
- Lorsque vous êtes en train d'éditer un élément, utilisez les touches directionnelles pour déplacer le curseur. Utilisez Ctrl+A/X/C/V (Windows) ou Commande+A/X/C/V (macOS) en raccourci des commandes du menu Edition : Tout Sélectionner/Couper/Copier/Coller.

Points d'arrêt et points d'arrêt sur commande

Aperçu

Les points d'arrêt et les points d'arrêts sur commande sont des techniques de débogage très efficaces. Elles ont toutes deux le même effet : elles interrompent l'exécution du code (et affichent la fenêtre du débogueur si elle n'est pas déjà affichée) à une étape souhaitée.

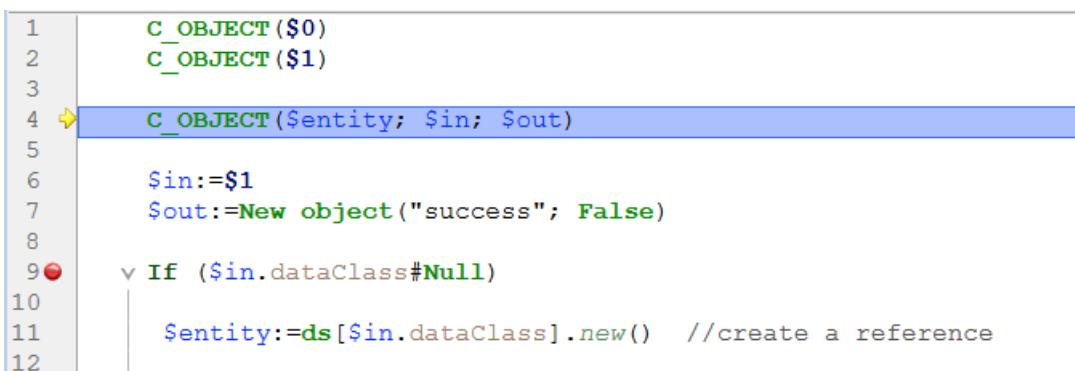
Définissez des points d'arrêt sur n'importe quelle ligne de code où vous souhaitez interrompre l'exécution. Vous pouvez associer une condition au point d'arrêt.

Les points d'arrêt sur commande vous permettent de commencer à tracer l'exécution d'un process dès qu'une commande est appelée par ce process.

Points d'arrêt

Pour créer un point d'arrêt, cliquez dans la marge gauche du volet Code source dans le débogueur ou dans l'éditeur de méthode.

Dans l'exemple suivant, un point d'arrêt (la puce rouge) a été défini dans le débogueur, sur la ligne `If ($in.dataClass#Null)` :



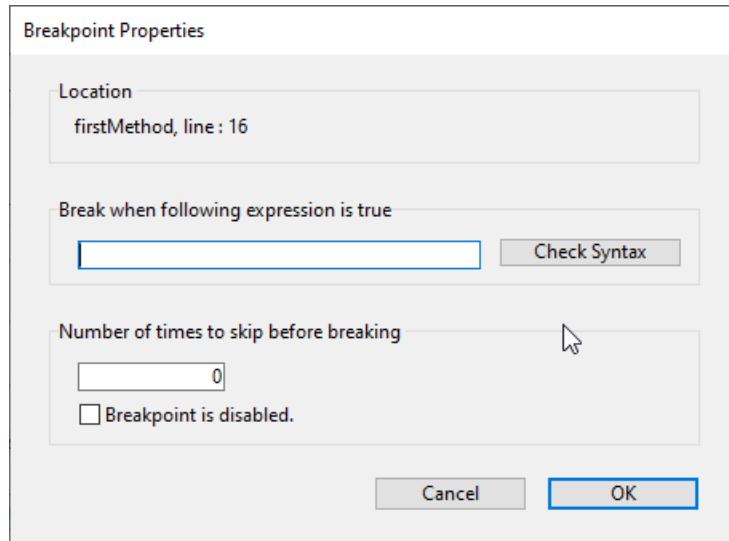
```
1 C_OBJECT($0)
2 C_OBJECT($1)
3
4 C_OBJECT($entity; $in; $out) ⏵
5
6     $in:=$1
7     $out:=New object("success"; False)
8
9 If ($in.dataClass#Null) ⏷
10
11     $entity:=ds[$in.dataClass].new() //create a reference
12
```

Dans l'exemple ci-dessus, le fait de cliquer sur le bouton `No Trace` reprend l'exécution normale jusqu'à la ligne marquée par le point d'arrêt. Cette ligne n'est pas exécutée - vous retournez en mode trace. Le fait de définir un point d'arrêt au-delà du compteur du programme et de cliquer sur le bouton `No Trace` vous permet de sauter des parties de la méthode tracée.

Pour supprimer un point d'arrêt, cliquez sur la puce correspondante.

Propriétés des points d'arrêt

Vous pouvez modifier le comportement d'un point d'arrêt à l'aide de la fenêtre Propriétés du point d'arrêt :



Cette fenêtre est disponible à partir de l'Editeur de méthode ou du [volet de code source](#). Vous pouvez :

- faites un clic droit sur une ligne et sélectionnez `Modifier le point d'arrêt` dans le menu contextuel, ou
- `Alt+clic` (Windows) ou `Option+clic` (macOS) dans la marge de gauche.

Si un point d'arrêt existe déjà, la fenêtre s'affiche pour ce point d'arrêt. Sinon, un point d'arrêt est créé et la fenêtre s'affiche pour le point d'arrêt nouvellement créé.

Voici une description des propriétés :

- Emplacement : indique le nom de la méthode et le numéro de ligne associés au point d'arrêt.
- Break when following expression is true : Vous pouvez créer des points d'arrêt conditionnels en saisissant une formule 4D qui retourne `True` ou `False`. Par exemple, insérez `Records in selection(\[aTable]\)=0` pour vous assurer que l'arrêt se produit uniquement si aucun enregistrement n'est sélectionné pour la table `[aTable]`. Les conditions d'arrêt sont disponibles dans la colonne Condition de la [liste des arrêts](#).
- Number of times to skip before breaking : Vous pouvez associer un point d'arrêt à une ligne située dans une boucle (`While`, `Repeat` ou `For`) ou située dans une sous-routine ou une fonction appelée depuis une boucle.
- Breakpoint is disabled : Si vous n'avez actuellement pas besoin d'un point d'arrêt, mais que vous pourriez en avoir besoin plus tard, vous pouvez le désactiver temporairement. Un point d'arrêt désactivé apparaît sous la forme d'un tiret (-) au lieu d'une puce (•)

Points d'arrêt dans le débogage à distance

La liste des points d'arrêt est stockée localement. En mode de débogage à distance, si le débogueur connecté est un 4D distant, la liste des points d'arrêt distants remplace temporairement la liste des points d'arrêt du serveur pendant la session de débogage.

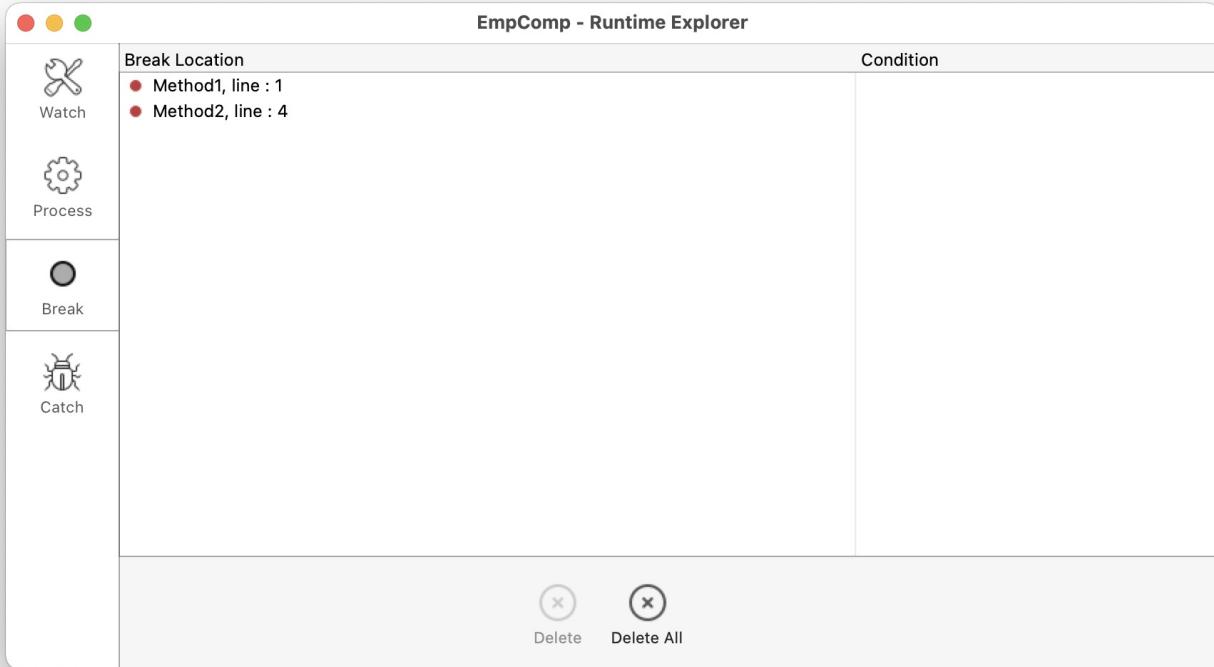
La liste de points d'arrêt du serveur est automatiquement restaurée s'il redevient le débogueur associé.

Liste des points d'arrêt

La liste des points d'arrêt est une page de l'Explorateur d'exécution qui vous permet de gérer les points d'arrêt créés dans la fenêtre du débogueur ou dans l'éditeur de méthode. Pour plus d'informations sur l'Explorateur d'exécution, voir sa page dédiée dans [le manuel de Développement](#).

Pour ouvrir la page de la liste des points d'arrêt :

1. Dans le menu Exécuter, cliquez sur `Explorateur d'exécution...`
2. Cliquez sur l'onglet `Break` pour afficher la liste des points d'arrêts :



À l'aide de cette fenêtre, vous pouvez :

- Définir des conditions pour les points d'arrêt dans la colonne Conditions
- Activer ou désactiver les points d'arrêt en cliquant sur les puces dans la marge. Les points d'arrêt désactivés affichent des puces transparentes
- Supprimer des points d'arrêt en appuyant sur la touche `Delete` ou `Backspace`, ou en cliquant sur le bouton `Delete` sous la liste.
- Ouvrez les méthodes où se trouvent les points d'arrêt en double-cliquant sur n'importe quelle ligne de la liste

Vous ne pouvez pas ajouter de nouveaux points d'arrêt à partir de cette fenêtre. Les points d'arrêt ne peuvent être créés qu'à partir de la fenêtre du débogueur ou de l'éditeur de méthodes.

Points d'arrêts sur les commandes

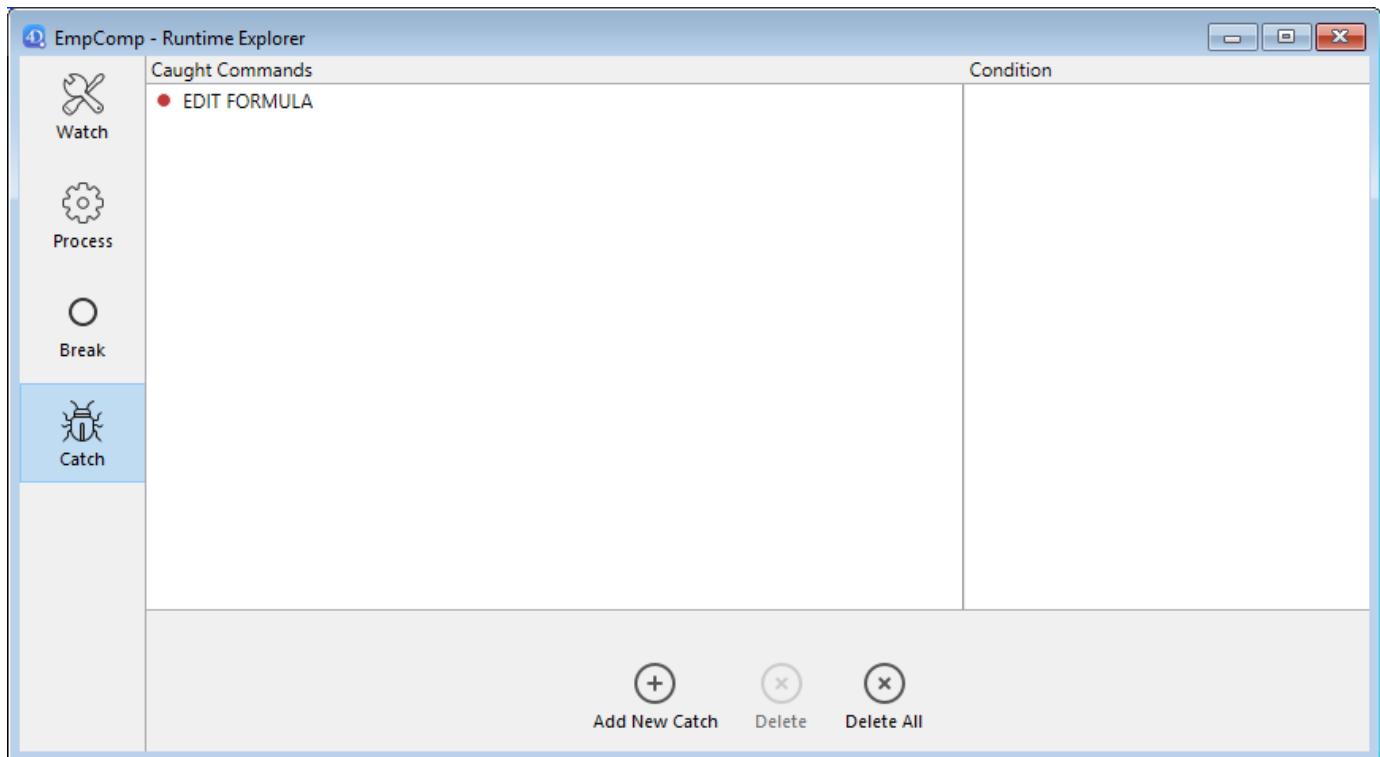
L'onglet **Catch** de l'Explorateur d'exécution vous permet d'ajouter des points d'arrêt supplémentaires à votre code via des appels aux commandes 4D. Contrairement à un point d'arrêt, qui est situé dans une méthode de projet particulière (et qui déclenche donc une exception de traçage uniquement lorsqu'il est atteint), la portée de la capture d'une commande inclut tous les process qui exécutent le code 4D et appellent cette commande.

Les points d'arrêt sur une commande sont un moyen pratique de tracer de grandes portions de code sans avoir à définir des points d'arrêt à des endroits arbitraires. Par exemple, si un enregistrement qui ne devrait pas être supprimé l'est malgré tout après avoir exécuté un ou plusieurs process, vous pouvez essayer de réduire le champ de votre investigation via des points d'arrêt sur des commandes telles que `DELETE RECORD` et `DELETE SELECTION`. Chaque fois que ces commandes sont appelées, vous pouvez vérifier si l'enregistrement en question a été supprimé, et ainsi isoler la partie défective du code.

N'hésitez pas à combiner les points d'arrêt et les points d'arrêt sur commandes.

Pour ouvrir la page des points d'arrêt sur commandes :

1. Cliquez sur Exécuter > Explorateur d'exécution... pour ouvrir l'explorateur d'exécution.
2. Cliquez sur **Catch** pour afficher la liste des points d'arrêt sur commandes :



Cette page répertorie les points d'arrêt sur commande pendant l'exécution. Elle est composée de deux colonnes :

- La colonne de gauche affiche l'état d'activation/désactivation du point d'arrêt sur commande, suivi du nom de la commande
- La colonne de droite affiche la condition associée au point d'arrêt sur commande, le cas échéant

Pour ajouter un point d'arrêt sur commande :

1. Cliquez sur le bouton Add New Catch (en forme de +) situé en dessous de la liste. Une nouvelle entrée est ajoutée à la liste avec la commande ALERT par défaut
2. Cliquez sur l'étiquette ALERT, tapez le nom de la commande que sur laquelle vous souhaitez mettre un point d'arrêt, puis appuyez sur Entrée.

Pour activer ou désactiver un point d'arrêt sur commande, cliquez sur la puce (•) devant l'étiquette de la commande. La puce est transparente lorsque la commande est désactivée.

La désactivation d'un point d'arrêt sur commande a pratiquement le même effet que sa suppression. Pendant l'exécution, le débogueur ne passe presque pas de temps sur l'entrée. L'avantage de désactiver une entrée est de ne pas avoir à la recréer lorsque vous en avez à nouveau besoin.

Pour supprimer un point d'arrêt sur commande :

1. Sélectionnez une commande dans la liste.
2. Appuyez sur la touche Backspace ou Delete sur votre clavier ou cliquez sur le bouton Delete sous la liste (Delete All supprime toutes les commandes de la liste).

Définir une condition pour un point d'arrêt sur commande

1. Cliquez sur l'entrée dans la colonne de droite
2. Saisissez une formule 4D (expression, appel de commande ou méthode de projet) qui retourne une valeur booléenne.

Pour supprimer une condition, supprimez sa formule.

L'ajout de conditions vous permet d'arrêter l'exécution lorsque la commande est invoquée uniquement si la condition est remplie. Par exemple, si vous associez la condition `Records in selection(\[Emp]>10)` au point d'arrêt de la commande `DELETE SELECTION`, le code ne sera pas arrêté pendant l'exécution de la commande `DELETE SELECTION` si la sélection courante de la table [Emp] ne contient que 9 enregistrements (ou moins).

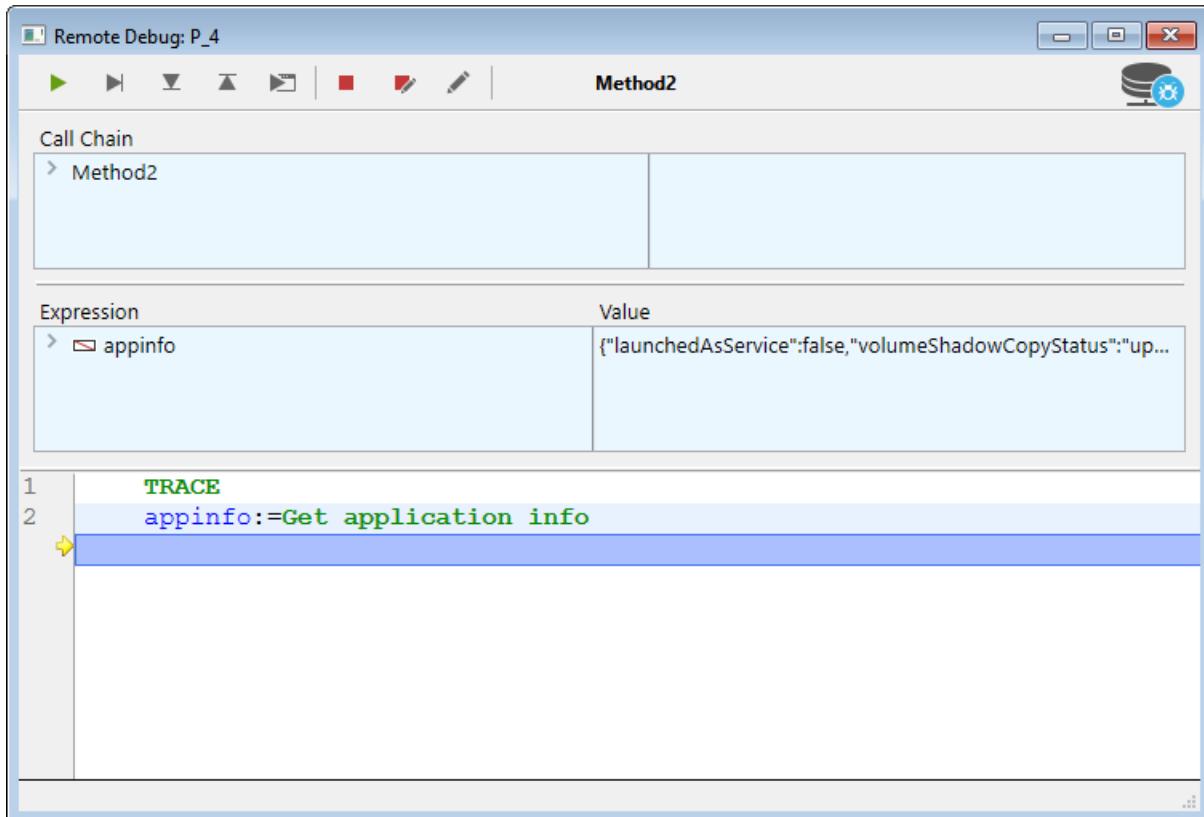
L'ajout de conditions aux points d'arrêt sur commandes ralentit l'exécution, car la condition doit être évaluée chaque fois qu'une exception est rencontrée. En revanche, l'ajout de conditions accélère le processus de débogage, car 4D ignore automatiquement les occurrences qui ne correspondent pas aux conditions.

Débogage depuis des machines distantes

Aperçu

Lorsqu'une base de données 4D est exécutée sur 4D Server en mode interprété, vous pouvez déboguer le code 4D exécuté sur le serveur à partir d'un client 4D distant connecté au projet. Il suffit d'associer le débogueur à une machine distante spécifique, et l'exécution du code peut être surveillée dans le débogueur directement sur la machine distante.

Sur une machine distante, la [fenêtre du débogueur](#) affiche une icône de serveur spécifique et une couleur de fond bleue pour indiquer que vous déboguez du code serveur :



Cette fonctionnalité est particulièrement utile lorsque 4D Server fonctionne en mode headless (voir [Interface en ligne de commande](#)), ou lorsque l'accès à la machine serveur n'est pas facile.

Débogueur associé

Un seul débogueur peut déboguer une application 4D Server à un moment donné. Il s'agit du débogueur associé. Le débogueur associé peut être :

- le débogueur local de 4D Server (par défaut) - si le serveur ne tourne pas en mode headless.
- le débogueur d'un client 4D distant - si la session distante a accès au mode Développement.

Le débogueur associé est appelé chaque fois que 4D Server rencontre :

- un point d'arrêt
- une commande `TRACE`
- un point d'arrêt sur commande
- une erreur

A noter que les messages d'erreur sont envoyés à la machine du débogueur associé. Cela signifie que dans le cas d'un débogueur distant, les messages d'erreur du serveur sont affichés sur le client 4D distant.

A noter que :

- Le code exécuté dans la méthode base `On Server Startup Database` ne peut pas être débogué à distance. Il ne peut être débogué que du côté serveur
- Si aucun débogueur n'est associé, le code en cours d'exécution n'est pas arrêté par les commandes de débogage

Associer le débogueur

By default when you start an interpreted application:

- si 4D Server ne fonctionne pas en mode "headless", le débogueur est associé au serveur,
- si 4D Server fonctionne en mode "headless", aucun débogueur n'est associé.

Vous pouvez associer le débogueur à tout client 4D distant autorisé à se connecter à l'application 4D Server.

La session utilisateur du client 4D distant doit avoir accès à l'environnement de développement de la base de données.

Pour associer le débogueur à un client 4D distant :

1. In the 4D Server menu bar, select Edit > Detach Debugger so that the debugger becomes available to remote machines (this step is useless if the 4D Server is running headless).
2. Dans un client 4D distant connecté au serveur, sélectionnez Exécuter > Attacher le débogueur distatin

Si le rattachement est accepté (voir [Rejected attachment requests](#)), la commande de menu devient Detach Remote Debugger.

Le débogueur est alors attaché au client 4D distant :

- jusqu'à la fin de la session utilisateur
- jusqu'à ce que vous sélectionniez `Detach Remote Debugger`

To attach the debugger back to the server:

1. On the remote 4D client that has the debugger attached, select Run > Detach Remote Debugger.
2. In the 4D Server menu bar, select Edit > Attach debugger.

When the debugger is attached to the server (default), all server processes are automatically executed in cooperative mode to enable debugging. This can have a significant impact on performance. When you don't need to debug on the server machine, it is recommended to detach the debugger and attach it to a remote machine if necessary.

Attaching debugger at startup

4D vous permet d'associer automatiquement le débogueur à un client 4D distant ou au serveur au démarrage :

- Sur le serveur (s'il n'est pas headless), cette option s'appelle `Attach Debugger At Startup`. Lorsque le serveur est démarré, il associe automatiquement le débogueur (par défaut).

Avertissement : Si cette option est sélectionnée pour un serveur qui est ensuite lancé en mode headless, le débogueur ne sera pas disponible pour ce serveur.

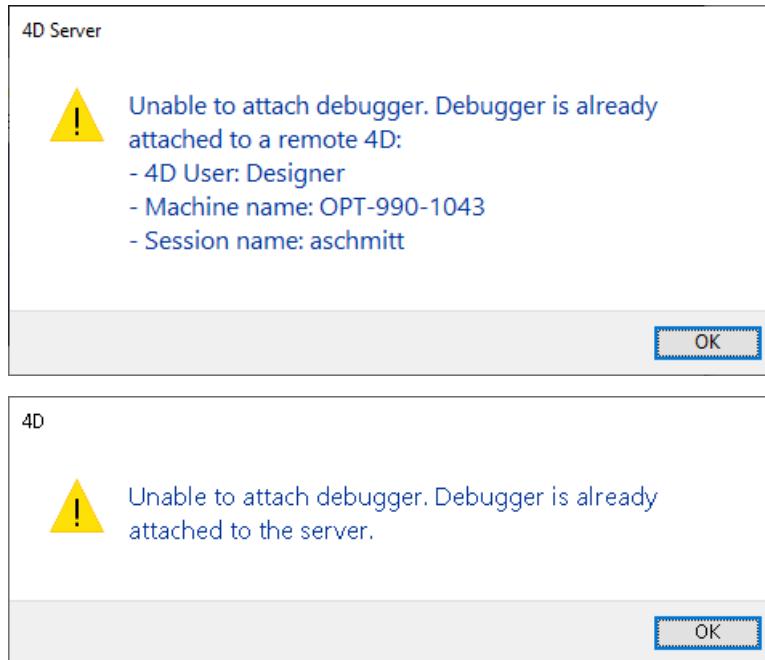
- Sur un client 4D distant, cette option s'appelle `Attach Remote Debugger At Startup`. Lorsqu'elle est sélectionnée, le client 4D distant tente automatiquement d'attacher le débogueur distant à chaque connexion ultérieure à la même base de données 4D Server. Si l'association est acceptée (voir [Rejected attachment requests](#)), le débogueur distant est automatiquement attaché au client 4D distant et l'option `Detach Remote Debugger` s'affiche.

Ce paramètre s'applique par projet et est stocké localement dans le fichier `.4DPreferences`.

Demandes d'association rejetées

While the debugger is already attached to a remote 4D client or to 4D Server, no other machine can attach the debugger.

Si une machine tente d'attacher le débogueur alors qu'il est déjà attaché, l'attachement est rejeté et une boîte de dialogue apparaît :



L'attachement du débogueur dans ce cas nécessite que :

- the attached debugger is detached from the server or from the remote 4D client using respectively the Detach debugger or Detach remote debugger menu command,
- the attached remote 4D client session is closed.

Description des fichiers d'historique

Les applications 4D peuvent générer divers fichiers d'historique (ou "logs") qui sont utiles pour le débogage ou l'optimisation de leur exécution. Les fichiers d'historique sont généralement démarrés ou arrêtés via des sélecteurs des commandes [SET DATABASE PARAMETER](#) ou [WEB SET OPTION](#) et sont stockés dans le dossier [Logs](#) du projet.

Les informations de l'historique doivent être analysées pour détecter et corriger les problèmes. Cette section fournit une description complète des fichiers journaux suivants :

- [4DRequestsLog.txt](#)
- [4DRequestsLog_ProcessInfo.txt](#)
- [HTTPDebugLog.txt](#)
- [4DDebugLog.txt \(standard & tabular\)](#)
- [4DDiagnosticLog.txt](#)
- [4DIMAPLog.txt](#)
- [4DPOP3Log.txt](#)
- [4DSMTPLog.txt](#)
- [Fichier d'historique des requêtes ORDA clientes](#)

Lorsqu'un fichier d'historique peut être généré soit sur 4D Server, soit sur le client distant, le mot " Server " est ajouté au nom du fichier d'historique côté serveur, par exemple " 4DRequestsLogServer.txt"

Les fichiers d'historique partagent certains champs, ce qui vous permet d'établir une chronologie et d'établir des connexions entre les entrées lors du débogage :

- `sequence_number` : ce numéro est unique parmi tous les fichiers d'historique de débogage et est incrémenté à chaque nouvelle entrée, quel que soit le fichier d'historique, de manière à ce que vous puissiez connaître la séquence exacte des opérations.
- `connection_uuid` : pour chaque process 4D créé sur un client 4D qui se connecte au serveur, cet UUID de connexion est stocké à la fois côté serveur et client. Il vous permet d'identifier facilement le client distant qui a lancé le process.

4DRequestsLog.txt

Ce fichier d'historique enregistre les requêtes standard envoyées par la machine de 4D Server ou par la machine 4D distante qui a exécuté la commande (hors requêtes web).

Pour démarrer ce fichier d'historique :

- sur le serveur :

```
SET DATABASE PARAMETER(4D Server log recording;1)
//côté serveur
```

- sur le client :

```
SET DATABASE PARAMETER(Client Log Recording;1)
//côté distant
```

Cette instruction démarre également le fichier d'historique [4DRequestsLog_ProcessInfo.txt](#).

Ce fichier commence avec les en-têtes suivants :

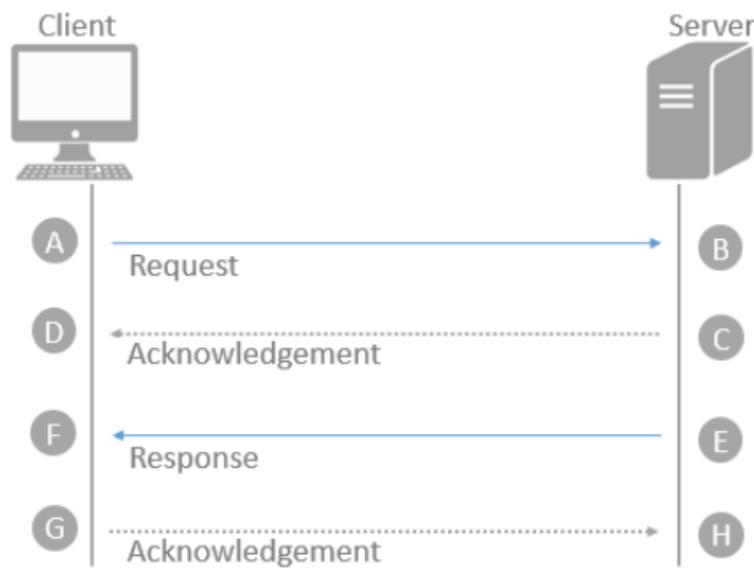
- Log Session Identifier (Identifiant de session d'historique)
- Nom du serveur qui héberge l'application
- User Login Name : Nom de l'utilisateur (défini dans l'OS) qui a exécuté l'application 4D sur le serveur.

Contenu

Pour chaque requête, les champs suivants sont enregistrés :

Nom des champs	Description
sequence_number	Numéro d'opération séquentiel et unique dans la session d'historique
time	Date et heure au format ISO 8601 : 'YYYY-MM-DDTHH:MM:SS.mmm'
systemid	ID système
component	Signature du composant (par exemple '4SQLS' ou 'dbmg')
process_info_	correspond au champ "index" dans le fichier d'historique 4DRequestsLog_ProcessInfo.txt, permettant de relier une requête à un process.
request	ID de requête en mode distant chaîne de message pour les requêtes SQL ou messages LOG EVENT
bytes_in	Nombre d'octets reçus
bytes_out	Nombre d'octets envoyés
server_duration exec_duration	Dépend de l'endroit où l'historique est généré : <ul style="list-style-type: none">• <i>server_duration</i> lorsqu'il est généré sur le client --Temps en microsecondes pris par le serveur pour traiter la requête et retourner une réponse. Correspond au chemin B vers F dans l'image ci-dessous, OU• <i>exec_duration</i> lorsqu'il est généré sur le serveur --Temps en microsecondes pris par le serveur pour traiter la requête. Correspond au chemin B vers F dans l'image ci-dessous.
write_duration	Temps en microsecondes pour envoyer : <ul style="list-style-type: none">• La requête (lorsqu'elle est exécutée sur le client). Correspond au chemin A vers B dans l'image ci-dessous.• La réponse (lorsqu'elle est exécutée sur le serveur). Correspond au chemin E vers F dans l'image ci-dessous.
task_kind	Préemptif ou coopératif (respectivement 'p' ou 'c')
rtt	Temps en microsecondes pris par le client pour envoyer la requête et pour qu'elle soit reçue par le serveur. Correspond respectivement aux chemins A vers D et E vers H dans l'image ci-dessous. <ul style="list-style-type: none">• Mesuré uniquement lorsque la couche réseau ServerNet est utilisée, retourne 0 lorsque l'ancienne couche réseau est utilisée.• Dans les versions antérieures à Windows 10 ou à Windows Server 2016, l'appel retournera la valeur 0.

Acheminement de la requête :



4DRequestsLog_ProcessInfo.txt

Ce fichier d'historique enregistre des informations sur chaque process créé sur la machine de 4D Server, ou sur la machine 4D distante qui a exécuté la commande (hors requêtes web).

Pour démarrer ce fichier d'historique :

- sur le serveur :

```
SET DATABASE PARAMETER(4D Server log recording;1) //côté serveur
```

- sur le client :

```
SET DATABASE PARAMETER(Client Log Recording;1) //côté distant
```

Cette instruction démarre également le fichier d'historique [4DRequestsLog.txt](#).

En-têtes

Ce fichier commence avec les en-têtes suivants :

- Log Session Identifier (Identifiant de session d'historique)
- Nom du serveur qui héberge l'application
- User Login Name : Nom de l'utilisateur (défini dans l'OS) qui a exécuté l'application 4D sur le serveur.

Contenu

Pour chaque process, les champs suivants sont enregistrés :

Nom des champs	Description
sequence_number	Numéro d'opération séquentiel et unique dans la session d'historique
time	Date et heure au format ISO 8601 : "YYYY-MM-DDTHH:MM:SS.mmm"
process_info_index	Numéro de process séquentiel et unique
CDB4DBaseContext	UUID du contexte de base du composant DB4D
systemid	ID système
server_process_id	ID du process sur le serveur
remote_process_id	ID du process sur le client
process_name	Nom du process
cID	Identifiant de la connexion 4D
uID	Identifiant du client 4D
IP Client	Adresse IPv4/IPv6 du client
host_name	Nom d'hôte du client
user_name	Nom de connexion utilisateur sur le client
connection_uuid	Identifiant UUID de process de connexion
server_process_unique_id	ID unique du process sur le serveur

HTTPDebugLog.txt

Ce fichier d'historique enregistre chaque requête HTTP et chaque réponse en mode brut (raw). Les requêtes sont enregistrées dans leur totalité (en-têtes compris). Les parts du body peuvent optionnellement être enregistrées.

Pour démarrer ce fichier d'historique :

```
WEB SET OPTION(Web debug log;wdl enable without body)
//d'autres valeurs sont disponibles
```

Les champs suivants sont enregistrés pour chaque requête et réponse :

Nom des champs	Description
SocketID	ID du socket utilisé pour la communication
PeerIP	Adresse IPv4 de l'hôte (client)
PeerPort	Port utilisé par l'hôte (client)
TimeStamp	Horodatage en millisecondes (depuis le démarrage du système)
ConnectionID	Connexion UUID (UUID du VTCPSocket utilisé pour la communication)
SequenceNumber	Numéro d'opération séquentiel et unique dans la session d'historique

4DDebugLog.txt (standard)

Ce fichier d'historique enregistre chaque évènement qui a lieu au niveau de la programmation 4D. Le mode standard offre une vue basique des événements.

Pour démarrer ce fichier d'historique :

```

SET DATABASE PARAMETER(Debug Log Recording;2)
//standard, tous les process

SET DATABASE PARAMETER(Current process debug log recording;2)
//standard, process courant uniquement

```

Les champs suivants sont enregistrés pour chaque événement :

Colonne #	Description
1	Numéro d'opération séquentiel et unique dans la session d'historique
2	Date et heure au format ISO 8601 (YYYY-MM-DDThh:mm:ss.mmm)
3	ID process (p=xx) et ID unique process (puid=xx)
4	Niveau de stack (pile)
5	Peut être Nom de commande / Nom de méthode / Message / Info Start Stop task / Nom, événement ou callback plugin / UUID connexion
6	Durée de l'opération de connexion en millisecondes (différent 2e colonne)

4DDebugLog.txt (tabulé)

Ce fichier d'historique enregistre chaque événement au niveau de la programmation 4D sous un format compact tabulé incluant des informations supplémentaires par rapport au format standard.

Pour démarrer ce fichier d'historique :

```

SET DATABASE PARAMETER(Debug Log Recording;2+4)
//format tabulé étendu, tous les process

SET DATABASE PARAMETER(Current process debug log recording;2+4)
//étendu, process courant uniquement

```

Les champs suivants sont enregistrés pour chaque événement :

Colonne #	Nom des champs	Description
1	sequence_number	Numéro d'opération séquentiel et unique dans la session d'historique
2	time	Date et heure au format ISO 8601 (YYYY-MM-DDThh:mm:ss.mmm)
3	ProcessID	ID du process
4	unique_processID	ID unique du process
5	stack_level	Niveau de stack (pile)
6	operation_type	<p>Type d'opération enregistrée. Il peut s'agir d'une valeur absolue :</p> <ol style="list-style-type: none"> 1. Commande 2. Méthode (méthode projet, méthode base, etc.) 3. Message (uniquement envoyé par la commande LOG EVENT) 4. PluginMessage 5. PluginEvent 6. PluginCommand 7. PluginCallback 8. Task (Process) 9. Méthode membre (méthode attachée à une collection ou un objet) <p>Lors de la fermeture d'un niveau de stack, les colonnes <code>operation_type</code>, <code>operation</code> et <code>operation_parameters</code> ont la même valeur que le niveau de stack d'ouverture enregistré dans la colonne <code>stack_opening_sequence_number</code>. Par exemple :</p> <ol style="list-style-type: none"> 1. 121 15:16:50:777 5 8 1 2 CallMethod Parameters 0 2. 122 15:16:50:777 5 8 2 1 283 0 3. 123 15:16:50:777 5 8 2 1 283 0 122 3 4. 124 15:16:50:777 5 8 1 2 CallMethod Parameters 0 121 61 <p>Les 1re et 2e lignes ouvrent un niveau de stack, les 3e et 4e lignes ferment un niveau de stack. Les valeurs des colonnes 6, 7 et 8 sont répétées dans la ligne du niveau de stack de fermeture. La colonne 10 contient les numéros de séquence d'ouverture du niveau de stack, c'est-à-dire 122 pour la 3e ligne et 121 pour la 4e.</p>
7	operation	Peut représenter (en fonction du type d'opération) : <ul style="list-style-type: none"> • un ID de commande du langage (lorsque type=1) • un nom de méthode (lorsque type=2) • une combinaison de pluginIndex;pluginCommand (lorsque type=4, 5, 6 ou 7). Peut contenir des éléments tels que '3;2' • un UUID de connexion de process (lorsque type=8)
8	operation_parameters	Paramètres passés aux commandes, méthodes ou aux plugins
9	form_event	Evénement formulaire, le cas échéant ; vide dans les autres cas (par conséquent cette colonne est utilisée lorsque le code est exécuté dans une méthode formulaire ou méthode objet)
10	stack_opening_sequence_number	Niveaux de fermeture de stacks uniquement : numéro de séquence du niveau d'ouverture de stack correspondant
11	stack_level_execution_time	Niveaux de fermeture de stacks uniquement : Durée en micro secondes de l'action enregistrée courante (cf. 10e colonne des lignes 123 et 124 dans l'historique ci-dessus)

4DDiagnosticLog.txt

Ce fichier d'historique enregistre de nombreux événements liés au fonctionnement interne de l'application et est lisible par un humain. Vous pouvez inclure des informations personnalisées dans ce fichier à l'aide de la commande [LOG EVENT](#).

Pour démarrer ce fichier d'historique :

```
SET DATABASE PARAMETER(Diagnostic log recording;1) //lancer l'enregistrement
```

Les champs suivants sont enregistrés pour chaque événement :

Nom des champs	Description
sequenceNumber	Numéro d'opération séquentiel et unique dans la session d'historique
timestamp	Date et heure au format ISO 8601 (YYYY-MM-DDThh:mm:ss.mmm)
loggerID	Optionnel
componentSignature	Optionnel - signature de composant interne
messageLevel	Info, Attention, Erreur
message	Description de la saisie de journal

En fonction de l'événement, d'autres champs peuvent également être enregistrés, tels que task, socket, etc.

Activation du fichier

The *4DDiagnosticLog.txt* file can log different levels of messages, from `ERROR` (most important) to `TRACE` (less important). By default, the `INFO` level is set, which means that the file will log only important events, including errors and unexpected results (see below).

You can select the level of messages using the `Diagnostic log level` selector of the [SET DATABASE PARAMETER](#) command, depending on your needs. When you select a level, levels above (which are more important) are implicitly selected also. The following levels are available:

Colonne #	Description	When selected, includes
ERROR	Numéro d'opération séquentiel et unique dans la session d'historique	ERROR
WARN	Date et heure au format RFC3339 (yyyy-mm-ddThh:mm:ss.ms)	ERROR, WARN
INFO	ID du Process 4D	ERROR, WARN, INFO
DEBUG	ID unique du process	ERROR, WARN, INFO, DEBUG
TRACE	Other internal information (for 4D technical services)	ERROR, WARN, INFO, DEBUG, TRACE

4DSMTPLog.txt, 4DPOP3Log.txt, et 4DIMAPLog.txt

Ces fichiers d'historique enregistrent chaque échange entre l'application 4D et le serveur de mail (SMTP, POP3, IMAP) initialisé par les commandes suivantes :

- SMTP - [SMTP New transporter](#)
- POP3 - [POP3 New transporter](#)
- IMAP - [IMAP New transporter](#)

Les fichiers d'historique peuvent être générés en deux versions :

- une version classique :

- fichiers nommés 4DSMTPLLog.txt, 4DPOP3Log.txt, ou 4DIMAPLog.txt
- sans pièces jointes
- avec un recyclage automatique tous les 10 MB
- conçue pour des fonctions de débogage habituelles

Pour lancer cet historique :

```
SET DATABASE PARAMETER(SMTP Log;1) //lancer log SMTP
SET DATABASE PARAMETER(POP3 Log;1) //lancer log POP3
SET DATABASE PARAMETER(IMAP Log;1) //lancer log IMAP
```

4D Server : Cliquez sur le bouton Démarrer les journaux de requêtes et de débogage dans la [Page Maintenance](#) de la fenêtre d'administration de 4D Server.

Ce chemin d'accès au journal est retourné par la commande `Get 4D file`.

- une version étendue :

- pièce(s) jointe(s) inclue(s)
- nom personnalisé
- conçue à des fins spécifiques

Pour lancer cet historique :

```
$server:=New object
...
//SMTP
$server.logFile:="MySMTPAuthLog.txt"
$transporter:=SMTP New transporter($server)

// POP3
$server.logFile:="MyPOP3AuthLog.txt"
$transporter:=POP3 New transporter($server)

//IMAP
$server.logFile:="MyIMAPAuthLog.txt"
$transporter:=IMAP New transporter($server)
```

Contenu

Pour chaque requête, les champs suivants sont enregistrés :

Colonne #	Description
1	Numéro d'opération séquentiel et unique dans la session d'historique
2	Date et heure au format RFC3339 (yyyy-mm-ddThh:mm:ss.ms)
3	ID du Process 4D
4	ID unique du process
5	<ul style="list-style-type: none"> Informations sur le lancement d'une session SMTP, POP3 ou IMAP, y compris le nom d'hôte du serveur, le numéro de port TCP utilisé pour se connecter au serveur SMTP, POP3 ou IMAP et l'état TLS, ou données échangées entre le serveur et le client, en commençant par "S <" (données reçues depuis le serveur SMTP, POP3 ou IMAP) ou "C>" (données envoyées par le client IMAP) : liste des modes d'authentification envoyés par le serveur et mode d'authentification sélectionné, toute erreur signalée par le serveur SMTP, POP3 ou IMAP, les informations sur l'en-tête de l'e-mail envoyé (version standard uniquement) et si l'e-mail est sauvegardé sur le serveur, ou Les informations sur la clôture de la session IMAP.

Requêtes client ORDA

Ce journal enregistre chaque requête ORDA envoyée depuis une machine distante. Vous pouvez diriger les informations du journal vers la mémoire ou vers un fichier sur le disque. Vous pouvez choisir le nom et l'emplacement de ce fichier journal.

Pour démarrer ce fichier d'historique :

```
//à exécuter sur une machine distante
ds.startRequestLog(File("/PACKAGE/Logs/ordaLog.txt"))
//peut aussi être envoyé à la mémoire
```

Si vous souhaitez utiliser la numérotation séquentielle unique dans le fichier des requêtes ORDA, vous devez l'activer :

```
// à exécuter sur une machine distante

SET DATABASE PARAMETER(Client Log Recording;1)
//pour activer la numérotation automatique de l'historique

ds.startRequestLog(File("/PACKAGE/Logs/ordaLog.txt"))
//peut aussi être envoyé à la mémoire

SET DATABASE PARAMETER(Client Log Recording;0)
//désactive la numérotation automatique
```

Les champs suivants sont enregistrés pour chaque requête :

Nom des champs	Description	Exemple
sequenceNumber	Numéro d'opération séquentiel et unique dans la session d'historique	104
url	URL de la requête ORDA effectuée par le poste client	"rest/Persons(30001)"
startTime	Date et heure de début au format ISO 8601	"2019-05-28T08:25:12.346Z"
endTime	Date et heure de fin au format ISO 8601	"2019-05-28T08:25:12.371Z"
duration	Durée de traitement client (ms)	25
response	Objet réponse du serveur	{"status":200,"body": {"__entityModel":"Persons", [...]}}

Utilisation d'un fichier de configuration de log

Vous pouvez utiliser un fichier de configuration de log pour gérer facilement l'enregistrement des journaux dans un environnement de production. Ce fichier est préconfiguré par le développeur. En général, il peut être envoyé aux clients pour qu'ils n'aient qu'à le sélectionner ou à le copier dans un dossier local. Une fois activé, le fichier de configuration de log déclenche l'enregistrement de journaux spécifiques.

Activation du fichier

Il existe plusieurs façons d'activer le fichier de configuration des logs :

- Sur le serveur 4D avec interface, vous pouvez ouvrir la page Maintenance et cliquer sur le bouton [Charger le fichier de configuration des logs, puis sélectionner le fichier](#). Dans ce cas, vous pouvez utiliser n'importe quel nom pour le fichier de configuration. Il est immédiatement activé sur le serveur.
- Vous pouvez copier le fichier de configuration du journal dans le [dossier Settings](#) du projet. Dans ce cas, le fichier doit être nommé `logConfig.json`. Il est activé au démarrage du projet (uniquement sur le serveur en client/serveur).
- Avec une application générée, vous pouvez copier le fichier `logConfig.json` dans le dossier suivant :
 - Windows : `Users\[userName]\AppData\Roaming\[application]`
 - macOS : `/Users/[userName]/Library/ApplicationSupport/[application]`

Si vous souhaitez activer le fichier de configuration log pour tous les projets dans les applications autonomes, serveur et distantes, copiez le fichier `logConfig.json` dans le dossier suivant : - Windows: `Users\[userName]\AppData\Roaming\4D ou \4D Server` - macOS: `/Users/[userName]/Library/ApplicationSupport/4D ou /4D Server`

Description du fichier JSON

Le fichier de configuration de log est un fichier `.json` qui peut contenir les propriétés suivantes :

```
{
  "$$schema": "http://json-schema.org/draft-07/schema",
  "title": "Logs Configuration File",
  "description": "A file that controls the state of different types of logs in 4D clients and servers",
  "type": "object",
  "properties": {
    "forceLoggingConfiguration": {
      "description": "Forcing the logs configuration described in the file ingoring changes coming",
      "type": "boolean",
      "default": true
    },
    "requestLogs": {
      "description": "Configuration for request logs",
      "type": "object",
      "properties": {
        "type": "string"
      }
    }
  }
}
```

```

"properties": {
    "clientState": {
        "description": "Enable/Disable client request logs (from 0 to N)",
        "type": "integer",
        "minimum": 0
    },
    "serverState": {
        "description": "Enable/Disable server request logs (from 0 to N)",
        "type": "integer",
        "minimum": 0
    }
},
"debugLogs": {
    "description": "Configuration for debug logs",
    "type": "object",
    "properties": {
        "commandList": {
            "description": "Commands to log or not log",
            "type": "array",
            "items": {
                "type": "string"
            },
            "minItems": 1,
            "uniqueItems": true
        },
        "state": {
            "description": "integer to specify type of debuglog and options",
            "type": "integer",
            "minimum": 0
        }
    }
},
"diagnosticLogs": {
    "description": "Configuration for debug logs",
    "type": "object",
    "properties": {
        "state": {
            "description": "Enable/Disable diagnostic logs 0 or 1 (0 = do not record, 1 = record",
            "type": "integer",
            "minimum": 0
        }
    }
},
"httpDebugLogs": {
    "description": "Configuration for http debug logs",
    "type": "object",
    "properties": {
        "level": {
            "description": "Configure http request logs",
            "type": "integer",
            "minimum": 0,
            "maximum": 7
        },
        "state": {
            "description": "Enable/Disable recording of web requests",
            "type": "integer",
            "minimum": 0,
            "maximum": 4
        }
    }
},
"POP3Logs": {
    "description": "Configuration for POP3 logs",
    "type": "object",
    "properties": {
        "state": {
            "description": "Enable/Disable recording of POP3 logs",
            "type": "integer",
            "minimum": 0,
            "maximum": 4
        }
    }
}
}

```

```
"type": "object",
"properties": {
    "state": {
        "description": "Enable/Disable POP3 logs (from 0 to N)",
        "type": "integer",
        "minimum": 0
    }
},
"SMTPLogs": {
    "description": "Configuration for SMTP logs",
    "type": "object",
    "properties": {
        "state": {
            "description": "Enable/Disable SMTP log recording (form 0 to N)",
            "type": "integer",
            "minimum": 0
        }
    }
},
"IMAPLogs": {
    "description": "Configuration for IMAP logs",
    "type": "object",
    "properties": {
        "state": {
            "description": "Enable/Disable IMAP log recording (form 0 to N)",
            "type": "integer"
        }
    }
},
"ORDALogs": {
    "description": "Configuration for ORDA logs",
    "type": "object",
    "properties": {
        "state": {
            "description": "Enable/Disable ORDA logs (0 or 1)",
            "type": "integer"
        },
        "filename": {
            "type": "string"
        }
    }
}
}
```

Exemple

Voici un exemple de fichier de configuration de log :

```
{  
    "forceLoggingConfiguration": false,  
    "requestLogs": {  
        "clientState": 1,  
        "serverState": 1  
    },  
    "debugLogs": {  
        "commandList": ["322", "311", "112"],  
        "state": 4  
    },  
    "diagnosticLogs": {  
        "state" : 1  
    },  
    "httpDebugLogs": {  
        "level": 5,  
        "state" : 1  
    },  
    "POP3Logs": {  
        "state" : 1  
    },  
    "SMTPLogs": {  
        "state" : 1  
    },  
    "IMAPLogs": {  
        "state" : 1  
    },  
    "ORDALogs": {  
        "state" : 1,  
        "filename": "ORDALog.txt"  
    }  
}
```

Balises de transformation

4D fournit un ensemble de balises de transformation qui vous permettent d'insérer des références à des variables ou des expressions 4D, ou d'effectuer différents types de traitement dans un texte source, appelé "template". Ces balises sont interprétées lors de l'exécution du texte source et génèrent un texte de sortie.

Ce principe est notamment utilisé par le serveur Web 4D pour créer [des pages de template Web](#).

Ces balises sont généralement insérées sous forme de commentaires de type HTML (`<!--#Tag Contents-->`) mais [une syntaxe alternative conforme à la norme xml](#) est disponible pour certaines d'entre elles.

Il est possible de combiner plusieurs types de balises. Par exemple, la structure HTML suivante est tout à fait envisageable :

```
<HTML>
<BODY>
<!--#4DSCRIPT/PRE_PROCESS-->    (Method call)
<!--#4DIF (myvar=1)-->    (If condition)
    <!--#4DINCLUDE banner1.html-->    (Subpage insertion)
<!--#4DENDIF-->    (End if)
<!--#4DIF (mtvar=2)-->
    <!--#4DINCLUDE banner2.html-->
<!--#4DENDIF-->

<!--#4DLOOP [TABLE]-->    (Loop on the current selection)
<!--#4DIF ([TABLE]ValNum>10)-->    (If [TABLE]ValNum>10)
    <!--#4DINCLUDE subpage.html-->    (Subpage insertion)
<!--#4DELSE-->    (Else)
    <B>Value: <!--#4DTEXT [TABLE]ValNum--></B><BR>    (Field display)
<!--#4DENDIF-->
<!--#4DENDLOOP-->    ](End for)
</BODY>
</HTML>
```

Principes d'utilisation des balises

Parsing

L'analyse ou le parsing du contenu d'une source de *template* se fait dans deux contextes :

- En utilisant la commande `PROCESS 4D TAGS` ; cette commande accepte un *template* en entrée, ainsi que des paramètres optionnels et retourne un texte résultant du traitement.
- En utilisant le serveur HTTP intégré de 4D : [pages de templates](#) envoyées au moyen des commandes `WEB SEND FILE` (.htm, .html, .shtm, .shtml), `WEB SEND BLOB` (text/html type BLOB), `WEB SEND TEXT`, ou appelées au moyen d'une URL. Dans ce dernier cas, à des fins d'optimisation, les pages suffixées par ".htm" et ".html" ne sont PAS parsées. In this last case, for reasons of optimization, pages that are suffixed with ".htm" and ".html" are NOT parsed.

Traitement récursif

Les balises 4D sont interprétées de manière récursive : 4D tente toujours de réinterpréter le résultat d'une transformation et, si une nouvelle transformation a eu lieu, une interprétation supplémentaire est effectuée, et ainsi de suite jusqu'à ce que le produit obtenu ne nécessite plus aucune transformation. Par exemple, étant donné l'énoncé suivant :

```
<!--#4DHTML [Mail]Letter_type-->
```

Si le champ de texte `[Mail]Letter_type` contient lui-même une balise, telle que `<!--#4DSCRIPT/m_Gender-->`, cette balise sera évaluée récursivement après l'interprétation de la balise 4DHTML.

Ce principe puissant répond à la plupart des besoins liés à la transformation des textes. Notez toutefois que, dans certains cas, cela peut également permettre l'insertion d'un code malveillant dans le contexte web, [ce qui peut être évité](#).

Identifiants avec tokens

Pour garantir l'évaluation correcte des expressions traitées via les balises, quel que soit le langage ou la version de 4D, il est recommandé d'utiliser la syntaxe tokenisée pour les éléments dont le nom peut varier selon les versions (commandes, tables, champs, constantes). Par exemple, pour insérer la commande `Current time`, entrez `Current time:C178`.

Utilisation du "." comme séparateur décimal

4D utilise toujours le caractère point (.) comme séparateur décimal lorsqu'il évalue une expression numérique à l'aide d'une balise `4DTEXT`, `4DHTML`, et `4DEVAL`. Les paramètres régionaux sont ignorés. Cette fonctionnalité facilite la maintenance du code et la compatibilité entre les langues et les versions de 4D.

4DBASE

Syntax: `<!--#4DBASE folderPath-->`

La balise `<!--#4DBASE -->` désigne le répertoire de travail à utiliser par la balise `<!--#4DINCLUDE-->`.

Lorsqu'elle est appelée dans une page Web, la balise `<!--#4DBASE -->` modifie tous les appels `<!--#4DINCLUDE-->` ultérieurs sur cette page, jusqu'au prochain `<!--#4DBASE -->`, le cas échéant. Si le dossier est modifié à partir d'un fichier inclus, il récupère sa valeur originale dans le fichier parent.

Le paramètre `folderPath` doit contenir un nom de chemin relatif à la page actuelle et il doit se terminer par une barre oblique (/). Le dossier désigné doit être situé à l'intérieur du dossier Web.

Passez le mot-clé "WEBFOLDER" pour rétablir le chemin par défaut (relatif à la page).

Le code suivant, qui doit spécifier un chemin relatif pour chaque appel :

```
<!--#4DINCLUDE subpage.html-->
<!--#4DINCLUDE folder/subpage1.html-->
<!--#4DINCLUDE folder/subpage2.html-->
<!--#4DINCLUDE folder/subpage3.html-->
<!--#4DINCLUDE ../folder/subpage.html-->
```

...équivaut à :

```
<!--#4DINCLUDE subpage.html-->
<!--#4DBASE folder-->
<!--#4DINCLUDE subpage1.html-->
<!--#4DINCLUDE subpage2.html-->
<!--#4DINCLUDE subpage3.html-->
<!--#4DBASE ../folder-->
<!--#4DINCLUDE subpage.html-->
<!--#4DBASE WEBFOLDER-->
```

Par exemple, pour définir un répertoire pour la page d'accueil :

```
/* Index.html */
<!--#4DIF LangFR=True-->
    <!--#4DBASE FR/-->
<!--#4DELSE-->
    <!--#4DBASE US/-->
<!--#4DENDIF-->
<!--#4DINCLUDE head.html-->
<!--#4DINCLUDE body.html-->
<!--#4DINCLUDE footer.html-->
```

Dans le fichier "head.html", le dossier courant est modifié par `<!--#4DBASE -->`, sans que cela ne change sa valeur dans "Index.html" :

```
/* Head.htm */
/* the working directory here is relative to the included file (FR/ or US/) */
<!--#4DBASE Styles-->
<!--#4DINCLUDE main.css-->
<!--#4DINCLUDE product.css-->
<!--#4DBASE Scripts-->
<!--#4DINCLUDE main.js-->
<!--#4DINCLUDE product.js-->
```

4DCODE

Syntax: `<!--#4DCODE codeLines-->`

La balise `4DCODE` permet d'insérer un bloc de code 4D de plusieurs lignes dans un template.

Lorsqu'une séquence `<!--#4DCODE` est détectée et qu'elle est suivie d'un espace, d'un caractère CR ou LF, 4D interprète toutes les lignes de code jusqu'à la séquence `-->` suivante. Le bloc de code lui-même peut contenir des retours chariot, des sauts de ligne ou les deux ; il sera interprété séquentiellement par 4D.

Par exemple, vous pouvez écrire dans un template :

```
<!--#4DCODE
//PARAMETERS initialization
C_OBJECT:C1216($graphParameters)
OB SET:C1220($graphParameters;"graphType";1)
$graphType:=1
//...your code here
If(OB Is defined:C1231($graphParameters;"graphType"))
    $graphType:=OB GET:C1224($graphParameters;"graphType")
    If($graphType=7)
        $nbSeries:=1
        If($nbValues>8)
            DELETE FROM ARRAY:C228 ($yValuesArrPtr{1}->;9;100000)
            $nbValues:=8
        End if
    End if
End if
-->
```

Voici les caractéristiques de la balise `4DCODE` :

- La commande `TRACE` est prise en charge et active le débogueur 4D, vous permettant ainsi de déboguer le code de votre modèle.
- Toute erreur affichera le dialogue d'erreur standard qui permet à l'utilisateur d'arrêter l'exécution du code ou d'entrer en mode débogage.

- Le texte entre `<!--#4DCODE` et `-->` est divisé en lignes acceptant toute convention de fin de ligne (cr, lf ou crlf).
- Le texte est tokenisé dans le contexte de la base de données qui a appelé `PROCESS 4D TAGS`. Ceci est important pour la reconnaissance des méthodes de projet par exemple. La propriété de la méthode [Available through tags](#) et [4D URLs \(4DACTION ...\)](#) n'est pas prise en compte.
- Même si le texte utilise toujours l'anglais-US, il est recommandé d'utiliser la syntaxe `:Cxxx` pour les noms de commandes et de constantes afin de se prémunir contre d'éventuels problèmes dus à des commandes ou des constantes renommées d'une version de 4D à une autre.

Le fait que les balises 4DCODE puissent appeler n'importe quelle commande du langage 4D ou méthode du projet pourrait être considéré comme un problème de sécurité, en particulier lorsque la base de données est disponible via HTTP. Toutefois, étant donné qu'elle exécute du code côté serveur appelé à partir de vos propres fichiers de modèle, la balise elle-même ne représente pas un problème de sécurité. Dans ce contexte, comme pour tout serveur Web, la sécurité est principalement gérée au niveau des accès distants aux fichiers du serveur.

4DEACH et 4DENDEACH

Syntax: `<!--#4DEACH variable in expression--> <!--#4DENDEACH-->`

Le commentaire `<!--#4DEACH-->` permet d'itérer un élément spécifié sur toutes les valeurs de l' *expression*. L'élément est défini comme une *variable* dont le type dépend du type d'*expression*.

Le commentaire `<!--#4DEACH-->` peut itérer sur trois types d'*expression* :

- [collections](#) : boucle à travers chaque élément de la collection,
- [entity selections](#) : boucle à travers chaque entity,
- [objets](#) : boucle à travers chaque propriété d'objet.

Le nombre d'itérations est évalué au démarrage et ne changera pas pendant le traitement. L'ajout ou la suppression d'éléments pendant la boucle est donc déconseillé car il pourra en résulter une redondance ou un manque d'itérations.

`<!--#4DEACH item in collection-->`

Cette syntaxe permet d'effectuer une itération sur chaque *élément* de la *collection*. La portion de code située entre `<!--#4DEACH -->` et `<!--#4DENDEACH-->` est répétée pour chaque élément de la collection.

Le paramètre de l'élément est une variable du même type que les éléments de la collection.

La collection ne doit contenir que des éléments de même type, sinon une erreur est retournée dès que la variable de l'élément se voit attribuer le premier type de valeur non concordant.

Le nombre de boucles est basé sur le nombre d'éléments de la collection. À chaque itération, la variable de l'élément correspond automatiquement à l'élément correspondant de la collection. Vous devez tenir compte des points suivants :

- The *item* variable gets the same type as the first collection element. Si un seul élément de la collection n'est pas du même type que la variable, une erreur est générée et la boucle s'arrête.
- If the *item* variable is of the object type or collection type (i.e. Si un seul élément de la collection n'est pas du même type que la variable, une erreur est générée et la boucle s'arrête.
- Si la collection contient des éléments de valeur Null, une erreur sera générée si le type de la variable ne prend pas en charge la valeur Null (comme par exemple les variables entier long).

Exemple avec une collection de valeurs scalaires

`getNames` retourne une collection de chaînes de caractères. La méthode a été déclarée comme "[disponible via les balises 4D et les URL](#)".

```


| Name           |
|----------------|
| #4DTEXT \$name |


```

Exemple avec une collection d'objets

`getSalesPersons` retourne une collection d'objets.

```


|                          |                                 |                                |
|--------------------------|---------------------------------|--------------------------------|
| #4DTEXT \$salesPerson.ID | #4DTEXT \$salesPerson.firstname | #4DTEXT \$salesPerson.lastname |
|--------------------------|---------------------------------|--------------------------------|


```

```
<!--#4DEACH entity in entitySelection-->
```

Cette syntaxe permet d'effectuer une itération sur chaque `entity` de l'`entitySelection`. La portion de code située entre `<!--#4DEACH -->` et `<!--#4DENDEACH-->` est répétée pour chaque entity de l'entity selection.

Le paramètre `entity` est une variable objet de la classe `entity selection`.

Le nombre de boucles est basé sur le nombre d'entities présentes dans l'`entity selection`. À chaque itération, la variable de l'objet `entity` correspond automatiquement à l'`entity` de l'`entity selection`.

Exemple avec un tableau html

```


| ID | Name | Total purchase |
|----|------|----------------|
|----|------|----------------|


```

Exemple avec des `BALISES 4D PROCESS`

```

var customers : cs.CustomersSelection
var $input; $output : Text

customers:=ds.Customers.all()
$input:="!--#4DEACH $cust in customers-->"
$input:=$input+"<!--#4DTEXT $cust.name -->"+Char(Carriage return)
$input:=$input+"<!--#4DENDEACH-->"
PROCESS 4D TAGS($input; $output)
TEXT TO DOCUMENT("customers.txt"; $output)

```

<!--#4DEACH property in object-->

This syntax iterates on each *property* of the *object*. The code portion located between `<!--#4DEACH -->` and `<!--#4DENDEACH-->` is repeated for each property of the object.

The *property* parameter is a text variable automatically filled with the name of the currently processed property.

The properties of the object are processed according to their creation order. Pendant la boucle, il est possible d'ajouter ou de supprimer des propriétés dans l'objet, sans pour autant modifier le nombre de boucles qui reste basé sur le nombre de propriétés initial de l'objet.

Example with the properties of an object

`getGamers` is a project method that returns an object like ("Mary"; 10; "Ann"; 20; "John"; 40) to figure gamer scores.

```





```

4DEVAL

Syntax: `<!--#4DEVAL expression-->`

Alternative syntax: `$4DEVAL(expression)`

The `4DEVAL` tag allows you to assess a 4D variable or expression. Like the `4DHTML` tag, `4DEVAL` does not escape HTML characters when returning text. However, unlike `4DHTML` or `4DTEXT`, `4DEVAL` allows you to execute any valid 4D statement, including assignments and expressions that do not return any value.

For example, you can execute:

```

$input:="!--#4DEVAL a:=42-->" //assignment
$input:=$input+"<!--#4DEVAL a+1-->" //calculation
PROCESS 4D TAGS($input;$output)
//$/output = "43"

```

In case of an error during interpretation, the text inserted will be in the form: `<!--#4DEVAL expr-->: ## error # error code .`

For security reasons, it is recommended to use the `4DTEXT` tag when processing data introduced from outside the application, in order to prevent the [insertion of malicious code](#).

4DHTML

Syntax: `<!--#4DHTML expression-->`

Alternative syntax: `$4DHTML(expression)`

The value of the 4D variable `vtSiteName` will be inserted in the HTML page when it is sent. This value is inserted as simple text, special HTML characters such as ">" are automatically escaped.

For example, here are the processing results of the 4D text variable `myvar` with the available tags:

myvar Value	Tags	Résultat
<code>myvar:=""</code>	<code><!--#4DTEXT myvar--></code>	<code>&lt;B&gt;</code>
<code>myvar:=""</code>	<code><!--#4DHTML myvar--></code>	<code></code>

In case of an interpretation error, the inserted text will be `<!--#4DHTML myvar--> : ## error # error code .`

For security reasons, it is recommended to use the `4DTEXT` tag when processing data introduced from outside the application, in order to prevent the [insertion of malicious code](#).

4DIF, 4DELSE, 4DELSEIF et 4DENDIF

Syntax: `<!--#4DIF expression--> { <!--#4DELSEIF expression2-->...<!--#4DELSEIF expressionN--> } <!--#4DELSE--> } <!--#4DENDIF-->`

Used with the `<!--#4DELSEIF-->` (optional), `<!--#4DELSE-->` (optional) and `<!--#4DENDIF-->` comments, the `<!--#4DIF expression-->` comment offers the possibility to execute portions of code conditionally.

The `expression` parameter can contain any valid 4D expression returning a Boolean value. It must be indicated within parenthesis and comply with the 4D syntax rules.

In case of an interpretation error, the text " `<!--#4DIF expression--> : A Boolean expression was expected`" is inserted instead of the contents located between `<!--#4DIF -->` and `<!--#4DENDIF-->`. Likewise, if there are not as many `<!--#4DENDIF-->` as `<!--#4DIF -->`, the text " `<!--#4DIF expression--> : 4DENDIF expected`" is inserted instead of the contents located between `<!--#4DIF -->` and `<!--#4DENDIF-->`.

In case of an interpretation error, the text " `<!--#4DIF expression--> : A Boolean expression was expected`" is inserted instead of the contents located between `<!--#4DIF -->` and `<!--#4DENDIF-->`. The `<!--#4DIF expression--> ... <!--#4DENDIF-->` blocks can be nested in several levels. Like in 4D, each `<!--#4DIF expression-->` must match a `<!--#4DENDIF-->`.

Using the `<!--#4DELSEIF-->` tag, you can test an unlimited number of conditions. Only the code that follows the first condition evaluated as `True` is executed. If no conditions are true, no statement is executed (if there is no final `<!--#4DELSE-->`). You can use a tag after the last . If all the conditions are false, the statements following the are executed.

The two following codes are equivalent.

Code using 4DELSE only:

```

<!--#4DIF Condition1-->
/* Condition1 is true*/
<!--#4ELSE-->
<!--#4DIF Condition2-->
/* Condition2 is true*/
<!--#4ELSE-->
<!--#4DIF Condition3-->
/* Condition3 is true */
<!--#4ELSE-->
/*None of the conditions are true*/
<!--#4ENDIF-->
<!--#4ENDIF-->
<!--#4ENDIF-->

```

Similar code using the `4ELSEIF` tag:

```

<!--#4DIF Condition1-->
/* Condition1 is true*/
<!--#4ELSEIF Condition2-->
/* Condition2 is true*/
<!--#4ELSEIF Condition3-->
/* Condition3 is true */
<!--#4ELSE-->
/* None of the conditions are true*/
<!--#4ENDIF-->

```

This example of code inserted in a static HTML page displays a different label according the `vname#""` expression result:

```

<BODY>
...
<!--#4DIF (vname#"")-->
Names starting with <!--#4DTEXT vname-->.
<!--#4ELSE-->
No name has been found.
<!--#4ENDIF-->
...
</BODY>

```

This example inserts different pages depending on which user is connected:

```

<!--#4DIF LoggedIn=False-->
    <!--#4DINCLUDE Login.htm -->
<!--#4ELSEIF User="Admin" -->
    <!--#4DINCLUDE AdminPanel.htm -->
<!--#4ELSEIF User="Manager" -->
    <!--#4DINCLUDE SalesDashboard.htm -->
<!--#4ELSE-->
    <!--#4DINCLUDE ItemList.htm -->
<!--#4ENDIF-->

```

4DINCLUDE

Syntax: `<!--#4DINCLUDE path-->`

This tag is mainly designed to include an HTML page (indicated by the `path` parameter) in another HTML page. By default, only the body of the specified HTML page, i.e. the contents found within the `<body>` and `</body>` tags, is

included (the tags themselves are not included). This lets you avoid conflicts related to meta tags present in the headers.

However, if the HTML page specified does not contain `<body>` `</body>` tags, the entire page is included. It is up to you to verify the consistency of the meta tags.

The `<!--#4DINCLUDE -->` comment is very useful for tests (`<!--#4DIF-->`) or loops (`<!--#4DL00P-->`). It is very convenient to include banners according to a criteria or randomly. When including, regardless of the file name extension, 4D analyzes the called page and then inserts the contents (modified or not) in the page originating the `4DINCLUDE` call.

An included page with the `<!--#4DINCLUDE -->` comment is loaded in the Web server cache the same way as pages called via a URL or sent with the `WEB SEND FILE` command.

In `path`, put the path leading to the document to include. Warning: In the case of a `4DINCLUDE` call, the path is relative to the document being analyzed, that is, the "parent" document. Use the slash character (/) as a folder separator and the two dots (..) to go up one level (HTML syntax). When you use the `4DINCLUDE` tag with the `PROCESS 4D TAGS` command, the default folder is the project folder.

You can modify the default folder used by the `4DINCLUDE` tag in the current page, using the `<!--#4DBASE -->` tag (see below).

The number of `<!--#4DINCLUDE path-->` within a page is unlimited. However, the `<!--#4DINCLUDE path-->` calls can be made only at one level. This means that, for example, you cannot insert `<!--#4DINCLUDE mydoc3.html-->` in the `mydoc2.html` body page, which is called by `<!--#4DINCLUDE mydoc2-->` inserted in `mydoc1.html`. Furthermore, 4D verifies that inclusions are not recursive.

In case of error, the inserted text is "`<!--#4DINCLUDE path-->` :The document cannot be opened".

Voici quelques exemples :

```
<!--#4DINCLUDE subpage.html-->
<!--#4DINCLUDE folder/subpage.html-->
<!--#4DINCLUDE ../folder/subpage.html-->
```

4DLOOP et 4DENDLOOP

Syntax: `<!--#4DL00P condition--> <!--#4DENDL00P-->`

This comment allows repetition of a portion of code as long as the condition is fulfilled. The portion is delimited by `<!--#4DL00P-->` and `<!--#4DENDL00P-->`.

The `<!--#4DL00P condition--> ... <!--#4DENDL00P-->` blocks can be nested. Like in 4D, each `<!--#4DL00P condition-->` must match a `<!--#4DENDL00P-->`.

There are five kinds of conditions:

```
<!--#4DL00P [table]-->
```

This syntax makes a loop for each record from the table current selection in the current process. The code portion located between the two comments is repeated for each current selection record.

When the `4DL00P` tag is used with a table, records are loaded in "Read only" mode.

Le code suivant :

```
<!--#4DLOOP [People]-->
<!--#4DTEXT [People]Name--> <!--#4DTEXT [People]Surname--><BR>
<!--#4DENDLOOP-->
```

... could be expressed in 4D language in the following way:

```
FIRST RECORD( [People])
While(Not(End selection([People])))
  ...
  NEXT RECORD( [People])
End while
```

```
<!--#4DLOOP array-->
```

This syntax makes a loop for each array item. The array current item is increased when each code portion is repeated.

This syntax cannot be used with two dimension arrays. In this case, it is better to combine a method with nested loops.

The following code example:

```
<!--#4DLOOP arr_names-->
<!--#4DTEXT arr_names{arr_names}--><BR>
<!--#4DENDLOOP-->
```

... could be expressed in 4D language in the following way:

```
For($Elem;1;Size of array(arr_names))
  arr_names:=$Elem
  ...
End for
```

```
<!--#4DLOOP method-->
```

This syntax makes a loop as long as the method returns `True`. The method takes a Long Integer parameter type. First it is called with the value 0 to allow an initialization stage (if necessary); it is then called with the values 1, then 2, then 3 and so on, as long as it returns `True`.

For security reasons, within a Web process, the `On Web Authentication` database method can be called once just before the initialization stage (method execution with 0 as parameter). If the authentication is OK, the initialization stage will proceed.

`C_BOOLEAN($0)` and `C_LONGINT($1)` MUST be declared within the method for compilation purposes.

The following code example:

```
<!--#4DLOOP my_method-->
<!--#4DTEXT var--> <BR>
<!--#4DENDLOOP-->
```

... could be expressed in 4D language in the following way:

```

If(AuthenticationWebOK)
  If(my_method(0))
    $counter:=1
    While(my_method($counter))
      ...
      $counter:=$counter+1
    End while
  End if
End if

```

The `my_method` method can be as follows:

```

C_LONGINT($1)
C_BOOLEAN($0)
If($1=0) `Initialisation
  $0:=True
Else
  If($1<50)
    ...
    var:...
    $0:=True
  Else
    $0:=False `Stops the loop
  End if
End if

```

`<!--#4DL00P expression-->`

With this syntax, the `4DL00P` tag makes a loop as long as the `expression` returns `True`. The expression can be any valid Boolean expression and must contain a variable part to be evaluated in each loop to avoid infinite loops.

For example, the following code:

```

<!--#4DEVAL $i:=0-->
<!--#4DL00P ($i<4)-->
<!--#4DEVAL $i-->
<!--#4DEVAL $i:=$i+1-->
<!--#4DENDLOOP-->

```

...produces the following result:

```

0
1
2
3

```

`<!--#4DL00P pointerArray-->`

In this case, the `4DL00P` tag works like it does with an array: it makes a loop for each element of the array referenced by the pointer. The current array element is increased each time the portion of code is repeated.

This syntax is useful when you pass an array pointer as a parameter to the `PROCESS 4D TAGS` command.

Exemple :

```

ARRAY TEXT($array;2)
$array{1}:="hello"
$array{2}:="world"
$input:="!--#4DEVAL $1-->"
$input:=$input+"!--#4DLOOP $2-->" 
$input:=$input+"!--#4DEVAL $2->{$2->}--> "
$input:=$input+"!--#4DENDLOOP-->" 
PROCESS 4D TAGS($input;$output;"elements = ";->$array)
// $output = "elements = hello world "

```

In case of an interpretation error, the text " <!--#4DLOOP expression--> : description" is inserted instead of the contents located between <!--#4DLOOP --> and <!--#4DENDLOOP--> .

The following messages can be displayed:

- Unexpected expression type (standard error);
- Incorrect table name (error on the table name);
- An array was expected (the variable is not an array or is a two dimension array);
- The method does not exist;
- Syntax error (when the method is executing);
- Access error (you do not have the appropriate access privileges to access the table or the method).
- 4DENDLOOP expected (the <!--#4DENDLOOP--> number does not match the <!--#4DLOOP -->).

4DSCRIPT/

Syntax: <!--#4DSCRIPT/MethodName/MyParam-->

The **4DSCRIPT** tag allows you to execute 4D methods when processing the template. The presence of the <!--#4DSCRIPT/MethodName/MyParam--> tag as an HTML comment launches the execution of the **MethodName** method with the **Param** parameter as a string in **\$1**.

If the tag is called in the context of a Web process, when the page is loaded, 4D calls the **On Web Authentication** database method (if it exists). If it returns True, 4D executes the method.

The method must return text in **\$0**. If the string starts with the code character 1, it is considered as HTML (the same principle is true for the **4DHTML** tag).

For example, let's say that you insert the following comment "Today is <!--#4DSCRIPT/MYMETH/MYPARAM-->" into a template Web page. When loading the page, 4D calls the **On Web Authentication** database method, then calls the **MYMETH** method and passes the string "/MYPARAM" as the parameter **\$1**. The method returns text in **\$0** (for example "12/31/21"); the expression " Today is <!--#4DSCRIPT/MYMETH/MYPARAM--> " therefore becomes "Today is 12/31/21".

The **MYMETH** method is as follows:

```

//MYMETH
C_TEXT($0;$1) //These parameters must always be declared
$0:=String(Current date)

```

A method called by **4DSCRIPT** must not call interface elements (**DIALOG**, **ALERT**, etc.).

As 4D executes methods in their order of appearance, it is absolutely possible to call a method that sets the value of many variables that are referenced further in the document, whichever mode you are using. You can insert as many <!--#4DSCRIPT...--> comments as you want in a template.

4DTEXT

Syntax: <!--#4DTEXT expression-->

Syntaxe alternative : 4DTEXT(expression)

La balise <!--#4DTEXT expression--> vous permet d'insérer une référence à une variable 4D ou à une expression retournant une valeur. Par exemple, si vous écrivez (dans une page HTML) :

```
<P>Welcome to <!--#4DTEXT vtSiteName-->!</P>
```

La valeur de la variable 4D vtSiteName sera insérée dans la page HTML lors de son envoi. Cette valeur est insérée comme du texte simple, les caractères HTML spéciaux tels que ">" sont automatiquement échappés.

Vous pouvez également insérer des expressions 4D. Par exemple, vous pouvez insérer directement le contenu d'un champ (<!--#4DTEXT [tableName] fieldName-->), un élément de tableau (<!--#4DTEXT tabarr{1}-->) ou une méthode retournant une valeur (<!--#4DTEXT mymethod-->). La conversion des expressions suit les mêmes règles que celles des variables. De plus, l'expression doit respecter les règles de syntaxe 4D.

Pour des raisons de sécurité, il est recommandé d'utiliser cette balise lors du traitement de données introduites en dehors de l'application, afin d'éviter [l'insertion de code malveillant](#).

En cas d'erreur d'évaluation, le texte inséré apparaîtra sous la forme <!--#4DTEXT myvar--> : ## erreur # code d'erreur .

- Vous devez utiliser des variables process.
- Vous pouvez afficher le contenu d'un champ image. Cependant, il n'est pas possible d'afficher le contenu d'un élément de tableau d'images.
- Il est possible d'afficher le contenu d'un champ objet à l'aide d'une formule 4D. Par exemple, vous pouvez écrire <!--#4DTEXT 0B Get:C1224([Rect]Desc;\\"color\\")--> .
- Vous travaillerez généralement avec des variables de type Texte. Cependant, vous pouvez également utiliser des variables BLOB. Il vous suffit de générer les BLOB en mode Texte sans longueur .

Syntaxe alternative pour 4DTEXT, 4DHTML, 4DEVAL

Plusieurs balises de transformation 4D existantes peuvent être exprimées à l'aide d'une syntaxe fondée sur le symbole \$:

4dtag (expression)

peut être utilisé à la place de

```
<!--#4dtag expression-->
```

Cette syntaxe alternative n'est disponible que pour les balises utilisées pour retourner des valeurs traitées :

- [4DTEXT](#)
- [4DHTML](#)
- [4DEVAL](#)

(Les autres balises, telles que 4DIF ou 4DSCRIPT, doivent être écrites avec la syntaxe régulière).

Par exemple, vous pouvez écrire :

```
$4DEVAL(UserName)
```

au lieu de :

```
<!--#4DEVAL(UserName)-->
```

Le principal avantage de cette syntaxe est qu'elle vous permet d'écrire des modèles conformes à XML. Certains développeurs 4D ont besoin de créer et de valider des modèles basés sur XML à l'aide d'outils d'analyse syntaxique XML standard. Le caractère "<" n'étant pas valide dans une valeur d'attribut XML, il n'était pas possible d'utiliser la syntaxe "" des balises 4D sans rompre la syntaxe du document. D'autre part, ne pas mentionner le caractère "<" empêche 4D d'interpréter correctement les balises.

Par exemple, le code suivant provoquerait une erreur d'analyse XML en raison du premier caractère "<" dans la valeur de l'attribut :

```
<line x1="<!--#4DEVAL $x-->" y1="<!--#4DEVAL $graphY1-->"/>
```

En utilisant la syntaxe \$, le code suivant est validé par le parseur :

```
<line x1="$4DEVAL($x)" y1="$4DEVAL($graphY1)"/>
```

A noter que \$4dtag et <--#4dtag --> ne sont pas strictement équivalents : contrairement à <--#4dtag -->, le traitement de \$4dtag n'interprète pas les balises 4D **de manière récursive**. Les balises \$ sont toujours évaluées une fois et le résultat est considéré comme du texte brut.

Cette différence consiste à empêcher l'injection de code malveillant. Comme [expliqué ci-dessous](#), il est fortement recommandé d'utiliser les balises 4DTEXT au lieu des balises 4DHTML lorsque vous manipulez du texte utilisateur, afin de se protéger contre une réinterprétation indésirable des balises : avec 4DTEXT, les caractères spéciaux tels que "<" ne sont pas mentionnés, ainsi toute balise 4D utilisant la syntaxe <!--#4dtag expression --> perdra sa signification particulière. Cependant, étant donné que 4DTEXT n'échappe pas au symbole ``\$, nous avons décidé de rompre la prise en charge de la récursion afin d'empêcher toute injection malveillante utilisant la syntaxe 4dtag (expression) .

Les exemples suivants illustrent le résultat du traitement en fonction de la syntaxe et du tag utilisés :

```
// exemple 1
myName:="<!--#4DHTML QUIT 4D-->" //injection malveillante
input:="My name is: <!--#4DHTML myName-->"
PROCESS 4D TAGS(input;output)
//4D va quitter !
```

```
// exemple 2
myName:="<!--#4DHTML QUIT 4D-->" //injection malveillante
input:="My name is: <!--#4DTEXT myName-->"
PROCESS 4D TAGS(input;output)
//la sortie est "My name is: <!--#4DHTML QUIT 4D-->"
```

```
// exemple 3
myName:="$4DEVAL(QUIT 4D)" //injection malveillante
input:="My name is: <!--#4DTEXT myName-->"
PROCESS 4D TAGS(input;output)
//la sortie est "My name is: $4DEVAL(QUIT 4D)"
```

A noter que la syntaxe \$4dtag prend en charge la correspondance de paires de guillemets ou de parenthèses. Par exemple, supposons que vous ayez besoin d'évaluer la chaîne complexe (fictive) suivante :

```
String(1) + "\"(hello)\""
```

Vous pouvez écrire :

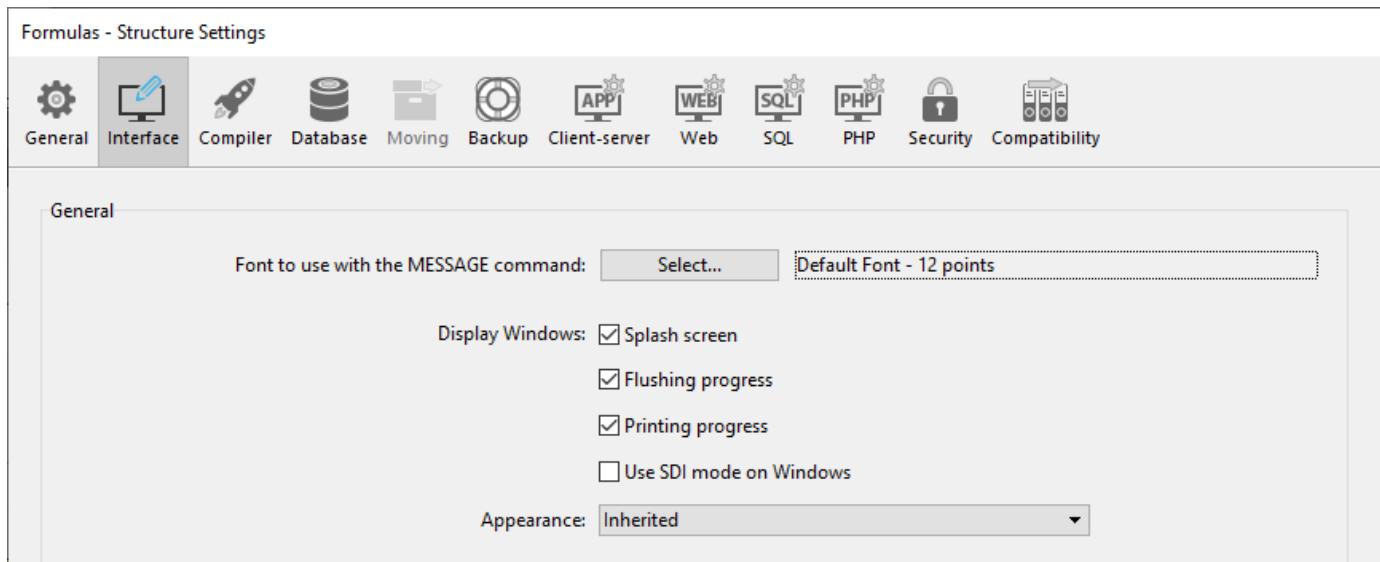
```
input:="$4DEVAL( String(1)+\"\\\""(hello)\\\"\\")"
PROCESS 4D TAGS(input;output)
-->output is 1"(hello)"
```

Page Interface

La page Interface vous permet de régler diverses options liées à l'interface du projet.

Général

Cette zone vous permet de personnaliser diverses options d'affichage.



Police à utiliser avec la commande MESSAGE

Le bouton Sélectionner... vous permet de définir la police et la taille des caractères utilisés par la commande `MESSAGE`.

La police et la taille de police par défaut dépendent de la plate-forme d'exécution de 4D.

Les parties suivantes de 4D sont également affectées par cette propriété :

- certaines zones d'aperçu de l'Explorateur,
 - la règle de L'Editeur de formulaires
- D'autres options permettent de configurer l'affichage de diverses fenêtres du mode Application.

- Accueil : lorsque cette option est désélectionnée, la [fenêtre d'accueil \(ou "splash screen"\) de la barre de menus courante](#) en mode Application n'apparaît pas. Lorsque vous masquez cette fenêtre, c'est à vous de gérer l'affichage de toutes vos fenêtres par programmation, par exemple dans la méthode base `On Startup`.
- Ecriture du cache : lorsque cette option est cochée, 4D affiche une fenêtre en bas à gauche de l'écran pendant que les données du cache sont vidées. Etant donné que cette opération bloque momentanément les actions de l'utilisateur, l'affichage de cette fenêtre lui permet de savoir que l'écriture du cache est en cours.

Vous pouvez définir la [fréquence d'écriture du cache](#) dans Propriétés > Base de données > Mémoire.

- Progression de l'impression : permet, lors de l'impression, d'activer ou de désactiver l'affichage de la boîte de dialogue de progression de l'impression.
- Utiliser le mode SDI sous Windows : lorsque cette option est cochée, 4D active automatiquement le mode SDI (Single-Document Interface) dans votre application fusionnée si elle est exécutée dans un contexte pris en charge.

Cette option peut être sélectionnée sur macOS mais sera ignorée lorsque l'application sera exécutée sur cette plateforme.

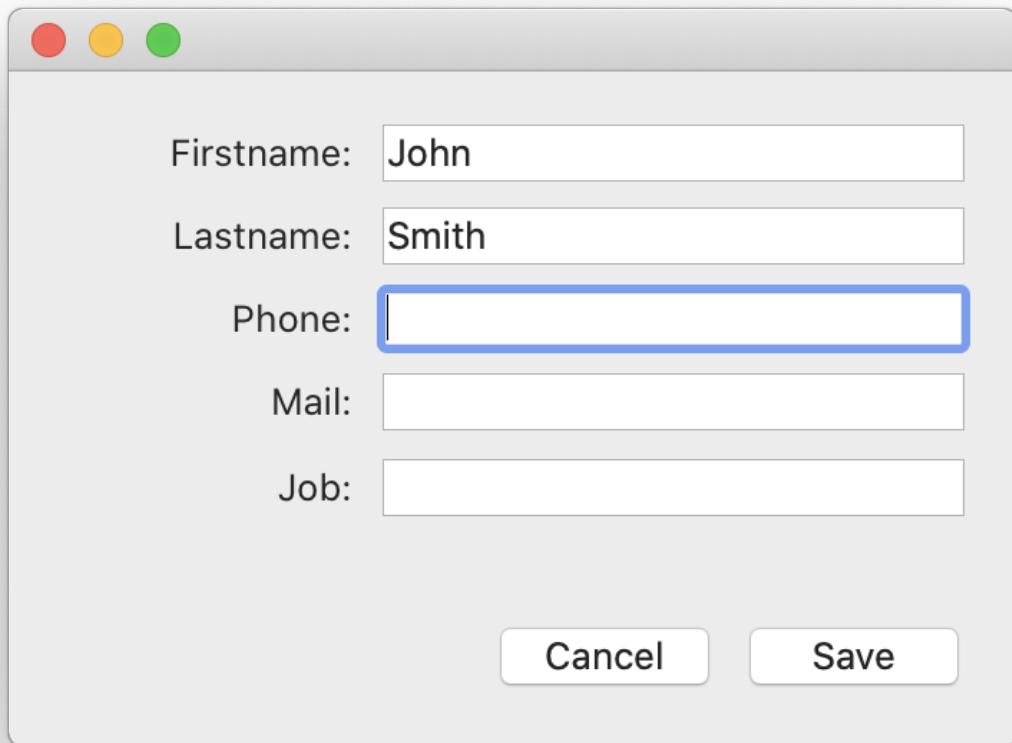
Apparence

Ce menu vous permet de sélectionner la palette de couleurs à utiliser au niveau de l'application principale. Une palette de couleurs définit un ensemble global de couleurs d'interface pour les textes, les arrière-plans, les fenêtres, etc., utilisés dans vos formulaires.

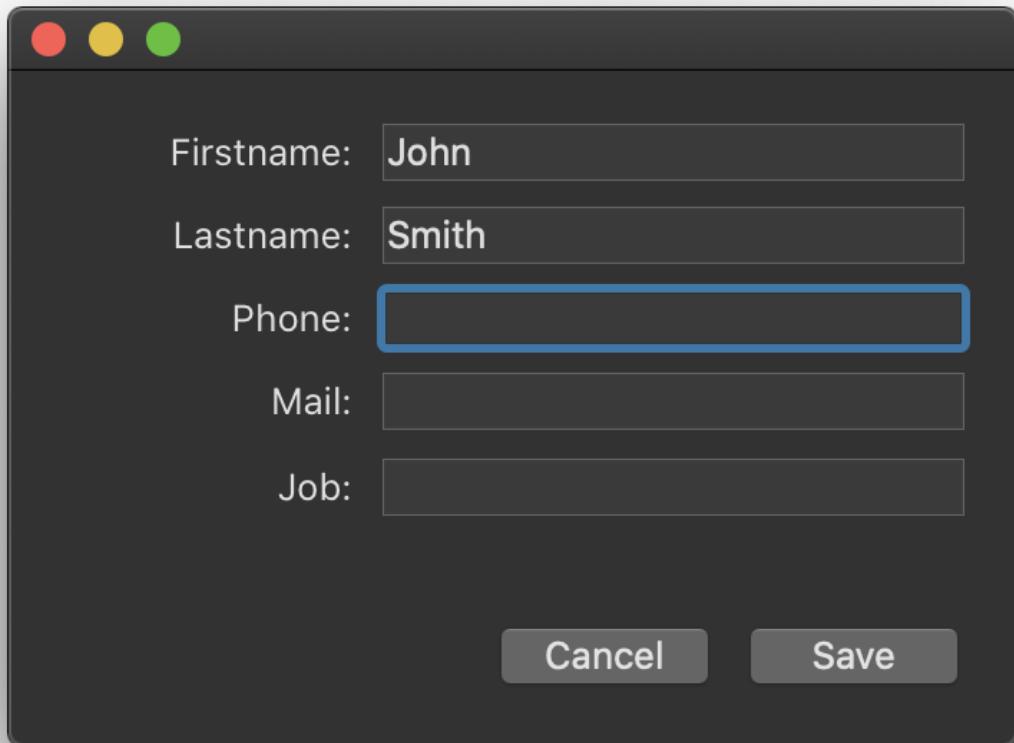
Cette option fonctionne uniquement sur macOS. Sous Windows, la palette "Light" est toujours utilisée.

Les palettes suivants sont disponibles :

- Clair : l'application utilisera le thème clair par défaut



- Foncé: l'application utilisera le thème foncé par défaut



- Héritée (par défaut) : l'application hérite du niveau de priorité le plus élevé (c'est-à-dire les préférences de l'utilisateur du système d'exploitation)

Les thèmes par défaut peuvent être gérés par du CSS. Pour plus d'informations, veuillez consulter la section [Media Queries](#).

La palette d'application principale sera appliquée aux formulaires par défaut. Toutefois, elle peut être remplacée :

- par la commande [SET APPLICATION COLOR SCHEME](#) au niveau de la session de travail ;
- en utilisant la propriété de formulaire [Color Scheme](#) à chaque niveau de formulaire (niveau de priorité le plus élevé). Note : à l'impression, les formulaires utilisent toujours la palette "Light".

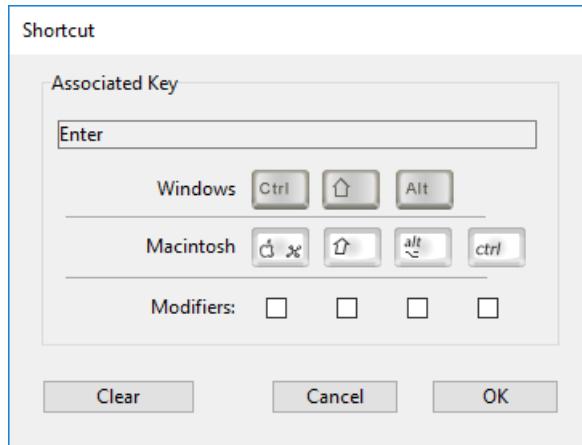
Raccourcis

La zone Raccourcis clavier permet d'afficher et de modifier les raccourcis clavier par défaut pour trois opérations de base du formulaire 4D dans vos applications desktop. Ces raccourcis clavier sont identiques pour les deux plateformes. Les icônes des touches indiquent les touches correspondant à Windows et à macOS.

Les raccourcis clavier par défaut sont les suivants :

- Acceptation de saisie : Entrée
- Annulation de saisie : Echap
- Ajout d'un sous-formulaire : Ctrl+Shift+ / (Windows) ou Command+Shift+ / (macOS)

Pour modifier le raccourci d'une opération, cliquez sur le bouton Modifier correspondant. La boîte de dialogue suivante apparaît :



Pour modifier le raccourci clavier, tapez la nouvelle combinaison de touches sur votre clavier et cliquez sur OK. Si vous préférez ne pas avoir de raccourci pour une opération, cliquez sur Effacer.

Page Compilateur

Ces paramètres sont détaillés dans la section [Paramètres du compilateur](#).

Page Base de données

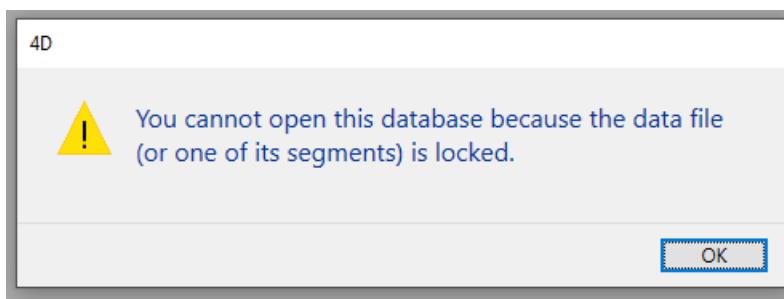
Page Stockage de données

Cette page vous permet de paramétrer le stockage des données de la base 4D sur le disque.

Paramètres généraux

Autoriser l'ouverture du fichier de données en lecture seule

Cette option permet de paramétrer le fonctionnement de l'application en cas d'ouverture d'un fichier de données verrouillé au niveau du système d'exploitation. 4D intègre un dispositif permettant d'empêcher automatiquement l'ouverture d'une base lorsque son fichier de données ou l'un de ses segments est verrouillé. Dans ce cas, lorsque la détection est activée, 4D affiche un message d'alerte et la base quitte :



Lorsque l'option n'est pas cochée, il n'est pas possible d'ouvrir la base avec un fichier de données verrouillé (fonctionnement par défaut pour les bases 4D).

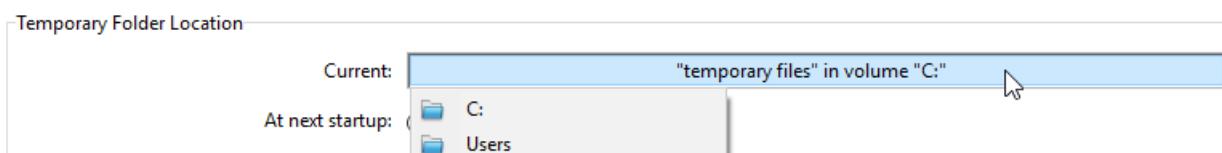
A propos du verrouillage des fichiers

Les fichiers verrouillés peuvent être lus mais leur contenu ne peut pas être modifié. Par exemple, les fichiers sont verrouillés lorsqu'ils sont stockés sur un support non réinscriptible (type DVD) ou lorsqu'ils sont recopiés depuis un tel support. 4D peut travailler de manière transparente avec des fichiers de données verrouillés, ce qui permet notamment d'exploiter des bases en lecture sur DVD. Toutefois, ce fonctionnement induit le risque d'utilisation involontaire d'un fichier de données verrouillé dans lequel les modifications ne seront pas sauvegardées. C'est la raison pour laquelle par défaut 4D n'autorise pas l'ouverture des bases avec un fichier de données verrouillé.

Emplacement du dossier temporaire

Cette zone vous permet de modifier l'emplacement des fichiers temporaires créés lors de l'exécution de 4D. Le dossier des fichiers temporaires est utilisé par l'application, en cas de nécessité, pour écrire provisoirement sur disque des données se trouvant en mémoire.

L'emplacement courant de ce dossier est affiché dans la zone "Actuel :". Vous pouvez cliquer dans cette zone pour faire apparaître le chemin d'accès sous forme de pop up menu :



Trois options d'emplacement sont proposées :

- ****Système **:** Lorsque cette option est sélectionnée, les fichiers temporaires 4D sont créés dans un dossier situé à l'emplacement spécifié par Windows ou macOS. Vous pouvez lire l'emplacement courant défini par votre système à l'aide de la commande `Temporary folder`. Les fichiers sont placés dans un sous-dossier dont le nom est construit à partir du nom de la base et d'un identifiant unique.

- Dossier du fichier de données (option par défaut) : Lorsque cette option est sélectionnée, les fichiers temporaires 4D sont créés dans un dossier nommé "temporary files" situé au même niveau que le fichier de données de la base.
- Défini par l'utilisateur : Cette option permet de définir un emplacement personnalisé. En cas de modification de cette option, sa prise en compte nécessitera le redémarrage de la base. 4D vérifie que le dossier sélectionné est accessible en écriture. Si ce n'est pas le cas, l'application essaiera les autres options jusqu'à ce qu'un dossier valide soit trouvé.

Cette option est stockée dans les propriétés additionnelles ("extra properties") de la structure, accessibles lors de l'exportation xml de la définition de structure (voir [Exporter et importer des définitions de structure](#)).

Comparaison de texte

Si vous modifiez ces options, vous devez quitter et rouvrir la base afin que la modification soit prise en compte. A la réouverture, une réindexation automatique est effectuée sur tous les index de la base.

- Considérer @ comme joker uniquement au début et à la fin des chaînes de caractères : cette option vous permet de définir la manière dont 4D doit évaluer le caractère @ (arobase) lors des recherches ou des comparaisons de chaînes de caractères, lorsqu'il se trouve au sein d'un mot. Lorsque l'option n'est pas cochée (valeur par défaut), le caractère @ est considéré comme un "joker", c'est-à-dire un remplaçant de tout caractère (voir [Joker de recherche \(@\)](#)).

Lorsque l'option est cochée, le caractère @ est considéré comme un simple caractère s'il se trouve au milieu d'un mot. Cette possibilité est particulièrement utile pour les bases de données stockant des adresses e-mail (qui se présentent sous la forme "nom@fournisseur.xx"). Cette option influe sur les recherches, les tris et les comparaisons de chaînes de caractères, pour les données stockées dans les tables ou en mémoire (tableaux). Sont concernés les champs alpha (indexés ou non) et texte ainsi que les variables alpha et texte.

Notes :

- En ce qui concerne les recherches, il est important de noter que si le critère de recherche commence ou se termine par @, le caractère @ est toujours considéré comme un joker. Seul le fait que ce caractère soit placé à l'intérieur d'un mot (exemple : bill@cgi.com) entraîne un traitement différent de la part de 4D.
- Cette option influe également sur le comportement des commandes du langage du thème [Objets \(Formulaires\)](#), qui acceptent le caractère @ dans leur paramètre objet.
- Pour des raisons de sécurité, seuls l'Administrateur et le Super_Utilisateur de la base peuvent modifier ce paramètre.
- Langue du fichier de données courant : cette option permet de paramétriser la langue utilisée pour le traitement et la comparaison des chaînes de caractères pour la base de données ouverte. Le choix d'une langue de comparaison influe sur le tri et la recherche des textes ainsi que le passage en minuscules/majuscules mais n'a pas d'incidence sur la traduction des libellés ou sur les formats de dates, d'heure ou monétaires qui restent, eux, dans la langue du système. Par défaut, 4D utilise la langue du système.

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

Il est possible de modifier ce paramètre dans les Préférences de l'application (voir [Page General](#)). Dans ce cas, il s'applique à toutes les nouvelles bases créées par 4D.

- **N'utiliser que les caractères non alphanumériques pour les mots-clés ** : cette option modifie l'algorithme utilisé par 4D pour identifier les séparateurs de mots-clés et donc construire les index de mots-clés. Par défaut, lorsque cette option n'est pas cochée, 4D utilise un algorithme sophistiqué tenant compte des spécificités linguistiques.

Cet algorithme est semblable à celui utilisé de façon standard par les logiciels de traitement de texte pour déterminer les limites d'une sélection en cas de double-clic dans un mot. Pour plus d'informations sur cet algorithme, reportez-vous à l'adresse <http://userguide.icu-project.org/boundaryanalysis>.

Lorsque cette option est cochée, 4D utilise un algorithme simplifié. Dans cette configuration, tout caractère non

alphanumérique (c'est-à-dire qui n'est ni une lettre ni un chiffre) est considéré comme séparateur de mot-clé. Ce paramétrage répond à des besoins spécifiques associés à certaines langues telles que le japonais.

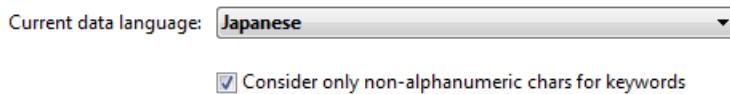
- Ordre de tri approprié pour la recherche : Cette option s'affiche uniquement lorsque la langue japonaise est sélectionnée. Elle modifie l'interprétation de caractères tels que "Marques Katakana-Hiragana de son prolongé" ou "長音記号" ou les "Marques d'itération japonaises" telles que ">" ou ">". Un locuteur japonais préférera les résultats issus d'une recherche effectuée lorsque le paramètre est activé.

Prise en charge de Mecab (version japonaise)

En système japonais, 4D prend en charge la librairie *MeCab*, comportant un algorithme d'indexation des mots-clés particulièrement adapté à la langue japonaise.

Cet algorithme est utilisé par défaut dans les versions japonaises de 4D. Si vous le souhaitez, vous pouvez désactiver l'utilisation de l'algorithme *MeCab* et utiliser la librairie classique *ICU*.

Pour désactiver *MeCab*, il vous suffit de cocher l'option N'utiliser que les caractères non alphanumériques pour les mots-clés :



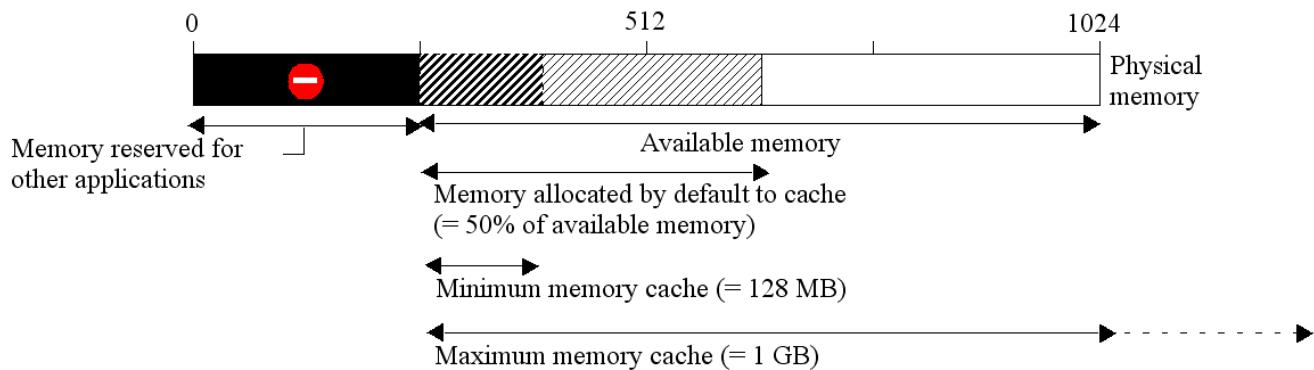
Page Mémoire

Utilisez les paramètres de cet onglet pour configurer la mémoire cache de la base.

Paramètres du cache pour la base

- Calcul du cache adaptatif : Lorsque cette option est cochée, la gestion de la mémoire cache est assurée dynamiquement par le système — dans des limites que vous définissez. Ce principe permet de configurer une mémoire cache performante, adaptée à la plupart des configurations. La taille du cache est alors calculée dynamiquement en fonction des paramétrages définis. Les valeurs proposées par défaut correspondent à une utilisation standard de 4D.
 - Mémoire à réservé pour le système et les autres applications : partie de la mémoire RAM à réservé au Système et aux autres applications. Cette valeur peut être augmentée à des fins d'optimisation lorsque d'autres applications consommatrices de mémoire tournent sur le même poste que 4D.
 - Mémoire disponible utilisée pour le cache : pourcentage de la mémoire restante alloué par défaut au cache. Pour obtenir la taille allouée par défaut au cache, il suffit donc d'effectuer le calcul suivant : (Mémoire physique - Mémoire physique à réservé) x Pourcentage de la mémoire utilisé pour le cache. Dans le mode adaptatif, la taille de mémoire cache varie dynamiquement en fonction des besoins de l'application et du système. Vous pouvez fixer les bornes de ces variations à l'aide des deux options suivantes :
 - Taille minimale : Quantité minimale de mémoire devant être réservée pour le cache. Cette valeur ne peut être inférieure à 100 Mo.
 - Taille maximale : Quantité maximale de mémoire pouvant être utilisée par le cache. Cette valeur est virtuellement illimitée. La définition de bornes est utile dans le cadre des bases diffusées sur des machines dont vous ne connaissez pas a priori la configuration mémoire. Dans ce cas, les bornes vous permettent de garantir des performances minimales dans tous les cas. Le schéma suivant illustre ce fonctionnement :

Exemple de calcul de la mémoire cache : Mémoire physique à réservé = 256 Mo Pourcentage de la mémoire disponible utilisé pour le cache = 50 % Taille maximale = 1 Go Taille minimale = 128 Mo



- Calcul du cache adaptatif non cochée : dans ce mode, vous définissez vous-même la taille de la mémoire cache pour la base. 4D affiche alors une zone de saisie permettant de définir la mémoire cache à utiliser ainsi que les informations relatives à la mémoire physique (mémoire RAM disponible sur la machine), le cache actuel et le cache après redémarrage (tenant compte de vos modifications).

La taille de mémoire cache que vous saisissez sera réservée pour la base 4D, quel que soit l'état des ressources de la machine. Ce paramétrage peut être utilisé dans certaines configurations spécifiques, ou lorsque la base est destinée à fonctionner sur des systèmes disparates en termes de mémoire. Dans la plupart des cas, le cache adaptatif est plus performant.

- Ecriture cache toutes les... Minutes/secondes : spécifie la fréquence de sauvegarde automatique du cache de données, c'est-à-dire son écriture sur le disque. 4D écrit les données placées dans le cache à intervalles fixes. Vous pouvez définir tout intervalle compris entre 1 seconde et 500 minutes. Par défaut, 4D stocke vos données toutes les 20 secondes. L'application écrit aussi vos données sur disque lorsque vous changez de mode ou quittez l'application. Vous pouvez aussi appeler la commande **FLUSH CACHE** pour déclencher l'écriture du cache à tout moment.

Quand vous prévoyez de saisir beaucoup de données, il est souhaitable de fixer un intervalle court. En effet, en cas de coupure de courant, vous ne perdriez que les données saisies depuis la dernière écriture (si la base fonctionne sans fichier d'historique).

Si chaque opération d'écriture du cache est accompagnée d'un fort ralentissement de la base de données, il faut ajuster la fréquence. Ce symptôme signifie une sauvegarde massive d'enregistrements. Dans ce cas, une fréquence d'écriture plus élevée, donc plus rapide, est plus efficace.

Par défaut, 4D affiche une petite fenêtre lors de l'écriture du cache. Si vous ne voulez pas ce rappel visuel, vous pouvez désélectionner l'option Ecriture du cache dans la [Page Interface](#).

Page Client/Serveur

Les pages Client-serveur regroupent les propriétés liées à l'utilisation de la base de données en mode client-serveur. Bien entendu, ces propriétés ne sont prises en compte que lorsque la base de données est utilisée en mode distant.

Page Options réseau

Réseau

Publier la base au démarrage

Cette option vous permet d'indiquer si la base de données 4D Server apparaîtra ou non dans la liste des bases de données publiées.

- Lorsque cette option est cochée (par défaut), la base de données est rendue publique et apparaît dans la liste des bases de données publiées (onglet Disponible).
- Lorsque l'option n'est pas cochée, la base de données n'est pas rendue publique et n'apparaît pas dans la liste des bases de données publiées. Pour se connecter, les utilisateurs doivent saisir manuellement l'adresse de la base de données dans l'onglet Personnalisé de la boîte de dialogue de connexion.

Si vous modifiez ce paramètre, vous devez redémarrer la base du serveur pour qu'il soit pris en compte.

Nom de publication

Cette option vous permet de modifier le nom de publication d'une base de données 4D Server, c'est-à-dire le nom affiché dans l'onglet dynamique Disponible de la boîte de dialogue de connexion (voir la section [Connexion à une base de données 4D Server](#)). Par défaut, 4D Server utilise le nom du fichier de projet. Vous pouvez saisir le nom personnalisé de votre choix.

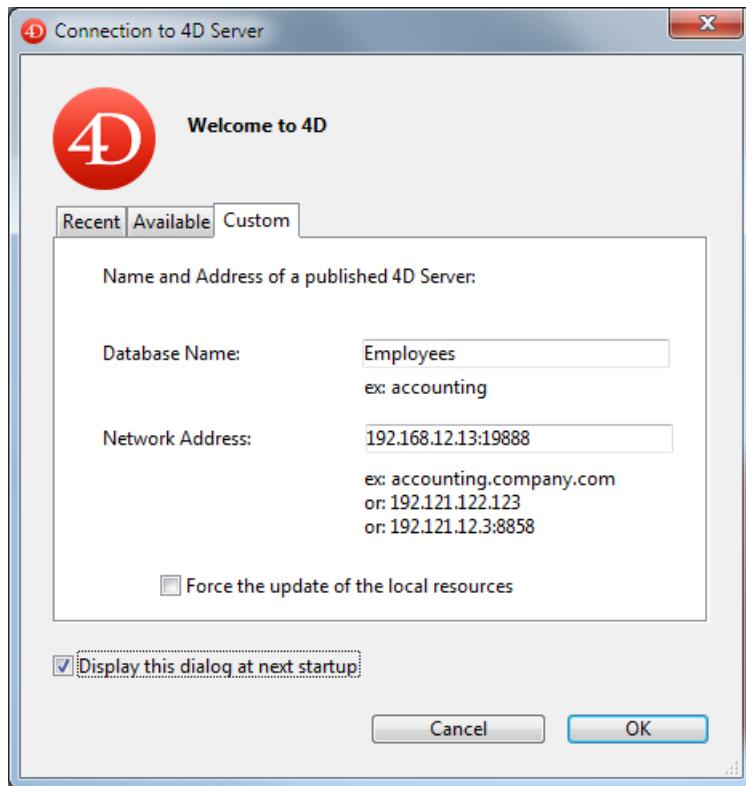
Ce paramètre n'est pas pris en compte dans les applications client-serveur personnalisées. En théorie, l'application client se connecte directement à l'application serveur, sans passer par la boîte de dialogue de connexion. Toutefois, en cas d'erreur, cette boîte de dialogue peut apparaître ; dans ce cas, le nom de publication de l'application serveur correspond au nom du projet compilé.

Numéro de port

Cette option vous permet de modifier le numéro du port TCP sur lequel 4D Server publie la base de données. Cette information est stockée dans le projet et sur chaque machine cliente. Par défaut, le numéro de port TCP utilisé par 4D Server et 4D en mode distant est 19813.

La personnalisation de cette valeur est nécessaire lorsque vous souhaitez utiliser plusieurs applications 4D sur la même machine ; dans ce cas, vous devez spécifier un numéro de port différent pour chaque application. Lorsque vous modifiez cette valeur depuis 4D Server ou 4D, elle est automatiquement passée à toutes les machines 4D connectées à la base de données.

Pour mettre à jour les autres machines clientes qui ne sont pas connectées, il suffit de saisir le nouveau numéro de port (précédé de deux points) après l'adresse IP de la machine serveur dans l'onglet Personnalisé de la boîte de dialogue de connexion lors de la prochaine connexion. Par exemple, si le nouveau numéro de port est 19888 :



Seules les bases de données publiées sur le même port que celui défini dans 4D client sont visibles sur la page de publication dynamique TCP/IP.

4D Server et numéros de port

4D Server utilise trois ports TCP pour les communications entre les serveurs internes et les clients :

- SQL Server : 19812 par défaut (peut être modifié via la page "SQL/Configuration" des Préférences).
- Serveur d'application : 19813 par défaut (peut être modifié via la page "Client-Serveur/Configuration" des Préférences, voir ci-dessus).
- Serveur DB4D (serveur de base de données) : 19814 par défaut. Ce numéro de port ne peut pas être modifié directement mais il est toujours constitué du numéro de port du serveur d'application + 1. Lorsqu'un client 4D se connecte à 4D Server, il utilise le port TCP du serveur d'application (19813 ou le port indiqué après les deux points ":" dans l'adresse IP affichée dans la boîte de dialogue de connexion). La connexion aux autres serveurs via leurs ports respectifs est alors automatique ; il n'est plus nécessaire de les spécifier. A noter que dans le cas d'un accès via un routeur ou un pare-feu, les trois ports TCP doivent être ouverts explicitement.

Authentification de l'utilisateur auprès du serveur de domaine

Cette option vous permet de mettre en œuvre des fonctionnalités SSO (*Single Sign On*) dans votre base de données 4D Server sous Windows. Lorsque vous cochez cette option, 4D se connecte de manière transparente à l'Active directory du serveur de domaine Windows et obtient les tokens d'authentification disponibles. Cette option est décrite dans la section [Single Sign On \(SSO\) sur Windows](#).

Service Principal Name

Lorsque l'authentification unique (SSO) est activée (voir ci-dessus), vous devez remplir ce champ si vous souhaitez utiliser Kerberos comme protocole d'authentification. Cette option est décrite dans la section [Single Sign On \(SSO\) sur Windows](#).

Délai avant déconnexion Client-Serveur

Ce dispositif permet de définir le timeout (période d'inactivité au-delà de laquelle la connexion est fermée) entre 4D Server et les machines clientes qui s'y connectent. L'option "Illimité" supprime le timeout. Lorsque cette option est sélectionnée, le contrôle de l'activité du client est éliminé.

Lorsqu'un délai d'attente est sélectionné, le serveur ferme la connexion d'un client s'il ne reçoit aucune demande de ce

dernier pendant le délai spécifié.

Communication client-serveur

Inscrire les clients au démarrage pour Exécuter sur client

Lorsque cette option est cochée, toutes les machines distantes 4D se connectant à la base de données peuvent exécuter des méthodes à distance. Ce mécanisme est détaillé dans la section [Procédures stockées sur les machines clientes](#).

Crypter les communications Client-Serveur

Cette option permet d'activer le mode sécurisé pour les communications entre la machine serveur et les machines distantes 4D. Cette option est détaillée dans la section [Crypter les connexions Client-Serveur](#).

Mise à jour du dossier Resources en cours de session

Ce paramètre permet de définir globalement le mode de mise à jour de l'instance locale du dossier Resources sur les machines 4D connectées lorsque le dossier Resources de la base de données est modifié en cours de session (le dossier Ressources est automatiquement synchronisé sur la machine distante à chaque ouverture de session). Trois paramètres sont disponibles :

- Jamais : Le dossier Resources local n'est pas mis à jour pendant la session. La notification envoyée par le serveur est ignorée. Le dossier Resources local peut être mis à jour manuellement à l'aide de la commande Update Local Resources du menu d'action (voir [Utilisation de l'explorateur de ressources](#)).
- Toujours : La synchronisation du dossier local Resources est automatiquement effectuée pendant la session chaque fois qu'une notification est envoyée par le serveur.
- Demander : Lorsque la notification est envoyée par le serveur, une boîte de dialogue est affichée sur les postes clients, indiquant la modification. L'utilisateur peut alors accepter ou refuser la synchronisation du dossier Resources local. Le dossier Resources centralise les fichiers personnalisés nécessaires à l'interface de la base (fichiers de traduction, images, etc.). Des mécanismes automatiques ou manuels permettent de notifier chaque client lorsque le contenu de ce dossier a été modifié. Pour plus d'informations, reportez-vous à la section [Gestion du dossier Resources](#).

Page Configuration IP

Table de configuration Autoriser-Refuser

Cette table vous permet de définir des règles de contrôle d'accès à la base en fonction de l'adresse IP des postes 4D distants. Cette option permet de renforcer la sécurité par exemple pour des applications stratégiques.

Cette table de configuration ne contrôle pas les connexions Web.

Le fonctionnement de la table de configuration est le suivant :

- La colonne "Autoriser-Refuser" permet de sélectionner le type de règle à appliquer (Autoriser ou Refuser) à l'aide d'un pop up menu. Pour ajouter une règle d'adresses, cliquez sur le bouton Ajouter. Une nouvelle ligne apparaît dans la table. Le bouton Supprimer permet de supprimer la ligne courante.
- La colonne "Adresse IP" permet de désigner la ou les adresse(s) IP concernée(s) par la règle. Pour spécifier une adresse, cliquez dans la colonne et saisissez l'adresse sous la forme 123.45.67.89 (format IPv4) ou 2001:0DB8:0000:85A3:0000:0000:AC1F:8001 (format IPv6). Vous pouvez utiliser le caractère * (étoile) pour spécifier des adresses du type "commence par". Par exemple, 192.168.* indique toutes les adresses débutant par 192.168.
- L'application des règles s'effectue dans l'ordre d'affichage de la table. Si deux règles sont contradictoires, la priorité sera accordée à la règle située le plus haut dans le tableau. Vous pouvez réordonner les lignes en modifiant le tri courant (cliquez sur un en-tête de colonne pour alterner le sens de tri). Vous pouvez également déplacer des lignes par glisser-déposer.
- Pour des raisons de sécurité, seules les adresses correspondant à une règle d'autorisation explicite pourront se connecter. En particulier, si la table contient uniquement une ou plusieurs règle(s) de type Refuser, toutes les adresses seront refusées car aucune ne satisfera à au moins une règle. Si vous souhaitez refuser certaines adresses

et autoriser toutes les autres, ajoutez une règle Autoriser * à la fin de la table. Par exemple :

- Refuser 192.168.* (refuser toutes adresses débutant par 192.168)
- Autoriser * (et autoriser les autres)

Par défaut, aucune restriction de connexion n'est appliquée par 4D Server : la première ligne de la table contient le libellé Autoriser et le caractère * (toute adresse).

Page Web

Les onglets de la page Web permettent de paramétrer les multiples aspects du serveur Web intégré de 4D (sécurité, démarrage, connexions, services Web, etc.). Pour plus d'informations sur le fonctionnement du serveur Web de 4D, reportez-vous au chapitre [Serveur Web](#). Pour plus d'informations sur les services Web de 4D, reportez-vous au chapitre [Publication et utilisation de Services Web](#).

Configuration

Information de publication

Lancer le serveur Web au démarrage

Indique si le serveur Web doit être démarré dès le lancement de l'application 4D. Cette option est détaillée dans la section [Administration du serveur Web](#).

Activer HTTP

Indique si le Web server accepte des connexions non sécurisées. Voir [Activer HTTP](#).

Port HTTP

Numéro de port IP (TCP) d'écoute pour HTTP. Voir [Port HTTP](#).

Adresse IP

Adresse IP sur laquelle le serveur web 4D recevra les requêtes HTTP (4D local et 4D Server). Voir [Adresse IP à écouter](#).

Activer HTTPS

Indique si le Web server accepte des connexions sécurisées. Voir [Activer HTTPS](#).

Port HTTPS

Permet de modifier le numéro du port TCP/IP utilisé par le serveur Web pour les connexions HTTP sécurisées sur TLS (protocole HTTPS). Voir [Port HTTPS](#).

Autoriser l'accès aux bases de données par le biais des URL 4DSYNC

Note de compatibilité : Cette option est **obsolète**. Pour l'accès aux bases de données via HTTP, il est désormais recommandé d'utiliser les fonctionnalités de stockage de données à distance ORDA et les requêtes REST.

Chemins

Racine HTML par défaut

Permet de définir l'emplacement par défaut des fichiers du site Web et indique le niveau hiérarchique sur le disque au-dessus duquel aucune requête ne pourra accéder. Voir [Dossier Racine](#).

Page d'accueil par défaut

Cette option permet de désigner la page d'accueil par défaut pour le serveur Web. Voir [page d'accueil par défaut](#).

Options (I)

Cache

Utiliser le cache Web de 4D

Active le cache de la page web. Voir [Cache](#).

Taille du cache des pages

Définit la taille du cache. Voir [Cache](#).

Vider le cache

À tout moment, vous pouvez vider le cache des pages et des images qu'il contient (si, par exemple, vous avez modifié une page statique et que vous souhaitez la recharger dans le cache). À tout moment, vous pouvez vider le cache des pages et des images qu'il contient (si, par exemple, vous avez modifié une page statique et que vous souhaitez la recharger dans le cache). Le cache est alors immédiatement effacé.

Vous pouvez également utiliser l'URL spécifique </4DCACHECLEAR>.

Process Web

Cette zone vous permet de configurer, par le serveur Web, la gestion des sessions utilisateur et leurs process associés.

L'option Sessions héritées n'est disponible que pour la compatibilité dans les bases/projets créé(e)s avec les versions de 4D antérieures à 4D v18 R6.

Sessions extensibles (sessions multi-process)

Lorsque vous sélectionnez cette option (recommandée), une session utilisateur est gérée via un objet Session. Voir la page [Sessions utilisateur](#).

Pas de sessions

Lorsque cette option est sélectionnée, le serveur Web ne fournit aucune prise en charge spécifique pour les [sessions utilisateur](#). Les requêtes successives des clients Web sont toujours indépendantes et aucun contexte n'est conservé sur le serveur.

Dans ce mode, vous pouvez configurer des paramètres de serveur Web supplémentaires :

- [Process Web simultanés maxi](#)
- [Réutilisation des contextes temporaires \(4D en mode distant\)](#)
- [Utiliser des process préemptifs](#)

Anciennes sessions (sessions process uniques)

Note de compatibilité : Cette option est disponible uniquement dans les bases/projets créé(e)s avec une version 4D antérieure à 4D v18 R6.

Cette option permet de gérer les anciennes sessions utilisateur par le serveur 4D HTTP. Ce mécanisme est décrit dans la section [Gestion des sessions Web](#). Voir [Garder session](#).

L'option [Réutilisation des contextes temporaires \(4D en mode distant\)](#) est automatiquement cochée (et verrouillée).

Process Web simultanés maxi

Non disponible avec les [sessions extensibles](#).

Cette option indique la limite strictement supérieure du nombre de process Web pouvant être simultanément ouverts sur le serveur. Voir [Process Web simultanés maxi](#).

Réutilisation des contextes temporaires

Non disponible avec les [sessions extensibles](#).

Permet d'optimiser le fonctionnement du serveur Web de 4D en mode distant. Voir [Réutilisation des contextes temporaires en mode distant](#).

Utiliser des process préemptifs

Non disponible avec les [sessions extensibles](#).

Active les process web préemptifs dans vos applications compilées. Lorsque l'option Utiliser des processus préemptifs est sélectionnée, l'éligibilité de votre code lié au Web (y compris les balises 4D et les méthodes base Web) à l'exécution préemptive sera évaluée pendant la compilation. Pour plus d'informations, voir [Utiliser des processus Web préemptifs](#).

Cette option ne s'applique pas aux sessions extensibles, aux process REST (mode compilé), ni aux process Web service (serveur et client). Voir [Activer le mode préemptif pour le serveur web](#).

Conservation des process inactifs

Non disponible avec les [sessions extensibles](#).

Permet de définir le délai maximum avant fermeture (timeout) des process Web inactifs sur le serveur. Voir [Durée de vie des process inactifs](#).

Mots de passe Web

Définit le système d'authentification que vous souhaitez utiliser pour le serveur Web. Trois options sont proposées :

Personnalisé (défaut) Mots de passe protocole BASIC Mots de passe protocole DIGEST

Il est recommandé d'utiliser l'authentification personnalisée. Voir le chapitre [Authentification](#) dans la section *Développement Web*.

Options (II)

Conversion texte

Envoyer directement les caractères étendus

Voir [Propriétés obsolètes](#).

Standard Set

Définit le jeu de caractères à utiliser par le serveur web 4D. Voir [Jeu de caractères](#).

Utiliser les connexions persistantes

Voir [Propriétés obsolètes](#).

Paramètres CORS

Activer CORS

Active le service Cross-origin resource sharing (CORS). Voir [Activer Cors](#).

Noms de domaines/Méthodes HTTP autorisées

Liste des hôtes et méthodes autorisées pour le service CORS. Voir [Paramètres CORS](#).

Journal (format)

Format du journal (logweb.txt)

Démarre ou arrête l'enregistrement des requêtes reçues par le serveur Web 4D dans le fichier *logweb.txt* et définit son format. Voir [Enregistrement des logs](#).

L'activation et la désactivation du fichier d'historique des requêtes peut également être effectuée par programmation, à l'aide de la commande [WEB SET OPTION](#).

Le menu de format du journal propose les options suivantes :

- Pas de journal : Lorsque cette option est sélectionnée, 4D ne génère pas d'historique des requêtes.
- CLF (Common Log Format) : Lorsque cette option est sélectionnée, l'historique des requêtes est généré au format CLF. Avec le format CLF, chaque ligne du fichier représente une requête sous la forme :
host rfc931 utilisateur [JJ/MMM/AAAA] "requête" statut longueur Chaque champ est séparé par un espace, chaque ligne se termine par la séquence CR/LF (caractère 13, caractère 10).
 - hôte : adresse IP du client (ex. 192.100.100.10)
 - rfc931 : information non gérée par 4D, c'est toujours - (signe moins)
 - utilisateur : nom de l'utilisateur tel qu'il s'est authentifié, sinon - (signe moins). Si le nom de l'utilisateur contient des espaces, ils sont remplacés par des _ (tiret bas).
 - JJ : jour, MMM : mois abrégé en 3 lettres et toujours en anglais (Jan, Feb, ...), AAAA : année, HH : heure, MM : minutes, SS : secondes

La date et heure sont locales au serveur.

- requête : requête envoyée par le client (ex. GET /index.htm HTTP/1.0)
- statut : réponse donnée par le serveur.
- longueur : taille des données renvoyées (hors en-tête HTTP) ou 0.

Note : Pour des raisons de performances, les opérations sont stockées dans une mémoire tampon par paquets de 1 Ko avant d'être écrites sur disque. Les opérations sont également écrites sur disque si aucune requête n'a été envoyée au bout de 5 secondes. Les valeurs possibles de statut sont : 200: OK 204: Pas de contenu 302: Redirection 304: Non modifiée 400: Mauvaise requête 401: Authentification requise 404: Non trouvé 500: Erreur interne Le format CLF ne peut pas être personnalisé.

- DLF (Combined Log Format) : Lorsque cette option est sélectionnée, l'historique des requêtes est généré au format DLF. Le format DLF est semblable au format CLF dont il reprend exactement la structure. Il ajoute simplement deux champs HTTP supplémentaires à la fin de chaque requête : Referer et User-agent.
 - Referer : contient l'URL de la page pointant vers le document demandé.
 - User-agent : contient le nom et la version du navigateur ou du logiciel client à l'origine de la requête.

Le format DLF ne peut pas être personnalisé.

- ELF (Extended Log Format) : Lorsque cette option est sélectionnée, l'historique des requêtes est généré au format ELF. Le format ELF est largement répandu dans le monde des serveurs HTTP. Il peut être utilisé pour construire des historiques sophistiqués qui répondent à des besoins spécifiques. Pour cette raison, le format ELF peut être personnalisé : il est possible de choisir les champs à enregistrer ainsi que leur ordre d'insertion dans le fichier.
- WLF (WebStar Log Format) : Lorsque cette option est sélectionnée, l'historique des requêtes est généré au format WLF. Le format WLF a été développé spécifiquement pour le serveur 4D WebSTAR. Il est semblable au format ELF, il dispose simplement de champs supplémentaires. Comme le format ELF, il est personnalisable.

Configurer les champs Lorsque vous choisissez le format ELF (Extended Log Format) ou WLF (WebStar Log Format), la zone "Formatage du journal" affiche les champs disponibles pour le format. Vous devrez sélectionner chaque champ à inclure dans le journal. Pour cela, utilisez les flèches de commande ou procédez par glisser-déposer.

Note : Il n'est pas possible de sélectionner deux fois le même champ.

Le tableau suivant répertorie les champs disponibles pour chaque format (par ordre alphabétique) et décrit leur contenu :

Champ	ELF	WLF	Valeur
BYTES_RECEIVED		X	Nombre d'octets reçus par le serveur
BYTES_SENT	X	X	Nombre d'octets envoyés par le serveur au client
C_DNS	X	X	Adresse IP du DNS (ELF : champ identique au champ C_IP)
C_IP	X	X	Adresse IP du client (par exemple 192.100.100.10)
CONNECTION_ID		X	Numéro unique de la connexion
CS(COOKIE)	X	X	Informations sur les cookies contenus dans la requête HTTP
CS(HOST)	X	X	Champ Host de la requête HTTP
CS(REFERER)	X	X	URL de la page pointant vers le document demandé
CS(USER_AGENT)	X	X	Informations sur le logiciel et le système d'exploitation du client
CS_SIP	X	X	Adresse IP du serveur
CS_URI	X	X	URI sur lequel la requête est effectuée
CS_URI_QUERY	X	X	Paramètres d'interrogation de la requête
CS_URI_STEM	X	X	Partie de la requête sans les paramètres d'interrogation
DATE	X	X	DD: jour, MMM: abréviation de 3 lettres pour le mois (Jan, Feb,...), YYYY: année
METHOD	X	X	Méthode HTTP utilisée pour la requête adressée au serveur
PATH_ARGS		X	Paramètres de la CGI : chaîne située après le caractère "\$"
STATUS	X	X	Réponse fournie par le serveur
TIME	X	X	HH: heure, MM: minutes, SS: secondes
TRANSFER_TIME	X	X	Délai ayant été nécessaire au serveur pour générer la réponse
USER	X	X	Nom d'utilisateur s'il s'est authentifié, sinon - (signe moins).
			Si le nom d'utilisateur contient des espaces, ils sont remplacés par des _ (traits de soulignement)
Variable URL		X	URL demandé par le client

Les dates et heures sont données au format GMT

Journal (périodicité)

Paramètres d'archivage automatique du journal des requêtes. Vous devez d'abord choisir la fréquence (jours, semaines, etc.) ou le critère de la taille limite du fichier en cliquant sur le bouton radio correspondant. Vous devez ensuite spécifier le moment précis du backup si nécessaire.

- Pas de sauvegarde du journal : La fonction de sauvegarde programmée est désactivée.
- Toutes les X heure(s) : Cette option est utilisée pour programmer des sauvegardes sur une base horaire. Vous pouvez entrer une valeur entre 1 et 24.
 - à partir de: Permet de définir l'heure du déclenchement du premier backup.
- Tous les N jour(s) à N : permet de programmer des backups sur une base journalière. Saisissez 1 si vous souhaitez une sauvegarde hebdomadaire. Lorsque vous cochez cette option, vous devez indiquer l'heure à laquelle la sauvegarde doit être déclenchée.

- Tous les N jour(s) à N : permet de programmer des backups sur une base hebdomadaire. Saisissez 1 si vous souhaitez une sauvegarde hebdomadaire. Lorsque vous cochez cette option, vous devez indiquer le ou les jours de la semaine et l'heure à laquelle chaque sauvegarde doit être déclenchée. Vous pouvez cocher un ou plusieurs jour(s) de la semaine. Par exemple, vous pouvez utiliser cette option pour définir deux sauvegardes hebdomadaires : une le mercredi et une le vendredi.
- Tous les N mois, Ne jour à N : permet de programmer des sauvegardes sur une base mensuelle. Saisissez 1 si vous souhaitez une sauvegarde mensuelle. Lorsque vous cochez cette option, vous devez indiquer le jour de chaque mois auquel la sauvegarde doit être déclenchée, ainsi que l'heure de déclenchement.
- Tous les N Mo : Cette option est utilisée pour programmer les sauvegardes en fonction de la taille du fichier journal courant. Un backup se déclenche automatiquement quand le fichier atteint la taille spécifiée. La taille limite du fichier peut être fixée à 1, 10, 100 ou 1000 Mo.

En cas de sauvegarde périodique, si le serveur Web n'était pas lancé au moment théorique de la sauvegarde, 4D considère au lancement suivant que la sauvegarde a échoué et applique les paramétrages adéquats, définis dans les Propriétés.

Web Services

Les options de cette page permettent d'activer et de configurer les services Web au sein de la base 4D, aussi bien en publication (serveur) qu'en souscription (client).

Pour plus d'informations sur la prise en charge des Services Web dans 4D, reportez-vous au chapitre [Publication et utilisation de Services Web](#).

Serveur

Cette zone affiche les options relatives à l'utilisation de 4D en tant que "serveur" de Web Services, c'est-à-dire publiant des méthodes projet sous forme de Web Services.

- Autoriser requêtes Web Services : Cette option permet d'initialiser la publication de Web Services. Si cette option n'est pas cochée, 4D refuse les requêtes SOAP et ne génère pas de WSDL — même si des méthodes disposent de l'attribut *Disponible via Web Service*. Lorsque cette option est cochée, 4D crée le fichier WSDL.
- Nom Web Service: cette zone permet de modifier le "nom générique" du Web Service. Ce nom permet de différencier les services au niveau du serveur SOAP (lorsque le serveur publie plusieurs Web Services), ainsi que dans les annuaires de Web Services. Par défaut, 4D utilise le nom A_WebService.
- Espace de nommage Web Services : cette zone permet de modifier l'espace de nommage (le namespace) des Web Services publiés par 4D. Chaque Web Service publié sur Internet doit être unique. L'unicité des noms de Web Services est assuré à l'aide des espaces de nommage XML (XML namespace). Un espace de nommage est une chaîne de caractères arbitraire permettant d'identifier de manière unique un ensemble de balises XML. Typiquement, l'espace de nommage début par l'URL de la société (<http://masociete.com/monespacedenommage>). Dans ce cas, il n'est pas indispensable qu'il y ait quelque chose à l'URL défini, il importe simplement que la chaîne de caractères utilisée soit unique. Par défaut, 4D utilise l'espace de nommage <http://www.4d.com/namespace/default>.

Conformément à la norme XML concernant les noms de balises, la chaîne de caractères utilisée ne doit pas contenir d'espaces ni débuter par un chiffre. En outre, pour éviter tout risque d'incompatibilité, il est recommandé de ne pas utiliser de caractères étendus (tels que des caractères accentués).

Client

Cette zone contient une option relative à l'utilisation de 4D en tant que "client" de Web Services, c'est-à-dire souscrivant à des services publiés sur le réseau.

- Préfixe des méthodes créées par l'assistant : cette zone vous permet de modifier le préfixe automatiquement ajouté par 4D devant le nom des méthodes proxy générées par l'assistant Web Services. Les méthodes projet proxy font le lien entre l'application 4D et le serveur de Web Services. Par défaut, 4D utilise le préfixe "proxy_".

Fonctionnalités Web

Les options de cette page permettent d'activer et de contrôler les fonctionnalités Web avancées telles que le serveur REST.

Publication

Activer le service REST

Démarre et stoppe le serveur REST. Voir [Configuration du serveur REST](#).

Accès

Cette option vous permet de désigner un groupe d'utilisateurs 4D qui sera seul autorisé à établir la connexion à la base 4D à l'aide des requêtes REST. Voir [Configuration de l'accès REST](#).

Web Studio

Activer l'accès au studio web

Active l'accès général au studio web. Vous devez également le configurer au niveau de chaque projet.

Page SQL

Cette page permet de configurer les paramètres de publication, les droits d'accès et les options du moteur du [serveur SQL 4D](#).

Publication du serveur SQL

Voir la page [Configuration du serveur SQL de 4D](#) sur doc.4d.com.

Contrôle d'accès SQL pour le schéma par défaut

Voir la page [Configuration du serveur SQL de 4D](#) sur doc.4d.com.

Options du moteur SQL

Voir le paragraphe [Options du moteur SQL](#) sur doc.4d.com.

Page PHP

Dans 4D, vous pouvez exécuter des scripts PHP en configurant directement la page PHP des Propriétés de la base de données (voir [Exécution de scripts PHP dans 4D](#) dans le manuel *Langage de 4D*).

Interpréteur

- Adresse IP et Numéro de port Par défaut, 4D fournit un interpréteur PHP, compilé en FastCGI. Pour des raisons liées à l'architecture interne, les demandes d'exécution vont vers l'interpréteur PHP à une adresse HTTP spécifique. Par défaut, 4D utilise l'adresse 127.0.0.1 et le port 8002. Vous pouvez modifier cette adresse et/ou ce port s'ils sont déjà utilisés par un autre service ou si vous avez plusieurs interpréteurs sur la même machine. Pour ce faire, modifiez les paramètres Adresse IP et Numéro de port. A noter que l'adresse HTTP doit se trouver sur la même machine que 4D.
- Interpréteur externe Si vous utilisez un interpréteur PHP externe, il doit être compilé en FastCGI et se trouver sur la même machine que 4D (voir "Utiliser un autre interpréteur PHP ou un autre fichier php.ini" dans [Exécution de scripts PHP dans 4D](#)). Sélectionnez cette option pour que 4D ne tente pas de se connecter avec l'interpréteur interne lors de l'exécution d'une requête PHP. A noter que cette configuration nécessite une exécution et un contrôle manuels de l'interpréteur externe.

4D Server : Ces paramètres sont partagés entre 4D Server et les machines distantes 4D ; il n'est donc pas possible d'utiliser un interpréteur externe sur le serveur et d'utiliser simultanément l'interpréteur interne sur les machines clientes (et vice versa). De plus, si le serveur utilise un interpréteur externe sur le port 9002, les machines clientes doivent également utiliser un interpréteur sur ce port.

Options

Ces options sont liées à la gestion automatique de l'interpréteur PHP 4D et sont désactivées lorsque l'option Interpréteur externe est sélectionnée.

- Nombre de process : L'interpréteur PHP 4D pilote un ensemble de processus d'exécution système appelés "process enfants". A des fins d'optimisation, il peut exécuter et conserver jusqu'à cinq processus enfants simultanément par défaut. Vous pouvez modifier le nombre de processus enfants en fonction de vos besoins. Par exemple, vous pouvez augmenter cette valeur si vous faites appel à l'interpréteur PHP de manière intensive. Pour plus d'informations, reportez-vous à la section "Architecture" dans [Exécution de scripts PHP dans 4D](#).

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

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

Note: Dans cette boîte de dialogue, les paramètres sont spécifiés par défaut pour toutes les machines connectées et toutes les sessions. Vous pouvez également les modifier et les lire séparément pour chaque machine et chaque session en utilisant les commandes **SET DATABASE PARAMETER** et **Get database parameter**. Les paramètres modifiés par la commande **SET DATABASE PARAMETER** sont prioritaires pour la session courante.

Page sécurité

Cette page regroupe les options relatives à la protection des accès et des données de vos applications desktop.

Note : Consultez le document [4D Security guide](#) pour une vue d'ensemble des fonctions de sécurité de 4D.

Accès des utilisateurs distants

Ces paramètres ne s'appliquent pas aux bases projets ouvertes en monoposte.

- Accès Développement et Explorateur d'exécution : permet au groupe d'utilisateurs spécifié d'accéder au mode Développement de la base et d'afficher l'Explorateur d'exécution.

A noter que :

- La définition d'un groupe d'accès en Développement permet également de désactiver l'option Créer une table de la boîte de dialogue d'import de données. Pour plus d'informations sur cette boîte de dialogue, reportez-vous à la section [Importer des données depuis des fichiers](#).
 - Le Super_Utilisateur et l'Administrateur ont toujours accès au mode Développement et à l'Explorateur d'exécution, même s'ils ne font pas explicitement partie du groupe d'accès défini.
- Utilisateur par défaut : lorsqu'un utilisateur est défini dans ce menu, chaque utilisateur ouvrant ou se connectant à la base dispose des priviléges et restrictions d'accès qui ont été assignés à cet utilisateur par défaut. Il n'est alors plus nécessaire de saisir un nom d'utilisateur. De plus, si vous n'avez pas associé de mot de passe à l'utilisateur par défaut, la boîte de dialogue de saisie du mot de passe n'apparaît pas, la base s'ouvre directement. L'intérêt de cette option est de simplifier les accès multiples à la base tout en maintenant un système complet de contrôle des données.
 - Si vous avez associé un mot de passe à l'utilisateur par défaut, une boîte de dialogue apparaît à l'ouverture de la base, les utilisateurs doivent uniquement saisir un mot de passe.
 - Si vous n'avez pas associé de mot de passe à l'utilisateur par défaut, aucune boîte de dialogue n'apparaît. Note : Vous pouvez "forcer" l'affichage de la boîte de dialogue standard de saisie du mot de passe lorsque le mode "Utilisateur par défaut" est actif, par exemple pour pouvoir vous connecter en tant que Super_Utilisateur ou Administrateur. Pour cela, ouvrez (ou connectez-vous à) la base de données tout en maintenant la touche Majuscule enfoncee.
 - Afficher les utilisateurs dans la fenêtre Mots de passe : si cette option est cochée, les utilisateurs doivent choisir leur nom dans une liste d'utilisateurs et saisir leur mot de passe dans la boîte de dialogue de saisie des mots de passe. Si cette option n'est pas cochée, les utilisateurs doivent saisir leur nom et leur mot de passe. Pour plus d'informations sur les deux versions de la boîte de dialogue de mots de passe, reportez-vous au paragraphe "Accès aux bases protégées" dans la section [Présentation du contrôle des accès](#).
 - Trier la liste des utilisateurs par ordre alphabétique (option utilisable uniquement si la précédente est cochée) : lorsque cette option est sélectionnée, la liste des utilisateurs dans la boîte de dialogue de saisie des mots de passe est triée par ordre alphabétique.
 - Les utilisateurs peuvent changer leur mot de passe : lorsque cette option est cochée, un bouton Changer est affiché dans la boîte de dialogue d'ouverture ou de connexion à la base. Ce bouton permet à l'utilisateur d'accéder à la boîte de dialogue lui permettant de modifier son mot de passe (cf. paragraphe "Modification du mot de passe par l'utilisateur" dans la section [Assurer la maintenance du système](#)). Si vous le souhaitez, vous pouvez masquer le bouton Changer afin de ne pas permettre aux utilisateurs de modifier leur mot du passe. Il suffit pour cela de désélectionner cette option.

Options

- Filtrage des commandes et méthodes projet dans l'éditeur de formules et documents 4D Write Pro Pour des raisons de sécurité, 4D limite par défaut les accès aux commandes, fonctions et méthodes projets dans l'[Editeur de](#)

[formules](#) du mode Application : seules certaines fonctions et méthodes projets ayant été explicitement déclarées à l'aide de la commande [SET ALLOWED METHODS](#) peuvent être utilisées. A l'aide des options suivantes, vous pouvez supprimer complètement ou partiellement ce filtrage.

- Activé pour tous (par défaut) : L'accès aux commandes, fonctions et méthodes projets est limité pour tous les utilisateurs, y compris au Super Utilisateur et à l'Administrateur.
 - Désactivé pour le Super Utilisateur et l'Administrateur : Cette option accorde, uniquement au Super Utilisateur et à l'Administrateur, un accès complet aux commandes et méthodes 4D. Elle peut être utilisée pour définir un mode d'accès illimité aux commandes et méthodes tout en gardant le contrôle des actions effectuées. En phase de développement, ce mode peut être utilisé pour tester librement toutes les formules, les états, etc. En cours d'exploitation, il peut être utilisé pour mettre en oeuvre des solutions sécurisées permettant un accès temporaire à toutes les commandes et méthodes. Le principe consiste à changer l'utilisateur courant (via la commande [CHANGE CURRENT USER](#)) avant d'appeler un dialogue ou de démarrer un process d'impression qui requiert un accès total aux commandes, puis de retourner à l'utilisateur initial une fois l'opération terminée.
Note : Si l'accès complet a été activé à l'aide de l'option précédente, cette option n'aura pas d'effet.
 - Désactivé pour tous : Cette option désactive le contrôle dans les formules. Lorsque cette option est cochée, les utilisateurs ont accès à l'ensemble des commandes, plug-ins et méthodes projets (à l'exception de ceux qui sont invisibles). Note : Cette option est prioritaire sur la commande [SET ALLOWED METHODS](#). Lorsqu'elle est cochée, cette commande ne fait rien.
- Autoriser les propriétés utilisateur : Vous devez cocher cette option si vous souhaitez utiliser la fonctionnalité d'externalisation des propriétés utilisateur. Lorsque cette option est cochée, jusqu'à trois boîtes de dialogue sont disponibles pour définir les propriétés : Propriétés structure, Propriétés utilisateur, et Propriétés utilisateur pour fichier de données. Pour plus d'informations, reportez-vous à la section [User settings](#).
 - Exécuter la méthode "Sur événement base hôte" des composants : La méthode base [Sur événement base hôte](#) permet de faciliter les phases d'initialisation et de sauvegarde des composants 4D. Pour des raisons de sécurité, vous devez autoriser explicitement l'exécution de cette méthode dans chaque base hôte. Pour cela, vous devez cocher l'option. Cette option n'est pas cochée par défaut.

Lorsque cette option est cochée :

- les composants 4D sont chargés,
- chaque méthode base [Sur événement base hôte](#) des composants (s'il y en a) est appelée par la base hôte,
- le code de la méthode est exécuté.

Lorsque cette option n'est pas cochée :

- les composants 4D sont chargés mais ils doivent gérer eux-mêmes leurs phases d'initialisation et de sauvegarde.
- le développeur du composant doit publier les méthodes du composant qui doivent être appelées par la base hôte lors de ces phases (démarrage et fermeture)
- le développeur de la base hôte doit appeler les méthodes appropriées du composant au bon moment (doit être exposé dans la documentation du composant).

Page de compatibilité

La page Compatibilité regroupe les paramètres relatifs au maintien de la compatibilité avec les versions précédentes de 4D.

ainsi que des paramétrages modifiés dans ce(tte) base/projet.

Cette page liste les options de compatibilité disponibles pour les bases/projets converti(e)s depuis 4D v18 et les versions plus récentes. Pour les options de compatibilité plus anciennes, consultez la [page Compatibilité](#) sur doc.4d.com.

- Utiliser l'ancienne couche réseau : à compter de 4D v15, les applications 4D proposent une nouvelle couche réseau, nommée *ServerNet*, pour 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 en fonction de vos besoins. *ServerNet* est automatiquement activé dans les nouvelles bases et les bases converties depuis les versions 15 et les versions plus récentes. A noter qu'en cas de modification de l'option, vous devez redémarrer l'application pour que le changement soit pris en compte. Toute application cliente qui était connectée doit également être redémarrée afin de se connecter avec la nouvelle couche réseau. Note : Cette option peut également être gérée par programmation via la commande `SET DATABASE PARAMETER`.
- Utiliser XPath standard : Par défaut, cette option est décochée pour les bases converties à partir d'une version 4D antérieure à la v18 R3 et est cochée pour les bases créées sous une version 4D v18 R3 ou une version plus récente. A partir de v18 R3, l'implémentation de XPath dans 4D a été modifiée pour une meilleure conformité et pour la prise en charge d'un plus grand nombre de prédictats. Par conséquent, les fonctionnalités non standard de l'implémentation antérieure ne fonctionnent plus. Elles incluent :
 - le caractère "/" initial n'est pas seulement le noeud racine - l'utilisation du caractère / comme premier caractère d'une expression XPath ne déclare pas un chemin absolu à partir du noeud racine
 - pas de noeud courant implicite - le noeud courant doit être intégré dans l'expression XPath
 - pas de requêtes récursives dans les structures répétées - seul le premier élément est parsé. ¥

Même si ces fonctionnalités ne sont pas standard, vous pouvez continuer de les utiliser afin que votre code continue de fonctionner -- dans ce cas, l'option doit simplement être *décochée*. Par ailleurs, si votre code ne dépend pas de l'implémentation non standard et si vous souhaitez profiter des fonctionnalités XPath avancées dans vos bases (cf. commande [DOM Find XML element](#)), assurez-vous que l'option Utiliser XPath standard est *cochée*.

- Utiliser LF comme caractère de fin de ligne sur macOS : À partir de 4D v19 R2 (et 4D v19 R3 pour les fichiers XML), 4D écrit des fichiers texte avec un saut de ligne ("line feed" (LF)) comme caractère de fin de ligne ("End Of Line" (EOL)) par défaut au lieu de Retour Chariot ("Carriage Return" CR)) sur macOS dans les nouveaux projets. Si vous souhaitez bénéficier de ce nouveau comportement dans les projets convertis à partir de versions antérieures de 4D, cochez cette option. Voir [TEXT TO DOCUMENT](#), [Document to text](#), et [XML SET OPTIONS](#).
- Ne pas ajouter de BOM lors de l'écriture d'un fichier texte unicode par défaut : À partir de 4D v19 R2 (et 4D v19 R3 pour les fichiers XML), 4D écrit des fichiers texte sans BOM ("Byte order mark") par défaut. Dans les versions antérieures, les fichiers texte étaient écrits avec un BOM par défaut. Sélectionnez cette option si vous souhaitez activer le nouveau comportement dans les projets convertis. Voir [TEXT TO DOCUMENT](#), [Document to text](#), et [XML SET OPTIONS](#).
- Map NULL values to blank values unchecked by default a field creation : Pour une meilleure conformité avec les spécifications ORDA, dans les bases de données créées avec 4D v19 R4 et avec des versions plus récentes, la propriété du champ Map NULL values to blank values n'est pas cochée par défaut lorsque vous créez des champs. Vous pouvez appliquer ce comportement par défaut à vos bases de données converties en cochant cette option (il est recommandé de travailler avec des valeurs Null car elles sont entièrement prises en charge par [ORDA](#)).

Page Général

Cette page regroupe diverses options permettant de paramétrer le fonctionnement général de votre application 4D.

Options

Au démarrage

Cette option permet de configurer l'affichage proposé par défaut par 4D au démarrage, lorsque l'utilisateur lance uniquement l'application.

- Ne rien faire : seule la fenêtre de l'application apparaît, vide.
- Dialogue d'ouverture de base de données locale : 4D affiche une boîte de dialogue standard d'ouverture de documents, permettant de désigner un projet local.
- Ouvrir le dernier projet utilisé : 4D ouvre directement le dernier projet utilisé, aucune boîte de dialogue d'ouverture n'apparaît. >Pour forcer l'affichage de la boîte de dialogue d'ouverture lorsque cette option est sélectionnée, maintenez enfoncée la touche Alt (Windows) ou Option (macOS) pendant le lancement du projet.
- Dialogue d'ouverture du projet distant : 4D affiche la boîte de dialogue standard de connexion avec 4D Server, permettant de désigner une base de données publiée sur le réseau.
- Dialogue Assistant de bienvenue (réglage d'usine) : 4D affiche la boîte de dialogue de l'Assistant de bienvenue.

4D Server : L'application 4D Server ne tient pas compte de cette option. Dans cet environnement, le mode Ne rien faire est toujours utilisé.

Création de formulaire automatique

Cette option n'est utilisée que dans les bases de données binaires ; elle est ignorée dans l'architecture projets.
Voir doc.4d.com.

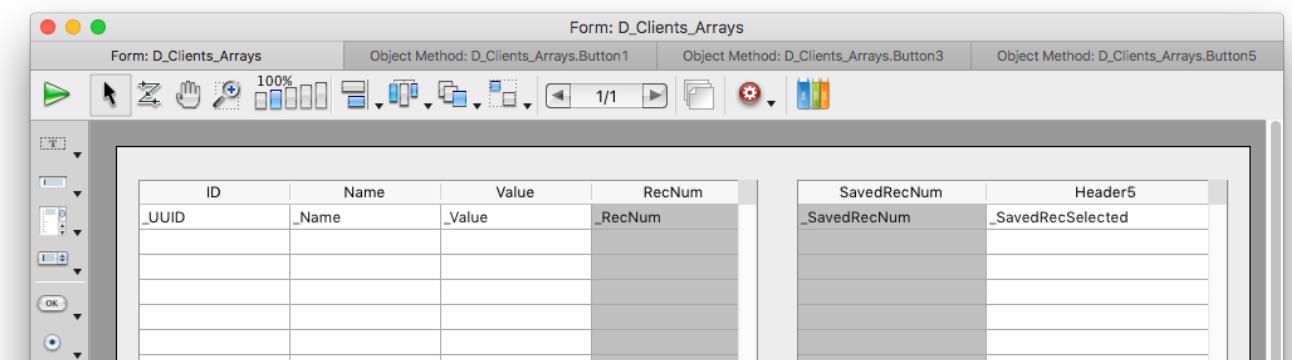
Fenêtre à onglets (macOS seulement)

Depuis macOS Sierra, les applications Mac bénéficiant d'onglets automatiques facilitent l'organisation de l'écran lors de l'ouverture de fenêtres multiples : les fenêtres documents sont regroupées dans une seule fenêtre parente et sont accessibles via des onglets. Cette fonctionnalité est particulièrement utile avec les écrans de taille réduite ou lorsqu'un trackpad est utilisé.

Vous pouvez bénéficier de cette fonctionnalité dans les environnements de 4D suivants (versions 64 bits uniquement) :

- Fenêtres de l'éditeur de méthodes
- Fenêtres de l'éditeur de formulaires

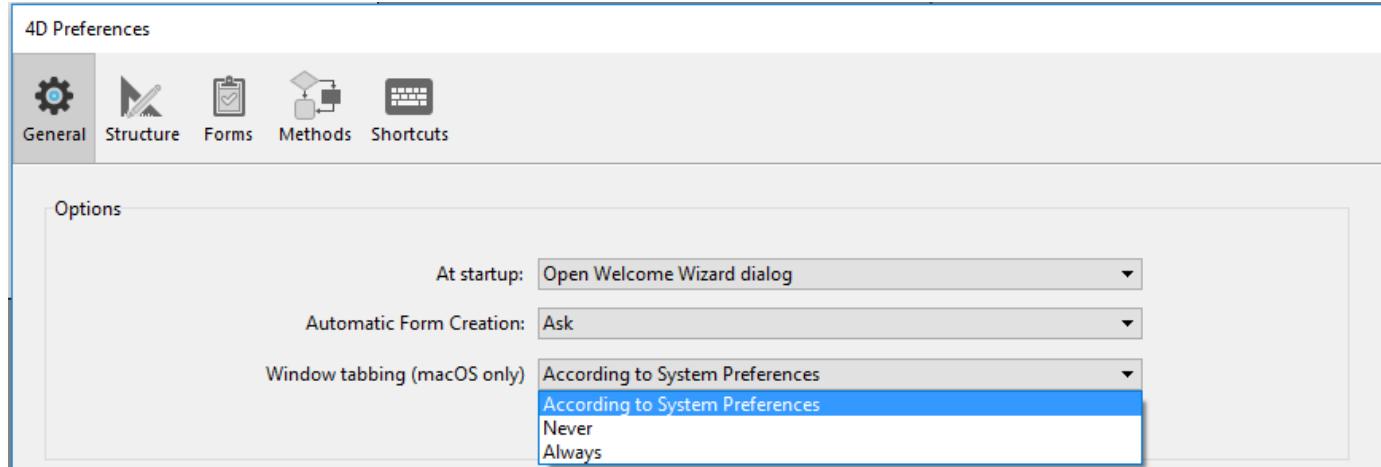
Toutes les fenêtres de ces éditeurs peuvent être ouvertes sous forme d'onglets :



Un ensemble de commandes du menu Fenêtre permet de gérer les onglets :



Dans la boîte de dialogue des Préférences de 4D, l'option Fenêtre à onglets vous permet de contrôler ce fonctionnement :



Trois options sont disponibles :

- Selon les préférences système (défaut) : les fenêtres de 4D se comporteront comme elles ont été définies dans les Préférences Système de macOS (En plein écran uniquement, Toujours ou Manuellement).
- Jamais : L'ouverture d'un nouveau document dans l'éditeur de formulaires ou l'éditeur de méthodes de 4D provoquera toujours la création d'une nouvelle fenêtre (les onglets ne sont jamais créés).
- Toujours : L'ouverture d'un nouveau document dans l'éditeur de formulaires ou l'éditeur de méthodes de 4D provoquera l'ajout d'un onglet.

Apparence (macOS uniquement)

Ce menu vous permet de sélectionner la palette de couleurs à utiliser pour l'environnement de développement 4D. La palette spécifiée sera appliquée à tous les éditeurs et fenêtres du mode Développement.

Vous pouvez également définir la palette de couleurs à utiliser dans vos applications de bureau dans la page "Interface" de la boîte de dialogue des Paramètres.

Trois options sont disponibles :

- Selon les préférences de la palette de couleurs du système (par défaut) : Utilisez la palette de couleurs définie dans les préférences système de macOS.
- Clair : Utiliser le thème clair
- Sombre : utiliser le thème sombre

Cette préférence n'est prise en charge que sur macOS. Sous Windows, la palette "Light" est toujours utilisée.

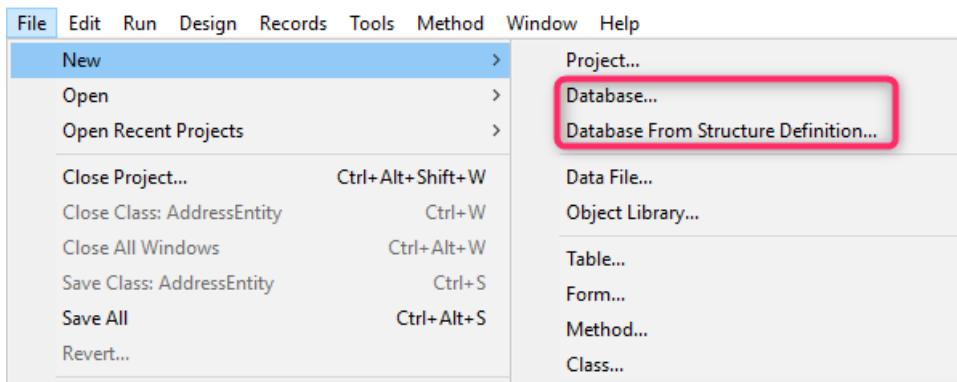
Quitter le mode Développement lors du passage en mode Application

Si cette option est cochée, lorsque l'utilisateur passe en mode Application via la commande de menu Tester l'application, toutes les fenêtres du mode Développement sont fermées. Si cette option n'est pas cochée (valeur par défaut), les fenêtres du mode Développement restent affichées à l'arrière-plan du mode Application.

Activer la création de bases de données en binaire

Si vous sélectionnez cette option, deux éléments s'ajoutent au menu Fichier > Nouveau et au bouton Nouveau de la barre d'outils :

- Base de données...
- Base de données à partir d'une définition de structure...



Ces éléments vous permettent de créer des bases binaires (voir la section [Créer une nouvelle base](#)). Ils ne sont plus proposés par défaut car 4D recommande, pour les nouveaux développements, l'utilisation d'une architecture de bases projets.

Lors de la création d'un nouveau projet

Utiliser le fichier d'historique

Lorsque cette option est cochée, un fichier d'historique est automatiquement démarré et utilisé dans chaque nouvelle base. Pour plus d'informations, reportez-vous à la section [Fichier d'historique \(.journal\)](#).

Créer un paquet

Lorsque cette option est cochée, les bases de données 4D sont automatiquement créées dans un dossier suffixé .4dbase.

Grâce à ce principe, sous macOS les dossiers des bases apparaissent sous forme de paquets (packages) disposant de propriétés spécifiques. Sous Windows, ce fonctionnement n'a pas d'incidence particulière.

Inclure les tokens dans les fichiers sources des projets

Lorsque cette option est cochée, les [fichiers sources des méthodes](#) enregistrées dans les nouveaux projets 4D contiendront des tokens pour les objets classiques du langage et de la base (constantes, commandes, tables et champs). Les tokens sont des caractères supplémentaires tels que :C10 ou :5 insérés dans les fichiers de code source, qui permettent de renommer les tables et les champs et d'identifier les éléments indépendamment de la version 4D (voir [Utilisation des tokens dans les formules](#)).

Si vous avez l'intention d'utiliser des VCS ou des éditeurs de code externes avec vos nouveaux projets, il est préférable de décocher cette option pour une meilleure lisibilité du code avec ces outils.

Vous pouvez toujours obtenir le code avec les tokens en appelant la `METHOD GET CODE` avec 1 dans le paramètre *option*.

Vous pouvez toujours obtenir le code avec les tokens en appelant la `METHOD GET CODE` avec 1 dans le paramètre *option*.

Exclusion des tokens dans les projets existants

Vous pouvez configurer vos projets existants pour enregistrer le code sans tokens en insérant la clé suivante dans le

fichier `<applicationName>.4DProject` à l'aide d'un éditeur de texte :

```
"tokenizedText" : false
```

Ce paramètre n'est pris en compte que lors de l'enregistrement des méthodes. Les méthodes existantes dans vos projets ne sont pas modifiées, sauf si vous les enregistrez à nouveau.

Créer le fichier `.gitignore`

Si vous avez besoin ou souhaitez que git ignore certains fichiers dans vos nouveaux projets.

Vous pouvez définir cette préférence en cochant l'option `Créer le fichier .gitignore`.

Lorsqu'un projet est créé dans 4D et que cette case est cochée, 4D crée un fichier `.gitignore` au même niveau que le dossier `Project` (voir [Architecture d'un projet](#)).

Vous pouvez définir le contenu par défaut du fichier `.gitignore` en cliquant sur l'icône du crayon. Cela ouvrira le fichier de configuration `.gitignore` dans votre éditeur de texte. Le contenu de ce fichier sera utilisé pour générer les fichiers `.gitignore` dans vos nouveaux projets.

La [documentation officielle de git](#) est une excellente ressource pour comprendre le fonctionnement des fichiers `.gitignore`.

Langue de comparaison de texte

Ce paramètre permet de définir la langue utilisée par défaut pour le traitement et la comparaison des chaînes de caractères dans les nouvelles bases. Le choix d'une langue de comparaison influe sur le tri et la recherche des textes ainsi que le passage en minuscules/majuscules mais n'a pas d'incidence sur la traduction des libellés ou sur les formats de dates, d'heure ou monétaires qui restent, eux, dans la langue du système. Par défaut (réglage d'usine), 4D utilise la langue courante de l'utilisateur définie dans le système.

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

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

Cette option n'est prise en compte que pour l'accès à la documentation des commandes (à l'exclusion des fonctions de classe).

Emplacement de la documentation

Cette zone permet de configurer l'accès à la documentation HTML de 4D qui s'affiche dans votre navigateur courant :

- Lorsque vous appuyez sur la touche F1 pendant que le curseur est inséré dans une fonction de classe 4D ou un nom de commande dans l'éditeur de méthode ;
- Lorsque l'utilisateur double-clique sur une commande dans la page des commandes de l'Explorateur .

Langue de documentation

Langue de la documentation HTML à afficher. Vous pouvez sélectionner une documentation dans une langue différente de celle de l'application.

Commencer par consulter le dossier local

Cette option n'est prise en compte que pour l'accès à la documentation des commandes (à l'exclusion des fonctions de classe).

Définit l'emplacement des pages de documentation recherchées par 4D.

- Lorsque cette option est cochée (par défaut), 4D recherche d'abord la page dans le dossier local (voir ci-dessous). Si la page est trouvée à l'emplacement défini, elle est affichée dans le navigateur courant. Si la page est trouvée à l'emplacement défini, elle est affichée dans le navigateur courant. Ce principe permet par exemple de travailler en mode déconnecté, en accédant à une version locale de la documentation.
- If it is not found, 4D displays an error message in the browser. Si elle n'est pas trouvée, 4D affiche un message d'erreur dans le navigateur.

Dossier local

Cette option n'est prise en compte que pour l'accès à la documentation des commandes (à l'exclusion des fonctions de classe).

Indique l'emplacement de la version statique de la documentation HTML. Par défaut, cet emplacement correspond au sous-dossier `¥Help¥Command¥langue`. Vous pouvez le visualiser en affichant le menu associé à la zone (clic sur la zone). Si le sous-dossier n'est pas présent, l'emplacement est affiché en rouge.

Vous pouvez modifier cet emplacement si vous le souhaitez, par exemple pour afficher la documentation dans une langue différente de celle de l'application. La documentation HTML statique peut être située sur un autre volume, un serveur web, etc. Pour désigner un autre emplacement, cliquez sur le bouton [...] à côté de la zone de saisie et choisissez un dossier racine de documentation (dossier correspondant à la langue : `fr`, `en`, `es`, `de` ou `ja`).

Page Structure

Clé primaire

Ces options des préférences permettent de modifier le nom et le type par défaut des champs clés primaires ajoutés automatiquement par 4D lors de la création de nouvelles tables ou via le [Gestionnaire de clés primaires](#).

Les options suivantes sont disponibles :

- Nom ("ID" par défaut) : Définit le nom par défaut des champs clés primaires. Vous pouvez utiliser le nom que vous souhaitez, dans le respect des [règles de nommage des tables standard de 4D](#) .
- Type ([Entier long](#) par défaut) : Définit le type des champs clés primaires par défaut. Vous pouvez choisir le type UUID. Dans ce cas, les champs clés primaires créés par défaut seront de [type Alpha](#) et auront les propriétés UUID Format et Auto UUID cochées.

Éditeur de structure

Ce groupe d'options permet de configurer l'affichage de l'éditeur de structure de 4D.

Qualité graphique de la structure

Cette option permet de faire varier le niveau de détail graphique de l'éditeur de structure. Par défaut, la qualité est définie sur Haute. Vous pouvez sélectionner la qualité Standard afin de privilégier la rapidité d'affichage. L'effet de ce paramétrage est principalement perceptible lors de l'utilisation de la fonction de zoom (cf. paragraphe "Zoom" dans la section [Éditeur de structure](#)).

Quand un dossier est masqué, son contenu est :

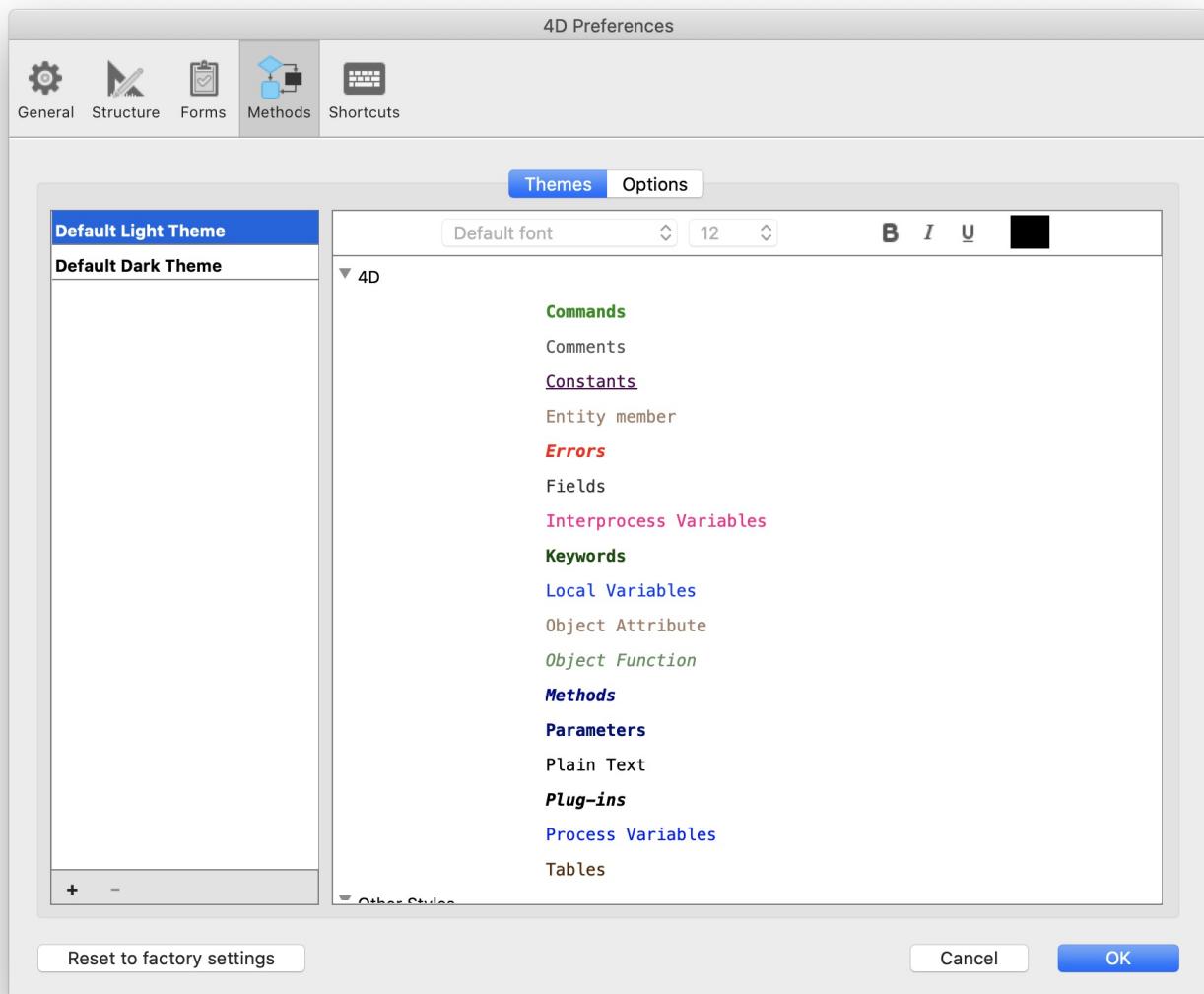
Cette option permet de configurer l'apparence graphique des tables masquées dans l'éditeur de structure, lorsque vous effectuez des sélections par dossier (cf. paragraphe [Filtrage des tables par dossier](#)). Les options possibles sont Atténué (une ombre subsiste à l'emplacement de la table) et Invisible (la table disparaît entièrement).

Page Méthodes

Cette page contient des paramètres permettant de définir l'interface, l'affichage par défaut et les options de l'éditeur de méthodes. Elle est divisée en deux parties, accessibles via les onglets Thèmes et Options.

Thèmes

Cette page permet de sélectionner, de créer ou de configurer les thèmes de l'éditeur de méthode. Un thème définit la police, la taille de la police, les couleurs et les styles des éléments de l'éditeur de code.



Liste des thèmes

Selectionnez, dans cette liste, le thème à appliquer à l'éditeur de code. Tous les thèmes disponibles sont affichés, y compris les thèmes personnalisés (le cas échéant). 4D propose deux thèmes par défaut :

- Default Light Theme
- Default Dark Theme

Les thèmes par défaut ne peuvent être ni modifiés ni supprimés.

Un thème myTheme est automatiquement ajouté si vous avez déjà personnalisé les styles de l'éditeur de méthodes

dans les précédentes versions de 4D.

Création de thèmes personnalisés

Vous pouvez créer des thèmes à personnaliser entièrement. Pour créer un thème, sélectionnez un thème existant et cliquez sur le + en bas de la liste des thèmes. Vous pouvez également ajouter des thèmes personnalisés en copiant les fichiers de thème dans le dossier `4D Editor Themes` (voir ci-dessous).

Fichiers de thèmes personnalisés

Chaque thème personnalisé est stocké dans un seul fichier JSON nommé `themeName.json`. Les fichiers JSON des thèmes personnalisés sont stockés dans le dossier `4D Editor Themes` situé au même niveau que le [fichier de préférences](#) de 4D.

Si des valeurs clés ne sont pas définies dans un thème personnalisé, elles prennent par défaut les valeurs du thème `Default Light Theme`. Si un fichier de thème JSON est invalide, le thème `Default Light Theme` est chargé et une erreur est générée.

Lorsqu'un fichier de thème est modifié par un éditeur externe, 4D doit être redémarré pour prendre en compte la ou les modifications.

Définition d'un thème

Définir un thème signifie :

- définir une police et une taille de police pour l'ensemble de l'éditeur de code,
- attribuer des styles et des couleurs spécifiques à chaque élément de langage 4D (champs, tables, variables, paramètres, SQL, etc.), à chaque élément de langage SQL (mots-clés, fonctions, etc.) et aux fonds de couleur.

La combinaison de couleurs et de styles différents est particulièrement utile à des fins de maintenance du code.

Police et taille de police

Les menus Police et Taille de la police vous permettent de sélectionner le nom et la taille de la police utilisés dans la zone de saisie de l'éditeur de méthode pour toutes les catégories.

Langage 4D et langage SQL

Vous pouvez définir différents styles et couleurs de police (couleur de police ou couleur de fond) pour chaque type d'élément de langage. Vous pouvez sélectionner le ou les éléments à personnaliser dans la liste Catégorie.

Autres styles

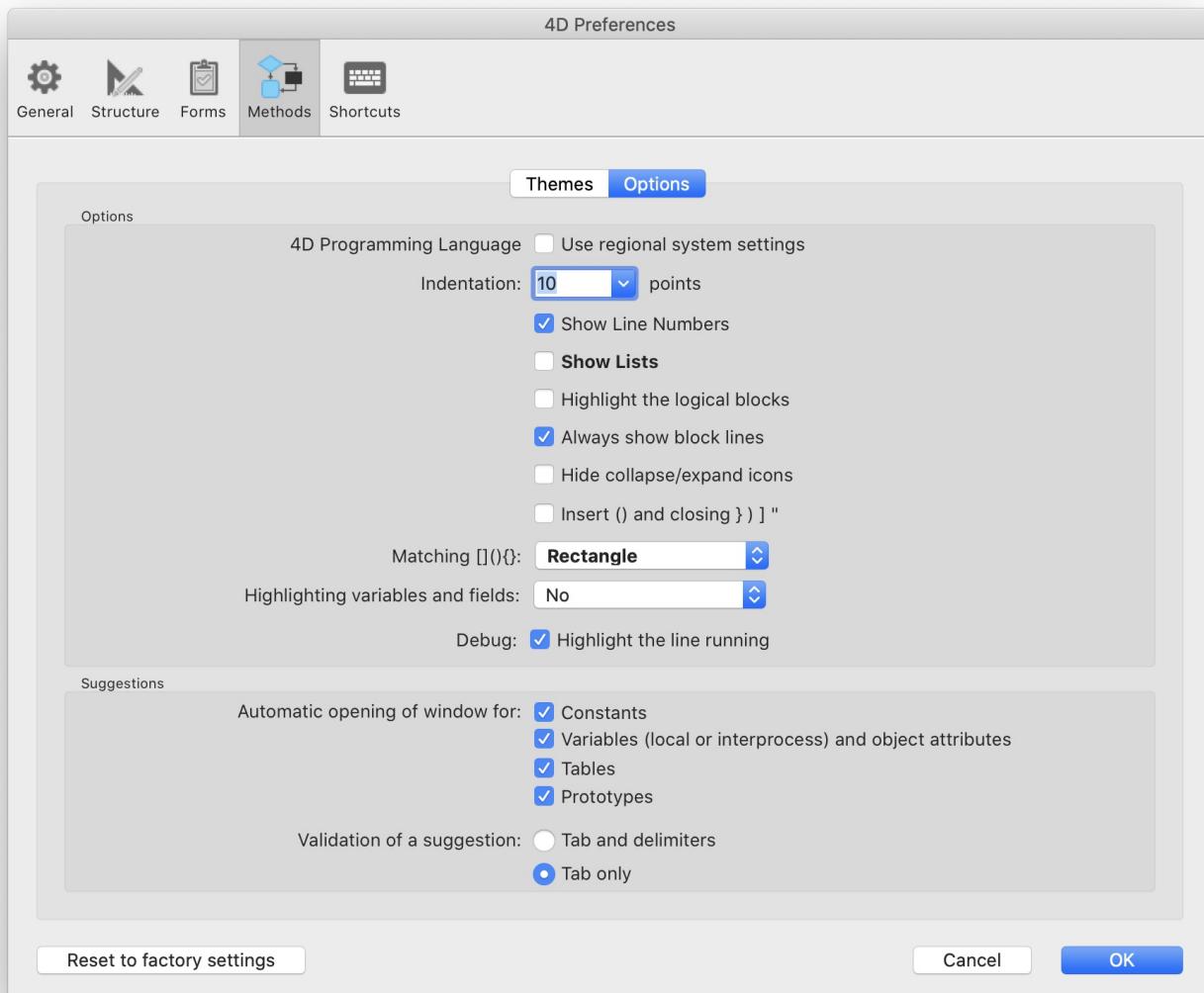
Ces options permettent de configurer les différentes couleurs utilisées dans les interfaces de l'éditeur de méthode et du débogueur.

Category	Element
> 4D Language	
▼ Other Styles	
	Background color
	Cursor line background color
	Highlight of the running line in the debugger
	Execution line background color
	Border of the running line in the debugger
	Highlight background color
	Highlight of the blocks
	Highlight of the parentheses
	Highlight of the found words
	Suggested text
	Selection back color
> SQL Language	

	Description
Couleur de fond	Couleur de fond de la fenêtre de l'éditeur de la méthode.
Bordure de la ligne en cours d'exécution dans le débogueur	Couleur de la bordure entourant la ligne en cours d'exécution dans le débogueur lorsque l'option "Surligner la ligne en cours d'exécution" est activée dans la page Options .
Couleur de fond de la ligne du curseur	Couleur de fond de la ligne contenant le curseur.
Couleur de fond de la ligne en cours d'exécution	Couleur de fond de la ligne en cours d'exécution dans le débogueur.
Mise en évidence des mots trouvés	Couleur de surlignage des mots trouvés à l'issue d'une recherche.
Mise en évidence des parenthèses	Couleur de mise en évidence des parenthèses correspondantes (utilisée lorsque des paires de parenthèses sont signalées par la mise en évidence, voir Options).
Mise en évidence des blocs	Couleur de mise en évidence des blocs logiques sélectionnés lorsque l'option "Mise en évidence des blocs logiques" est activée dans les Options .
Mise en évidence de la même variable ou du même champ	Couleur de mise en évidence pour les autres occurrences de la même variable ou du même texte de champ lorsque l'option "Mise en évidence des variables et du texte" est activée dans les Options .
Mise en évidence de la ligne courante dans le débogueur	Couleur de mise en évidence de la ligne courante dans le débogueur lorsque l'option "Surlignage de la ligne en cours" est activée dans les Options .
Couleur de fond de la sélection	Couleur de fond de la sélection.
Texte suggéré	Couleur du texte de l'autocomplétion suggéré par l'éditeur de méthode.

Options

Cette page configure les options d'affichage de l'éditeur de méthodes.



Options

Langage de programmation 4D (Utiliser les paramètres régionaux du système)

Permet de désactiver/activer les paramètres du code "international" pour l'application 4D locale.

- non coché (par défaut) : Les paramètres Anglais-US et la langue de programmation anglaise sont utilisés dans les méthodes 4D.
- coché : Les paramètres régionaux sont utilisés dans les méthodes 4D.

Si vous modifiez cette option, vous devez redémarrer l'application 4D pour que le changement soit pris en compte.

Indentation

Modifie la valeur d'indentation du code 4D dans l'éditeur de méthodes. La largeur doit être définie en points (10 par défaut).

Le code 4D est automatiquement indenté afin de faire apparaître sa structure :

```

1  If ($vListItemPos#0)
2      // Get the list item information
3      GET LIST ITEM(hList;$vListItemPos;$v
4          // Is the item a Department item
5      If ($vListItemRef ?? 31)
6          // If so, it is a double-click
7          ALERT("You double-clicked on the
8      Else
9          // If not, it is a double-click
10         // Using the parent item ID to
11         $vDepartmentID:=List item parent
12         QUERY([Departments];[Departments]
13             // Tell where the Employee is
14             ALERT("You double-clicked on the
15         End if
16     End if
17
18  standard indentation

```

Modifier cette valeur par défaut peut être utile si vos méthodes contiennent des algorithmes complexes avec de nombreux niveaux d'intégration. Une indentation plus étroite peut être utilisée afin de limiter le défilement horizontal.

Afficher les numéros de ligne

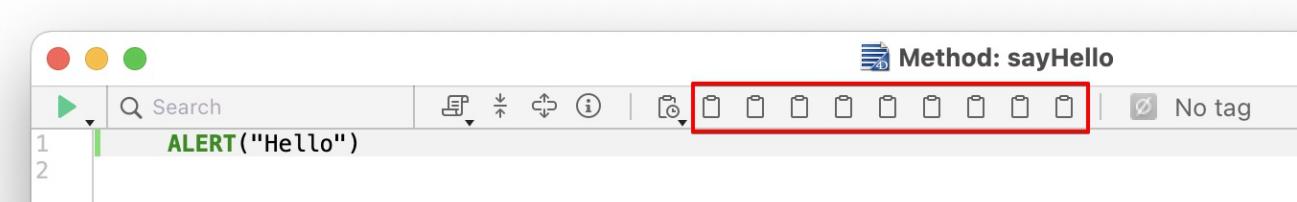
Cela vous permet d'afficher les numéros de ligne par défaut dans chaque fenêtre de l'éditeur de méthode. Vous pouvez également afficher/masquer les numéros de ligne pour la fenêtre courante directement depuis l'éditeur de méthode.

Afficher les listes

Cela vous permet de choisir d'afficher ou non les listes d'objets (Commandes, Tables et champs, etc.) par défaut lorsque la fenêtre de l'éditeur de méthode est ouverte. Vous pouvez également afficher ou masquer chaque liste directement à partir de l'éditeur de méthode.

Afficher les presse-papiers

Vous permet de choisir d'afficher ou non les multiples presse-papiers dans l'éditeur de code.



Les [raccourcis du presse-papiers](#) correspondants demeurent actifs lorsque ces presse-papiers sont masqués.

Surbrillance des blocs logiques

Lorsque vous cochez l'option, tout le code correspondant à un bloc logique (Si/Fin de si par exemple) est surligné lorsque la souris est placée au-dessus de l'icône plier/déplier :

```

12  If (<>PS_EditMovies=0)
13      <>PS_EditMovies:=New process ($CurrentMethName;
14
15      Else
16          BRING TO FRONT(<>PS_EditMovies)
17      End if

```

La couleur de surlignage peut être modifiée dans la page [Définition d'un thème](#).

Toujours afficher les lignes de blocs

Permet de cacher de façon permanente les lignes verticales de blocs. Les lignes de blocs sont conçues pour visualiser les blocs logiques. Par défaut, elles sont toujours affichées, excepté lorsque les icônes plier/déplier sont masquées (voir ci-dessous).

9	└ If (Count	9	└ If (Count
10		10	
11	\$CurrentM	11	\$CurrentM
12	└ If (<>PS_	12	└ If (<>PS_
13	<>PS_Ed:	13	└ <>PS_Ed:
14	└ Else	14	└ Else
15	BRING_TC	15	BRING_TC
16	End if	16	End if
17		17	
18	└ Else	18	└ Else
19		19	

Masquer les icônes plier/déplier

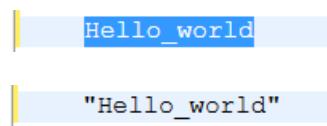
Permet de masquer, par défaut, toutes les icônes plier/déplier lorsque vous affichez le code. Lorsque cette option est cochée, les icônes plier/déplier (ainsi que les lignes de blocs locales, voir ci-dessus) sont affichées temporairement lorsque la souris survole une icône :

9	└ If (Count	9	└ If (Count
10		10	
11	\$CurrentM	11	\$CurrentM
12	└ If (<>PS_	12	└ If (<>PS_
13	<>PS_Ed:	13	└ <>PS_Ed:
14	└ Else	14	└ Else
15	BRING_TC	15	BRING_TC
16	End if	16	End if
17		17	
18	└ Else	18	└ Else
19		19	

Insérer () et ajouter })] " fermants

Active l'insertion automatique de () et de caractères fermants lors de la saisie du code. Cette option contrôle deux fonctionnalités automatiques :

- paire de parenthèses () : Ajoutée après une commande 4D, un mot-clé ou une méthode projet insérée à partir d'une liste de suggestions ou de complétion, si l'élément inséré nécessite un ou plusieurs arguments obligatoires. Par exemple, si vous tapez "C_OB" et vous appuyez sur la touche Tab, 4D écrit "C_OBJECT()" et place le point d'insertion à l'intérieur du () .
- fermeture },),] ou " : caractère ajouté lorsque vous tapez respectivement une ouverture {, (, [ou ". Cette fonction permet d'insérer des paires de symboles correspondants au point d'insertion ou autour d'un texte sélectionné. Par exemple, si vous mettez en surbrillance une chaîne de caractères et que vous tapez un simple caractère ", toute la chaîne sélectionnée sera entourée de "" :



Correspondance [](){}()

Permet de modifier la signalisation graphique des caractères d'encadrement correspondants dans le code. Cette signalisation apparaît lorsqu'un crochet, une parenthèse ou une accolade est sélectionné(e). Les options suivantes sont disponibles :

- Aucun : Aucun signal
- Rectangle (défaut): Caractères encadrés par un filet noir
`INSERT MENU ITEM(main_bar;-1;Get_indexed_string(79;1);FileMenu)`
- Couleur de fond : Caractères mis en surbrillance (la couleur est définie dans la page [Thème](#)).
- Gras : caractères affichés en gras.

Surbrillance des variables et champs

Permet de mettre en surbrillance toutes les occurrences d'un(e) même variable ou champ dans une fenêtre de méthode

ouverte.

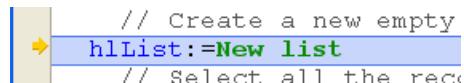
```
4 C_LONGINT (<>PS_EditMovies)
5 C_LONGINT ($Window)
6 C_TEXT ($CurrentMethName)
7 C_LONGINT ($Win)
8
9 If (Count parameters=0)
10
11 $CurrentMethName:=Current method
12 If (<>PS_EditMovies=0)
13   <>PS_EditMovies:=New process ($C:
14 Else
15   BRING TO FRONT (<>PS_EditMovies)
16 End if
```

- Non (par défaut) : Pas de surbrillance
- Sur curseur : Toutes les occurrences sont mises en évidence lorsque le texte est cliqué
- Sur sélection : Toutes les occurrences sont mises en évidence lorsque le texte est sélectionné

La couleur de surlignage peut être modifiée dans la page [Définition d'un thème](#).

Mode trace (Surbrillance de la ligne en exécution)

Permet de mettre en surbrillance la ligne en exécution dans le Débogueur, en plus de la flèche jaune.

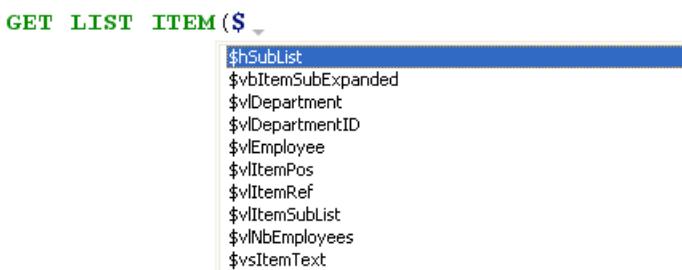


```
// Create a new empty
hList:=New list
// Select all the rec
```

Si vous désélectionnez cette option, seule la flèche jaune est affichée.

Suggestions

Cette zone vous permet de configurer les mécanismes d'autocomplétion de l'éditeur de méthodes afin de l'adapter à vos propres habitudes de travail.

	Description
Affichage automatique de la fenêtre	<p>Ouverture automatique de la fenêtre de suggestion pour :</p> <ul style="list-style-type: none"> ● Constantes ● Variables (locales et interprocess) et attributs d'objets ● Tables ● Prototypes (c'est-à-dire les fonctions de classe) <p>Par exemple, lorsque l'option "Variables (locales ou interprocess) et attributs d'objets" est cochée, une liste de suggestions apparaît lorsque vous tapez le caractère \$:</p>  <p>The screenshot shows a code editor with the text "GET LIST ITEM(\$" typed. A dropdown menu is open, listing various variables and objects starting with "\$". The item "\$hSubList" is highlighted in blue.</p> <p>Vous pouvez désactiver ce fonctionnement pour certains éléments du langage en décochant leur option correspondante.</p>
Validation d'une suggestion	<p>Définit le contexte de saisie qui permet à l'éditeur de méthodiser automatiquement la suggestion courante affichée dans la fenêtre d'autocomplétion.</p> <ul style="list-style-type: none"> ● Tabulation et délimiteurs <p>Lorsque cette option est cochée, vous pouvez valider la sélection courante à l'aide de la touche Tab ou de tout autre délimiteur pertinent. Par exemple, si vous entrez "ALE" et ensuite "(", 4D écrit automatiquement "ALERT(" dans l'éditeur. Voici la liste des délimiteurs qui sont pris en compte :</p> <p>(; : = < [{</p> ● Tabulation uniquement <p>Lorsque cette option est cochée, vous pouvez utiliser uniquement la touche Tab pour insérer la suggestion courante. Ceci peut être utilisé plus particulièrement pour faciliter la saisie de caractères délimiteurs dans les noms d'éléments, tels que \${1}.</p> <p>Note : Vous pouvez également double-cliquer sur la fenêtre ou appuyer sur la touche Retour chariot pour valider une suggestion.</p>

Page Raccourcis

Cette page affiche la liste de tous les raccourcis-clavier utilisés dans le mode Développement de 4D (hormis les raccourcis "système" standard, tels que Ctrl+c/Commande+c pour la commande Copier).

The screenshot shows the '4D Preferences' dialog box with the 'Shortcuts' tab selected. The main area is a table listing various actions and their keyboard shortcuts. The columns are labeled 'Actions', 'Cmd', 'Shift', 'Alt', 'Ctrl', and 'Letter'. The 'Letter' column shows the corresponding key or key combination. Buttons at the bottom include 'Reset to factory settings', 'Cancel', and 'OK'.

Actions	Cmd	Shift	Alt	Ctrl	Letter
Query > Query	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Y
Order By	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Y
Close Project	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	W
Close Window	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	W
Close All Windows	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	W
Start Of Block	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	up arrow
Move Lines Up	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	up arrow
Show Current Table	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	U
List of Tables	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	U
4D Server Administration	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	U
Show Next Tab	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Tab
Show Previous Tab	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Tab
Tool Box > Tool Box	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	T
Swap Expression	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	T
Save Window	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	S
Save All	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	S
Flush Data Buffers	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	S
Method	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	R
Apply Formula	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	R
Quit	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Q
Quit and Keep Windows	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Q
Page Setup	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	P
Print	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	P

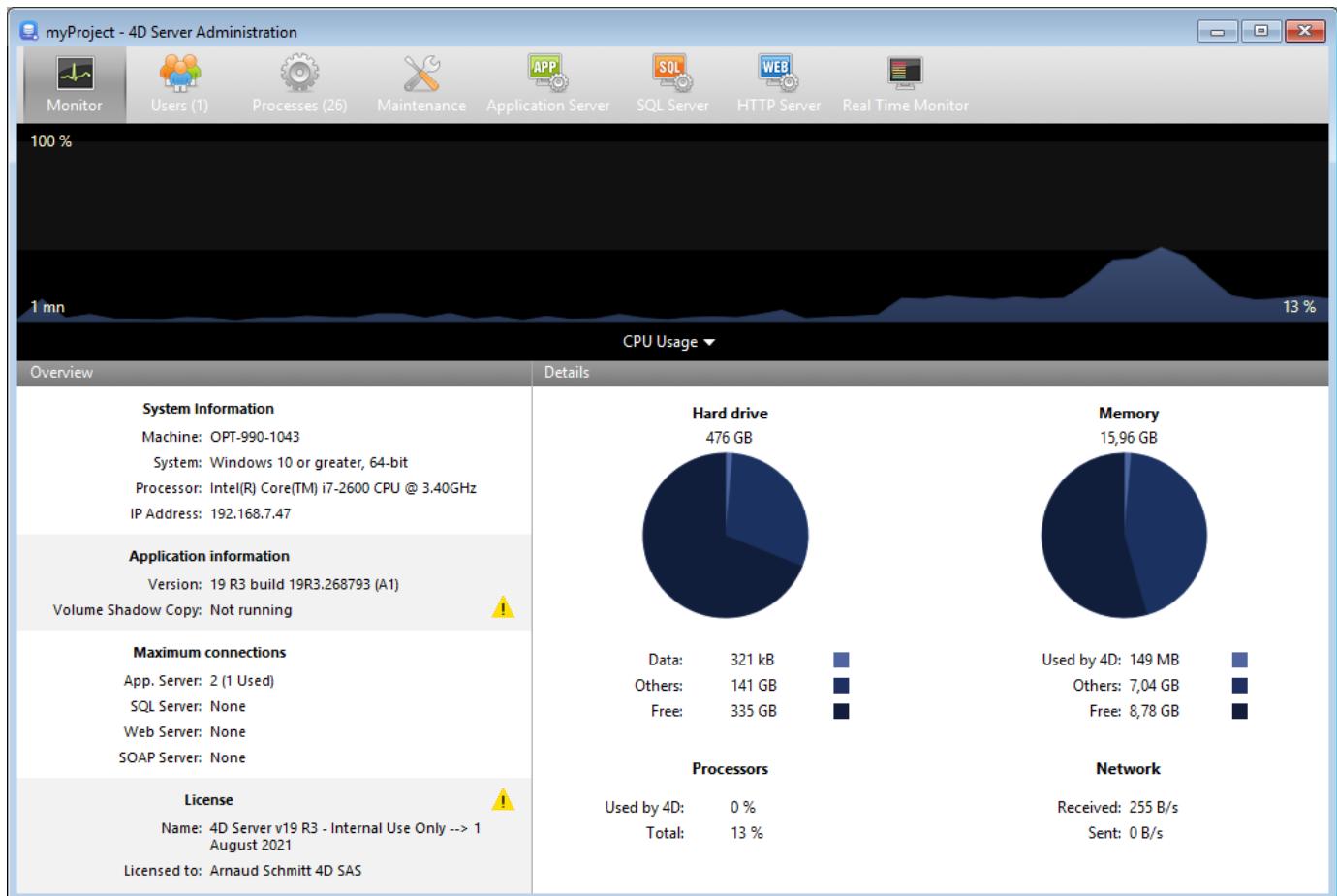
Pour modifier un raccourci, il suffit de sélectionner/désélectionner l'élément à modifier (Majuscule, Option ou touche du clavier) dans la liste des raccourcis. Vous pouvez également double-cliquer sur la ligne d'un raccourci afin de le configurer dans la boîte de dialogue.

A noter que chaque raccourci clavier inclut implicitement la touche Ctrl (Windows) ou Commande (macOS).

Si vous modifiez cette liste, vos paramètres de raccourcis personnalisés sont stockés dans un fichier *4DShortcutsvXX.xml*, créé au même niveau que [le fichier de préférences utilisateur](#). Ainsi, à chaque mise à jour de 4D, vous conservez vos préférences de raccourcis clavier.

Page Moniteur

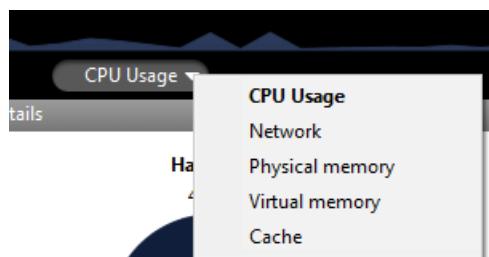
La Page Moniteur affiche des informations dynamiques relatives à l'exploitation de la base de données ainsi que des informations sur le système et l'application 4D Server.



Sous Windows, certaines informations système affichées sur cette page sont récupérées via les outils "Analyseur de performance" de Windows. Ces outils sont accessibles uniquement si l'utilisateur ayant ouvert la session sur laquelle 4D Server a été lancé dispose des autorisations d'administration nécessaires.

Zone graphique

La zone graphique permet de visualiser l'évolution en temps réel de plusieurs paramètres : le taux d'utilisation des processeurs, le trafic réseau et l'état de la mémoire. Vous sélectionnez le paramètre à afficher via le menu situé au centre la fenêtre :



- Utilisation processeurs : Taux d'utilisation globale du ou des processeur(s) de la machine, toutes applications confondues. La part spécifique de 4D Server dans ce taux d'utilisation est fournie dans la zone d'informations "Processseurs".
- Réseau : Nombre d'octets reçus par seconde par la machine (serveur ou client). Le nombre d'octets envoyés est fourni dans la zone d'informations "Réseau".

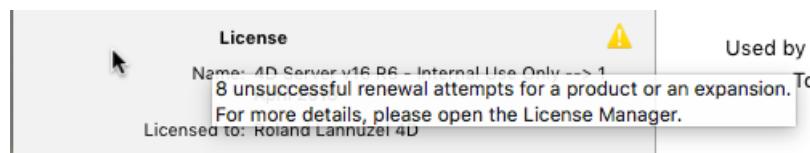
- Mémoire physique : Quantité de mémoire RAM de la machine utilisée par 4D Server. Une vue plus détaillée de l'utilisation de la mémoire est fournie dans la zone d'informations "Mémoire".
- Mémoire virtuelle : Affiche dans la zone graphique la quantité de mémoire virtuelle utilisée par l'application 4D Server. Cette mémoire est allouée par le système en fonction des besoins de l'application. La valeur située en bas à droite de la zone indique la quantité de mémoire en cours d'utilisation. La valeur située en haut à gauche indique la quantité maximale de mémoire virtuelle utilisable. La valeur maximale est calculée dynamiquement en fonction des paramètres mémoire généraux de l'application.
- Cache : Affiche dans la zone graphique la quantité de mémoire cache utilisée par l'application 4D Server. La valeur située en bas à droite de la zone indique la quantité de mémoire en cours d'utilisation. La valeur située en bas à droite de la zone indique la quantité de mémoire en cours d'utilisation.

A noter que lorsque cette option est sélectionnée, le défilement de la zone graphique est ralenti car une analyse efficace du cache s'effectue généralement sur une période d'observation assez longue.

Zone Vue d'ensemble

La zone "Vue d'ensemble" fournit diverses informations relatives au système, à l'application et aux licences installées sur la machine de 4D Server.

- Informations système : Ordinateur, système et adresse IP du serveur
- Informations application : Numéro de version interne de 4D Server et statut du Volume Shadow Copy
- Connexions maximum : Nombre de connexions simultanées autorisées par type de serveur
- Licence : Description de la licence. Lorsque la licence produit ou l'une de ses expansions expire dans moins de 10 jours, dans le cas d'un abonnement, 4D Server tente de renouveler automatiquement la licence depuis le compte de l'utilisateur 4D. Dans ce cas, si le renouvellement automatique échoue pour une raison quelconque (erreur de connexion, statut du compte invalide, contrat non prolongé...), une icône d'avertissement est affichée à côté de la licence afin d'alerter l'administrateur du serveur. Des informations supplémentaires relatives au statut du renouvellement de la licence peuvent être affichées dans une info-bulle lorsque vous survolez la zone avec la souris :



Généralement, vous devrez vérifier le [Gestionnaire de licences](#).

Zone Détails

La zone "Détails" reprend une partie des informations affichées dans la zone graphique et propose des informations complémentaires.

- Disque dur : Capacité globale du disque dur et répartition entre l'espace occupé par les données de la base (fichier de données + index des données), l'espace occupé par les autres fichiers et l'espace disponible.
- Mémoire : Mémoire RAM installée sur la machine et quantité de mémoire occupée par 4D Server, par les autres applications ainsi que mémoire disponible. La mémoire occupée par 4D Server peut également être affichée dynamiquement dans la zone graphique.
- Processeurs : Taux instantané d'occupation du ou des processeur(s) de la machine par 4D Server et par les autres applications. Ce taux est recalculé en permanence. Le taux d'occupation par 4D Server peut également être affiché dynamiquement dans la zone graphique.
- Réseau : Nombre instantané d'octets envoyés et reçus par la machine (serveur ou client). Cette valeur est réactualisée en permanence. Le nombre d'octets reçus peut également être affiché dynamiquement dans la zone graphique.

Page Utilisateurs

La page Utilisateurs liste les utilisateurs 4D connectés au serveur.

4D User	Machine name	Session name	IP Address	Login date	CPU Time	Activity
Designer	WIN7-ESMITH	esmith	localhost	03/05/2016 17:24	00:00:02	2%
Designer	MACWIN7	Arnaud	192.168.10.11	03/05/2016 17:27	00:00:01	0%

Le bouton "Utilisateurs" indique entre parenthèses le nombre total d'utilisateurs connectés au serveur (ce nombre ne tient pas compte des éventuels filtres d'affichage appliqués à la fenêtre). La page contient également une zone de recherche dynamique et des boutons de commande. Vous pouvez modifier l'ordre des colonnes par simple glisser-déposer de la zone d'en-tête des colonnes.

Vous pouvez également trier la liste sur les valeurs d'une colonne en cliquant sur son en-tête. Cliquez plusieurs fois pour définir alternativement un ordre croissant/décroissant.



Liste des utilisateurs

Pour chaque utilisateur connecté à la base, la liste fournit les informations suivantes :

- Système de la machine cliente (macOS ou Windows) sous forme d'icône.
- Utilisateur 4D : Nom d'utilisateur 4D ou alias s'il est défini avec la commande `SET USER ALIAS` sur le poste de l'utilisateur. Si les mots de passe ne sont pas activés et si aucun alias n'a été défini, tous les utilisateurs 4D sont nommés "Super_Utilisateur".
- Nom de machine : Nom de la machine distante.
- Nom de session : Nom de la session ouverte sur la machine distante.
- Adresse IP : Adresse IP de la machine distante.
- Connexion : Date et heure de la connexion de la machine distante.
- Temps CPU : Temps processeur consommé par cet utilisateur depuis la connexion.
- Activité : Ratio du temps que 4D Server consacre à cet utilisateur (affichage dynamique). "Endormi" si la machine du poste client est passée en veille (cf. ci-dessous).

Gestion des utilisateurs endormis

4D Server gère spécifiquement le cas où la machine d'une application distante 4D passe en mode veille alors que la connexion au serveur est toujours active. Dans ce cas, l'application distante 4D connectée notifie automatiquement 4D Server de sa déconnexion imminente. Sur le serveur, l'utilisateur connecté prend le statut d'activité Endormi :

4D User	Machine name	Session name	IP Address	Login date	CPU Time	Activity
Designer	macmini-program	program	192.168.18.11	20/10/15 09:43	00:00:18	Sleeping
Designer	iMac-VTalbot-0833	Vanessa Talbot	localhost	20/10/15 08:40	00:02:16	0%

Ce statut libère les ressources sur le serveur. En outre, l'application 4D distante se reconnecte automatiquement à 4D Server après la sortie du mode veille.

Le scénario suivant est pris en charge : un utilisateur distant cesse de travailler durant un certain laps de temps, par exemple durant la pause déjeuner, mais garde ouverte la connexion au serveur. La machine passe en mode veille. Au retour de l'utilisateur, la machine sort du mode veille et l'application 4D distante récupère automatiquement sa connexion au serveur ainsi que son contexte de session.

Une session distante en veille est automatiquement abandonnée par le serveur après 48 heures d'inactivité. Vous pouvez modifier ce timeout par défaut à l'aide de la commande [SET DATABASE PARAMETER](#) avec le sélecteur `Remote connection sleep timeout`.

Zone de recherche/filtrage

Cette fonction permet de réduire le nombre de lignes affichées dans la liste à celles qui correspondent au texte saisi dans la zone de recherche. La zone indique les colonnes dans lesquelles la recherche/le filtrage sera effectué(e). Dans la page Utilisateurs, il s'agit des colonnes Utilisateur 4D, Nom de machine et Nom de session.

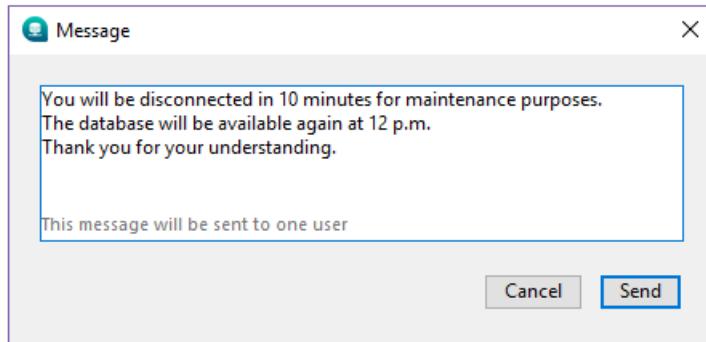
La mise à jour de la liste est effectuée en temps réel à mesure que vous saisissez du texte dans la zone. Il est possible de saisir plus d'une valeur à chercher. Utilisez un point-virgule pour séparer les valeurs. L'opérateur utilisé dans ce cas est du type `OU`. Par exemple, si vous saisissez "Jean;Marie;Pierre", seules les lignes comportant Jean OU Marie OU Pierre dans l'une des colonnes cibles seront conservées.

Boutons d'administration

La page comporte trois boutons de commande. Ces boutons sont actifs si au moins une ligne est sélectionnée. Vous pouvez sélectionner plusieurs lignes en appuyant sur la touche Maj pour une sélection continue ou Ctrl (Windows) / Commande (macOS) pour une sélection discontinue.

Envoyer message

Ce bouton permet d'envoyer un message aux utilisateurs 4D sélectionnés dans la fenêtre. Si aucun utilisateur 4D n'est sélectionné, le bouton est désactivé. Lorsque vous cliquez sur le bouton, une boîte de dialogue apparaît, vous permettant de saisir le message. La boîte de dialogue indique le nombre d'utilisateurs qui recevront le message :



Le message sera affiché sous forme d'alerte sur les postes distants.

Vous pouvez effectuer la même opération pour les utilisateurs distants à l'aide de la commande [SEND MESSAGE TO REMOTE USER](#).

Visualiser process

Ce bouton permet de visualiser directement les process du ou des utilisateur(s) sélectionné(s) dans la [page Process](#) de la fenêtre d'administration. Lorsque vous cliquez sur ce bouton, 4D Server bascule sur la page Process et pré-remplit la zone de recherche/filtrage de cette page avec les noms des utilisateurs sélectionnés.

Déconnecter

Ce bouton permet de forcer la déconnexion du ou des utilisateur(s) sélectionné(s). Lorsque vous cliquez sur ce bouton, une boîte de dialogue d'alerte apparaît, vous permettant de confirmer ou d'annuler l'opération (appuyez sur la touche Alt avant de cliquer sur le bouton Déconnecter pour déconnecter directement les utilisateurs sélectionnés sans afficher la boîte de dialogue de confirmation).

Vous pouvez effectuer la même opération pour les utilisateurs distants à l'aide de la commande [DROP REMOTE USER](#).

Page Process

La page Process liste les process en cours d'exécution.

Process name		Session / Info	Type	Num v	State	CPU Time	Activity
DB4D CRRIV	-	DB4D SERVER	Running	0	00:00:00	0 %	
DB4D Flush	-	DB4D Server	0	Running	00:00:00	0 %	
DB4D Index builder	-	DB4D Server	0	Running	00:00:00	0 %	
DB4D Server	-	DB4D Server	0	Running	00:00:00	0 %	
DB4D Sockets	-	DB4D Server	0	Running	00:00:00	0 %	
Garbage Handler	-	DB4D Server	0	Running	00:00:00	0 %	
HTTP Listener	-	Web Server	0	Running	00:00:00	0 %	
Internal Timer Process	-	Application server	2	Executing	00:30:04	0 %	
ServerNet select I/O handler	-	Other process type	0	Running	00:00:01	0 %	
ServerNet select I/O handler	-	Other process type	0	Running	00:00:01	0 %	
Task managers	-	SQL Server	0	Running	00:00:00	0 %	
TCP connection listener	-	TCP Connection listener	0	Running	00:00:00	0 %	
TCP connection listener	-	SQL Server	0	Running	00:00:00	0 %	
User Interface	-	Application server	1	Waiting for event	01:09:41	9 %	
Application process	aschmitt	Application server	9	Waiting for event	00:00:14	0 %	
Application process	aschmitt	4D Client Database process	12	Executing	00:00:01	0 %	
Design process	aschmitt	Application server	11	Waiting for event	00:00:14	0 %	
Method9	Launched by aschmitt	Stored procedure	10	Executing	00:00:13	87 %	

Le bouton "Process" indique entre parenthèses le nombre total de process en cours d'exécution sur le serveur (ce nombre ne tient pas compte des éventuels filtres d'affichage appliqués à la fenêtre ni de l'état de l'option Afficher les process par groupes).

Vous pouvez modifier l'ordre des colonnes par simple glisser-déposer de la zone d'en-tête des colonnes. Vous pouvez également trier la liste sur les valeurs d'une colonne en cliquant sur son en-tête.

Tout comme la Page Utilisateurs, cette page contient une [zone de recherche/filtrage](#) dynamique, permettant de réduire le nombre de lignes affichées dans la liste à celles qui correspondent au texte saisi dans la zone de recherche. La recherche/le filtrage est effectué(e) dans les colonnes Session et Nom de process.

Vous disposez également de trois boutons-raccourcis permettant de filtrer par famille les process affichés dans la fenêtre :



- Process utilisateurs : Process générés par et pour les sessions utilisateurs. Ces process sont précédés d'une icône en forme de personnage.
- Process 4D : Process générés par le moteur de 4D Server. Ces process sont précédés d'une icône en forme de roue crantée.
- Process en attente : Process inactifs mais conservés temporairement et pouvant être réutilisés à tout moment. Ce mécanisme permet d'optimiser la réactivité de 4D Server. Ces process sont précédés d'une icône grisée en forme de personnage.

L'option Afficher les process par groupes vous permet de regrouper les process internes de 4D Server ainsi que les process clients, pour plus de lisibilité. Lorsque vous cochez cette option :

- les process clients 4D "jumeaux" (Process client 4D principal et Process base 4D client, cf. paragraphe [Type du](#)

process) sont groupés en un seul,

- le groupe "Gestionnaires de tâches" est créé ; il inclut les process internes dédiés à la répartition des tâches (Shared balancer, Net session manager, Exclusive pool worker),
- le groupe "Gestionnaires clients" est créé ; il inclut les différents process internes clients.

La zone inférieure de la fenêtre permet d'afficher la représentation graphique de l'activité du ou des process sélectionné(s).

Vous pouvez sélectionner plusieurs lignes en appuyant sur la touche Maj pour une sélection continue ou Ctrl (Windows) / Commande (macOS) pour une sélection discontinue.

L'activité du process est le pourcentage du temps que 4D Server a consacré à ce process (ratio). La fenêtre fournit les informations suivantes pour chaque process :

- Type de process (cf. ci-dessous),
- Session:
 - Process 4D - vide,
 - Process utilisateur - nom de l'utilisateur 4D,
 - Process web - chemin URL,
- Nom du process,
- Numéro du process (tel que retourné par la fonction [Nouveau process](#) par exemple). Le numéro du process est le numéro attribué sur le serveur. Dans le cas d'un process global, ce numéro peut être différent de celui attribué sur le poste client.
- Etat courant du process,
- Temps (en secondes) d'exécution du process depuis sa création,
- Pourcentage du temps que 4D Server a consacré à ce process (ratio).

Type du process

Chaque process est identifié par une icône et un type. La couleur et la forme de l'icône indiquent la famille du process :

icon	type
	Serveur d'application
	Serveur SQL
	Serveur DB4D (moteur de base de données)
	Serveur Web
	Serveur SOAP
	Process client 4D protégé (process développement d'un 4D connecté)
	Process base 4D client (process parallèle à un process 4D client. Process préemptif chargé de contrôler le process client 4D principal correspondant)
	Process client 4D principal (process principal d'un 4D connecté. Process préemptif chargé de contrôler le process client 4D principal correspondant)
	Process en attente (ancien ou futur "Process client 4D base de données")
	Worker serveur SQL
	Worker serveur HTTP
	Process 4D client (process coopératif tournant sur le 4D connecté)
	Procédure stockée (process coopératif lancé par un 4D connecté et tournant sur le serveur)
	Méthode Web (lancée par un 4DACTION par exemple)
	Méthode Web (process préemptif)
	Méthode SOAP (lancée par un Web Service)
	Méthode SOAP (process préemptif)
	Logger
	Listener connexion TCP
	Manager session TCP
	Autre process
	Process worker (coopératif)
	Process 4D client (préemptif)
	Procédure stockée (process préemptif)
	Process worker (préemptif)

Chaque process client 4D principal et son process base 4D client "jumeau" sont regroupés lorsque l'option Afficher les process par groupes est cochée.

Boutons d'administration

La page comporte cinq boutons de commande permettant d'agir sur le ou les process sélectionné(s). A noter que vous ne pouvez agir que sur les process utilisateurs.



- Tuer process : permet de tuer le ou les process sélectionné(s). Lorsque vous cliquez sur ce bouton, une boîte de dialogue d'alerte apparaît, vous permettant de confirmer ou d'annuler l'opération.

Vous pouvez tuer directement les process sélectionnés sans afficher la boîte de dialogue de confirmation, en

appuyant sur la touche Alt avant de cliquer sur le bouton, ou en utilisant la commande **ABORT PROCESS BY ID**.

- Endormir process : permet d'endormir le ou les process sélectionné(s).
- Réactiver process : permet de réactiver le ou les process sélectionné(s). Les process doivent avoir été auparavant endormis (via le bouton précédent ou par programmation) sinon le bouton est sans effet.
- Tracer process : permet d'ouvrir sur le poste serveur une ou plusieurs fenêtre(s) du débogueur pour le ou les process sélectionné(s). Lorsque vous cliquez sur ce bouton, une boîte de dialogue d'alerte apparaît, vous permettant de confirmer ou d'annuler l'opération. A noter que la fenêtre du débogueur ne s'affiche que lorsque du code 4D est effectivement exécuté sur le poste serveur (par exemple dans le cadre d'un trigger ou de l'exécution d'une méthode ayant l'attribut "Exécuter sur serveur").

Vous pouvez déboguer un process directement, sans afficher la boîte de dialogue de confirmation : pour cela, appuyez sur la touche Alt avant de cliquer sur le bouton.

- Voir utilisateurs : permet d'afficher dans la [page Utilisateurs](#) tous les process du ou des utilisateur(s) sélectionné(s). Le bouton est actif lorsqu'un process utilisateur au moins est sélectionné.

Page maintenance

La page Maintenance fournit diverses informations relatives au fonctionnement courant de la base. Elle donne également accès aux fonctions de maintenance élémentaires :

The screenshot shows the 4D Server Administration interface with the title bar "myProject - 4D Server Administration". The "Maintenance" tab is selected. The main content area contains several sections:

- Last verification:** Unknown Date. Includes a "Verify Records and Indexes" button and a note: "Verification can help you detect performance problems or problems concerning data and/or index validity."
- Last compacting:** Unknown Date. Includes a "Compact Data..." button and a note: "Compacting your database reduces the space taken up by the data and optimizes their organization. Compacting should be used when you notice a decrease in performance or when you want to reduce the weight of your data. Compacting requires the server to be restarted. All users will be disconnected."
- Uptime:** 13 minutes. Includes a "Restart server..." button and a note: "Restarting the server will disconnect all the users."
- Last backup:** 30/07/2019 at 15:36. Includes a "Start backup" button and a note: "The server will not be restarted but the users will be blocked during the operation."
- Request and Debug Logs:** 0 second logged. Includes buttons for "Stop Request and Debug Logs", "View Report", "Load logs configuration file", and "Pause Logging". A note states: "The server performance might be altered slightly during the generation of log files."

Dernière vérification/dernier compactage

Ces zones indiquent la date, l'heure et le statut de la dernière [vérification des données](#) et de la dernière [opération de compactage](#) effectuées sur la base.

Vérifier enregistrements et index

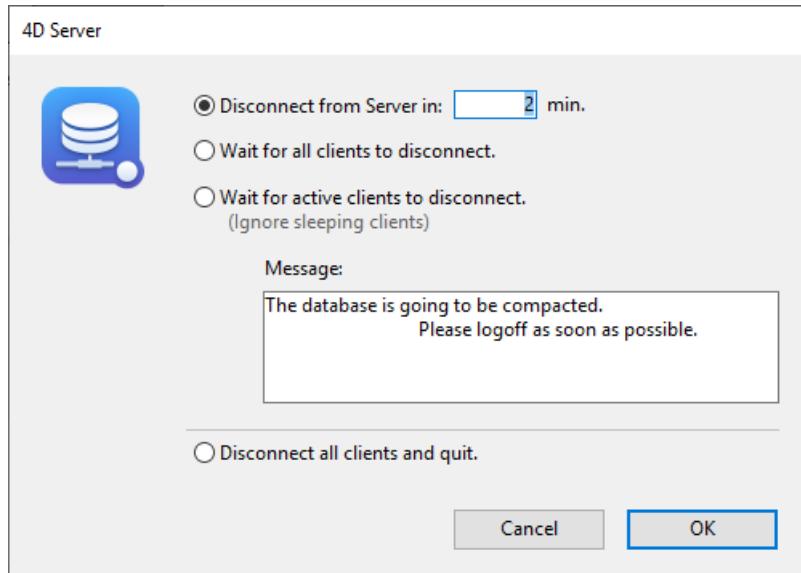
Ce bouton permet de lancer directement l'opération de vérification, sans interruption du serveur. A noter que le serveur peut être sensiblement ralenti durant l'opération.

Tous les enregistrements et tous les index de la base sont vérifiés. Si vous souhaitez pouvoir cibler la vérification ou disposer d'options supplémentaires, vous devez utiliser le [Centre de sécurité et de maintenance \(CSM\)](#).

A l'issue de la vérification, un fichier de compte-rendu est généré au format XML sur le serveur, dans le dossier [maintenance Logs](#). Le bouton [Voir le compte rendu](#) (nommé Télécharger le compte rendu si l'opération a été effectuée depuis un poste distant) vous permet d'afficher le fichier dans votre navigateur.

Compackter les données...

Ce bouton permet de lancer directement une opération de compactage des données. Cette opération nécessite de stopper le serveur : lorsque vous cliquez sur le bouton, la boîte de dialogue de fermeture de la base 4D Server apparaît, vous permettant de choisir le mode d'interruption de l'exploitation :



Après l'interruption effective de la base, 4D Server effectue un compactage standard des données de la base. Si vous souhaitez disposer d'options supplémentaires, vous devez utiliser le [Centre de sécurité et de maintenance \(CSM\)](#).

Une fois le compactage terminé, 4D Server relance automatiquement l'application. Les utilisateurs 4D peuvent alors se reconnecter.

Si la demande de compactage a été effectuée depuis un client 4D distant, ce poste est automatiquement reconnecté par 4D Server.

A l'issue de la vérification, un fichier de compte-rendu est généré au format XML sur le serveur, dans le dossier [maintenance Logs](#). Le bouton [Voir le compte rendu](#) (nommé Télécharger le compte rendu si l'opération a été effectuée depuis un poste distant) vous permet d'afficher le fichier dans votre navigateur.

Durée de fonctionnement

Cette zone indique la durée de fonctionnement du serveur depuis son dernier démarrage (jours, heures et minutes).

Redémarrer le serveur...

Ce bouton vous permet de provoquer un redémarrage immédiat du serveur. Lorsque vous cliquez sur ce bouton, la boîte de dialogue de fermeture de la base 4D Server apparaît, vous permettant de choisir le mode d'interruption de l'exploitation. Après le redémarrage, 4D Server relance automatiquement la base. Les utilisateurs 4D peuvent alors se reconnecter.

Si la demande de redémarrage a été effectuée depuis un client 4D distant, ce poste est automatiquement reconnecté par 4D Server.

Dernière sauvegarde

Cette zone indique la date et l'heure de la [dernière sauvegarde](#) de la base et fournit des informations relatives à la prochaine sauvegarde automatique, le cas échéant. Les sauvegardes automatiques sont paramétrées via la page [Périodicité des propriétés de la base](#).

- Dernière sauvegarde : date et heure de la dernière sauvegarde automatique.
- Prochaine sauvegarde : date et heure de la prochaine sauvegarde.
- Place nécessaire estimée : estimation de la taille nécessaire pour la sauvegarde. La véritable taille réelle du fichier de sauvegarde pourra varier en fonction des paramétrages (compression...) et des variations du fichier de données.
- Place disponible : place disponible sur le volume de sauvegarde.

Le bouton [Sauvegarder la base](#) permet de déclencher une sauvegarde immédiate de la base en utilisant les paramètres de sauvegarde courants (fichiers sauvegardés, emplacement des archives, options, etc.). Vous pouvez visualiser ces

paramètres en cliquant sur le bouton Propriétés.... Pendant une sauvegarde sur le serveur, les postes clients sont "bloqués" (mais pas déconnectés) et il n'est pas possible à de nouveaux clients de se connecter.

Requêtes et débogage

Cette zone indique la durée d'enregistrement des fichiers d'historique (lorsqu'ils sont activés) et vous permet de gérer leur activation.

Consultez la section [Description des fichiers d'historique](#) pour obtenir plus de détails sur les fichiers d'historique.

Start/Stop Request and Debug Logs

Le bouton Démarrer les journaux des requêtes et de débogage permet de démarrer les fichiers de compte-rendu. Ce mode pouvant dégrader sensiblement les performances du serveur, il est à réserver à la phase de mise au point de l'application.

Ce bouton enregistre uniquement les opérations exécutées sur le serveur.

Une fois l'enregistrement des requêtes activé, le libellé du bouton devient Arrêter les journaux des requêtes et de débogage, permettant de stopper l'enregistrement des requêtes à tout moment. A noter qu'une reprise de l'enregistrement après un arrêt "écrase" le fichier précédent.

Voir le compte rendu

Le bouton Voir le compte rendu (nommé Télécharger le compte rendu si l'opération a été effectuée depuis un client distant) permet d'ouvrir une fenêtre système affichant le fichier journal.

Load logs configuration file

Ce bouton vous permet de charger un [fichier de configuration de log](#) (fichier `.json`) pour un serveur spécifique. Such a file can be provided by 4D technical services to monitor and study specific cases.

Pause logging

This button suspends all currently logging operations started on the server. This feature can be useful to temporarily lighten the server tasks.

When the logs have been paused, the button title changes to Resume logging, so that you can resume the logging operations.

You can pause and resume logging using the [SET DATABASE PARAMETER](#) command.

Page Serveur d'application

La Page Serveur d'application regroupe les informations relatives à la base de données publiée par 4D Server et permet de gérer cette publication.

The screenshot shows the 'myProject - 4D Server Administration' window. The top navigation bar includes icons for Monitor, Users (1), Processes (26), Maintenance, Application Server (selected), SQL Server, HTTP Server, and Real Time Monitor. The main content area displays the following information:

- Application Server**: State: Started, Starting time: 27/07/2021 at 18:29, Uptime: 1 day 15 hours 59 minutes, with a 'Reject new connections' button.
- Configuration**: Structure file: "myProject.4DProject" in volume "D:", Data file: "data.4DD" in volume "D:", Log file: data.journal, Mode: Interpreted.
- Launch**: Launched as service: No, Listening to IP: 192.168.7.47, Port: 19813, TLS enabled: Yes.
- Memory**: Used cache memory: 733,88 kB, Total cache memory: 400 MB.
- Application Server Connections**: Maximum: 2, Used: 1.

La partie supérieure de la page fournit des informations sur le statut courant du serveur d'application de 4D Server.

- Etat : Démarré ou Arrêté.
- Date de démarrage : Date et heure de lancement de la base serveur. Cette date correspond à l'ouverture de la base par 4D Server.
- Durée de fonctionnement : Durée écoulée depuis la dernière ouverture de la base.

Refuser / Accepter nouvelles connexions

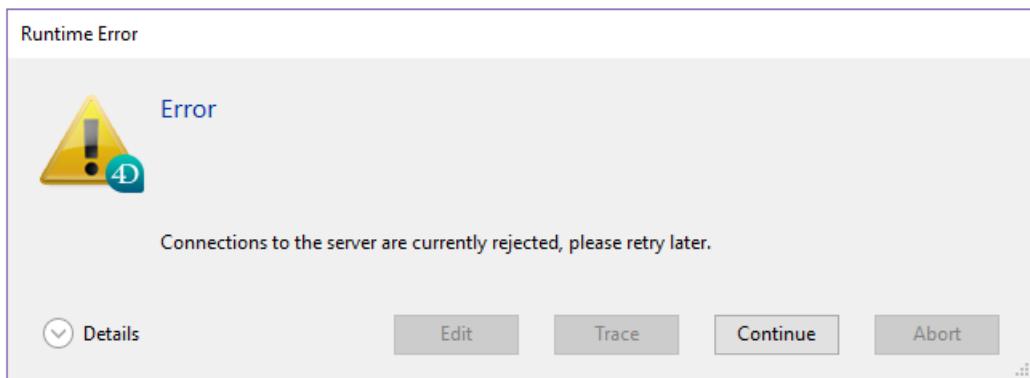
Il permet de gérer l'accès de nouveaux postes clients à l'application serveur.

Par défaut, lorsque la base est publiée :

- Le libellé du bouton est "Refuser nouvelles connexions"
- De nouveaux clients peuvent se connecter librement (dans les limites des connexions accordées par la licence).
- Le nom de la base est publié dans la boîte de dialogue de connexion (si l'option "Publier le nom de la base au démarrage dans le dialogue de connexion" est cochée dans les Préférences).

Si vous cliquez sur le bouton Refuser nouvelles connexions :

- Le libellé du bouton devient "Accepter nouvelles connexions"
- Plus aucun nouveau client ne peut alors se connecter. Les clients qui tenteront de se connecter recevront le message suivant :



- Le nom du projet n'apparaît plus dans la boîte de dialogue de connexion.
- Les clients déjà connectés ne sont pas déconnectés et peuvent continuer à travailler normalement.

Vous pouvez effectuer la même opération à l'aide de la commande [REJETER NOUVELLES CONNEXIONS DISTANTES](#).

- Si vous cliquez sur le bouton Accepter nouvelles connexions, la base retourne dans l'état "par défaut".

Cette fonction permet par exemple à un administrateur, juste après avoir démarré le serveur, d'effectuer diverses opérations de maintenance (vérification, compactage...). S'il utilise une connexion cliente, il a la certitude d'être le seul à modifier les données. Il est également possible d'utiliser cette fonction en préparation d'une opération de maintenance nécessitant qu'aucun poste client ne soit connecté.

Information

Configuration

Cette zone fournit plusieurs informations sur la base de données 4D publiée par le serveur : nom et emplacement des fichiers de structure et de données et nom du fichier journal (fichier d'historique de la base). Vous pouvez cliquer sur le nom du fichier de structure ou de données afin de visualiser son chemin d'accès complet.

Le champ Mode indique le mode d'exécution courant de l'application : compilé ou interprété.

La partie inférieure de la zone indique les paramètres de configuration du serveur (démarré comme service, port et adresse IP) et l'activation du TLS pour les connexions client-serveur (ne concerne pas les connexions SQL ni HTTP).

Mémoire

Cette zone indique la mémoire cache totale (paramètre défini dans les préférences de la base) et la mémoire cache utilisée (allocation dynamique par 4D Server en fonction des besoins).

Connexions application serveur

- Maximum : indique le nombre maximum de connexions clientes simultanées autorisées pour le serveur d'application. Cette valeur dépend de la licence installée sur le poste serveur.
- Utilisées : indique le nombre de connexions actuellement consommées.

Page Serveur SQL

La page SQL Server regroupe les informations relatives au serveur SQL intégré de 4D Server. Elle comporte également un bouton permettant de contrôler l'activation du serveur.

The screenshot shows the 4D Server Administration window titled "myProject - 4D Server Administration". The top navigation bar includes links for Monitor, Users (1), Processes (26), Maintenance, Application Server, SQL Server (selected), WEB, HTTP Server, and Real Time Monitor. The SQL Server section displays the following information:

- State:** Started
- Starting time:** 29/07/2021 at 10:30
- Uptime:** Less than one minute
- Stop SQL Server** button (disabled)

Configuration details:

- Auto-launched at startup: No
- Listening to IP: 192.168.7.47
- Listening on port: 19812
- TLS enabled: No

Connections status: Number of connections: 0

Maximum connections setting: SQL Server: None

La partie supérieure de la page fournit des informations sur le statut courant du serveur SQL de 4D Server.

- Etat : Démarré ou Arrêté
- Date de démarrage : Date et heure du dernier lancement du serveur SQL.
- Durée de fonctionnement : Délai écoulé depuis le dernier démarrage du serveur SQL.

Démarrer / Arrêter le serveur SQL

Ce bouton fonctionne en bascule. Il permet de contrôler l'activation du serveur SQL de 4D Server.

- Lorsque l'état du serveur SQL est "Démarré", le bouton est libellé Arrêter le serveur SQL. Si vous cliquez sur ce bouton, le serveur SQL de 4D Server est immédiatement stoppé, il ne répond plus aux requêtes SQL externes reçues sur le port TCP désigné.
- Lorsque l'état du serveur SQL est "Arrêté", le bouton est libellé Démarrer le serveur SQL. Si vous cliquez sur ce bouton, le serveur SQL de 4D Server est immédiatement démarré, il répond aux requêtes SQL externes reçues sur le port TCP désigné. A noter que vous devez disposer d'une licence adéquate pour pouvoir exploiter le serveur SQL de 4D.

Le serveur SQL peut également être lancé automatiquement au démarrage de l'application (option des Préférences) ou par programmation.

Information

Configuration

Cette zone fournit plusieurs informations sur les paramètres de configuration du serveur SQL : lancement automatique au démarrage, adresse IP d'écoute, port TCP (19812 par défaut) et activation du SSL pour les connexions SQL (ne concerne pas les connexions 4D ni HTTP).

Ces paramètres peuvent être modifiés via les Préférences de 4D.

Connexions

Nombre de connexions SQL actuellement ouvertes sur 4D Server.

Connexions maximum

Nombre maximum de connexions SQL simultanées autorisées. Cette valeur dépend de la licence installée sur le poste serveur.

Page Serveur HTTP

La page HTTP Server regroupe les informations relatives aux opérations du serveur Web et du serveur SOAP de 4D Server. Le serveur Web permet de publier du contenu Web tel que des pages HTML ou des images à destination de navigateurs Web et gérer des requêtes REST. Le serveur SOAP gère la publication de Web Services. Ces deux serveurs s'appuient sur le serveur HTTP interne de 4D Server.

The screenshot shows the 'myProject - 4D Server Administration' window. The top navigation bar includes icons for Monitor, Users (1), Processes (26), Maintenance, Application Server, SQL Server, HTTP Server (selected), and Real Time Monitor. The main content area is titled 'HTTP Server' and displays the following information:

- State:** Started
- Starting time:** 27/07/2021 at 18:29
- Uptime:** 1 day 16 hours 3 minutes
- Total HTTP hits:** 0
- Stop HTTP server** button

Web information: Web requests: Accepted, Maximum Connections: None

SOAP information: SOAP requests: Accepted, Maximum Connections: None

HTTP server Configuration:

- Auto-launched at startup: Yes
- HTTP Server processes (used/total): 5/7
- Cache memory: 0 pages (0%)
- Listening to IP: 192.168.7.47
- HTTP port: 80
- TLS enabled: Yes
- HTTPS Port: 443
- Log file: Logweb.txt
- Log format: -
- Next log backup: 00/00/00 at 00:00
- Clear cache** button

La partie supérieure de la page fournit des informations sur le statut courant du serveur HTTP de 4D Server.

- Etat : Démarré ou Arrêté
- Date de démarrage : Date et heure du dernier lancement du serveur HTTP.
- Durée de fonctionnement : Délai écoulé depuis le dernier démarrage du serveur HTTP.
- Nombre de hits HTTP : nombre de hits HTTP (bas niveau) reçus par le serveur HTTP depuis son démarrage.

Démarrer / Arrêter le serveur HTTP

Il permet de contrôler l'activation du serveur HTTP de 4D Server.

- Lorsque l'état du serveur HTTP est "Démarré", le bouton est libellé Arrêter le serveur HTTP. Si vous cliquez sur ce bouton, le serveur HTTP de 4D Server est immédiatement stoppé, le serveur Web et le serveur SOAP n'acceptent plus aucune requête.
- Lorsque l'état du serveur HTTP est "Arrêté", le bouton est libellé Démarrer le serveur HTTP. Si vous cliquez sur ce bouton, le serveur HTTP de 4D Server est immédiatement démarré : les requêtes Web, REST et SOAP sont acceptées.

Vous devez disposer d'une licence appropriée pour pouvoir démarrer le serveur HTTP.

Le serveur HTTP peut également être lancé automatiquement au démarrage de l'application (option des Préférences) ou par programmation.

Informations Web

Cette zone fournit des informations spécifiques relatives au serveur Web de 4D Server.

- Requêtes Web : Acceptées ou Refusées. Cette information indique si le serveur Web est actif. Le serveur Web étant directement relié au serveur HTTP, les requêtes Web sont acceptées lorsque le serveur HTTP est démarré et refusées lorsqu'il est stoppé.
- Connexions maximum : Nombre maximum de connexions Web autorisées. Cette valeur dépend de la licence installée sur le poste serveur.

Informations SOAP

This area provides specific information about the SOAP server of 4D Server and includes a control button.

- SOAP requests: Accepted or Rejected. Cette information indique si le serveur SOAP est actif. Pour que les requêtes SOAP soient acceptées, le serveur HTTP doit être démarré et le serveur SOAP doit explicitement accepter les requêtes (cf. bouton Accepter/Refuser).
- Connexions maximum : Nombre maximum de connexions SOAP autorisées. Cette valeur dépend de la licence installée sur le poste serveur.
- Accepter/Refuser les requêtes SOAP : Ce bouton fonctionne en bascule. Il permet de contrôler l'activation du serveur SOAP de 4D Server. Ce bouton modifie la valeur de l'option Autoriser requêtes Web Services dans la page "Web services" des Propriétés de la base (et inversement). Vous pouvez également utiliser la commande **SOAP REJETER NOUVELLES REQUETES** pour refuser de nouvelles requêtes SOAP. Toutefois, elle ne modifie pas la valeur de l'option Autoriser requêtes Web Services .

Si vous cliquez sur le bouton Accepter les requêtes SOAP et que le serveur HTTP est arrêté, 4D le démarre automatiquement.

Configuration serveur HTTP

Cette zone fournit plusieurs informations sur les paramètres de configuration et le fonctionnement du serveur HTTP :

- Lancement automatique au démarrage : paramètre défini via les Propriétés.
- Process serveur HTTP (en cours/total) : nombre de process HTTP créés sur le serveur (nombre courant de process / cumul de tous les process créés).
- Mémoire cache : taille de la mémoire cache du serveur HTTP, lorsqu'elle est activée (taille réellement occupée par le cache / taille maximale théorique allouée au cache dans les Propriétés). You can click on the Clear Cache button to empty the current cache.
- Adresse IP d'écoute, Port HTTPS (80 par défaut), TLS activé pour les connexions HTTP (ne concerne pas les connexions SQL ou 4D) et Port HTTPS utilisés : [paramètres de configuration](#) courants du serveur HTTP, définis dans les Propriétés ou par programmation.
- Informations sur le fichier journal : nom, format et date de la prochaine sauvegarde automatique du journal du serveur HTTP (fichier logweb.txt).

Page Moniteur temps réel

La page Moniteur temps réel permet de surveiller en temps réel le déroulement des opérations "longues" effectuées par l'application. Ces opérations sont par exemple les recherches séquentielles, l'exécution de formules, etc.

The screenshot shows the 'Real Time Monitor' tab selected in the top navigation bar of the 'Moniteur - 4D Server Administration' window. A table lists two operations:

Start Time	Duration (ms)	Information
2014-06-03 10:09:26.562	75 829	Sequential searching on Table_1: 4398275 of 24728607 records
2014-06-03 10:10:27.910	14 481	Deleting records: 41998 of 24728607

To the right of the table, detailed information is provided for the selected operation (the second row):

- Created on client**
- Operation Details**
 - Operation Type: Delete Records
 - Table: Table_1
- Process Details**
 - Client Process Num: 6
 - Process Name: P_2
 - 4D User: Super_Utilisateur
 - Session Name: Arnaud Schmitt
 - Machine Name: MACWIN7-SCHMITT

At the bottom of the window, there are buttons for **PAUSED**, **Snapshot**, and **Resume**.

Cette page est disponible dans la fenêtre d'administration du poste serveur et également depuis un poste 4D distant. Dans le cas d'un poste distant, la page affiche les données des opérations effectuées sur le poste serveur.

Chaque opération longue sur les données entraîne l'ajout d'une ligne. La ligne disparaît automatiquement lorsque l'opération est terminée (l'option Afficher opérations au moins 5 secondes permet de conserver à l'écran les opérations exécutées rapidement, cf. ci-dessous).

Les informations suivantes sont fournies pour chaque ligne :

- Heure début : heure de démarrage de l'opération au format "jj/mm/aaaa - hh:mm:ss"
- Durée (ms) : durée en cours de l'opération en millisecondes
- Informations : libellé de l'opération.
- Détails : cette zone affiche un ensemble d'informations détaillées dont le contenu varie en fonction du type d'opération sélectionné. En particulier :
 - Créeé sur : indique si l'opération résulte d'une action d'un client (Créeé sur client) ou si elle a été démarrée explicitement sur le serveur via une procédure stockée ou l'option "Exécuter sur serveur" (Créeé sur serveur).
 - Détails de l'opération : décrit le type d'opération ainsi que (pour les opérations de recherche) le plan de recherche.
 - Sous-opérations (le cas échéant) : affiche les opérations dépendantes de l'opération sélectionnée (par exemple, suppression des enregistrements liés avant suppression de l'enregistrement parent).
 - Détails du process : fournit des informations supplémentaires concernant la table, le champ, le process ou le client, en fonction du type d'opération

La page d'observation en temps réel utilise en interne la commande [LTRE APERCU ACTIVITE](#). More information can be found in this command description.

La page est active et mise à jour en permanence dès qu'elle est affichée. Il est à noter que son fonctionnement peut

ralentir sensiblement l'exécution de l'application. Il est possible de suspendre la mise à jour de la page d'une des manières suivantes :

- en cliquant sur le bouton Pause,
 - en cliquant dans la liste,
 - en appuyant sur la barre d'espace.

Lorsque la page est en pause, le message "SUSPENDU" est affiché et le libellé du bouton devient Reprendre. Il est possible de reprendre l'observation des opérations en effectuant la même action que pour la mise en pause.

Mode avancé

La page MTR peut afficher des informations supplémentaires, si nécessaire, pour chaque opération listée.

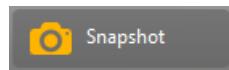
Pour accéder au mode avancé pour une opération, appuyez sur la touche Maj et sélectionnez l'opération. Toutes les informations disponibles sont alors affichées dans la zone "Détails du process" sans aucun filtre (à l'instar de ce qui est retourné par la commande `LIRE APERCU ACTIVITE`). Les informations disponibles dépendent de l'opération sélectionnée.

Voici un exemple d'information affichée en mode standard :

En mode avancé (Maj+Clic sur la ligne de l'opération), des informations supplémentaires sont affichées :

Bouton Instantané

Le bouton Instantané vous permet de copier dans le presse-papiers toutes les opérations affichées dans le panneau du MTR, ainsi que les détails associés (informations sur les process et les sous-opérations) :



Afficher opérations au moins 5 secondes

Si vous cochez l'option Afficher opérations au moins 5 secondes, toutes les opérations listées seront affichées dans la page pendant au moins cinq secondes, même après que l'exécution de l'opération soit terminée. Les opérations terminées restant affichées sont grises dans la liste. Cette fonction est utile lorsque vous voulez observer des

opérations dont l'exécution est très rapide.

WebAdmin

4D et 4D Server ont un composant intégré appelé `WebAdmin` qui permet de lancer un serveur web qui fournit un accès sécurisé à des outils de gestion de données, tel que l'[Explorateur de données Web](#). Ce serveur est accessible en local ou à distance, depuis un navigateur ou une application web, et permet d'accéder à l'application 4D associée.

Le WebAdmin gère l'authentification des utilisateurs via des privilèges "WebAdmin", leur permettant d'ouvrir des sessions en tant qu'administrateurs et d'accéder à des interfaces dédiées.

Cette fonctionnalité est disponible pour les applications 4D avec ou sans interfaces.

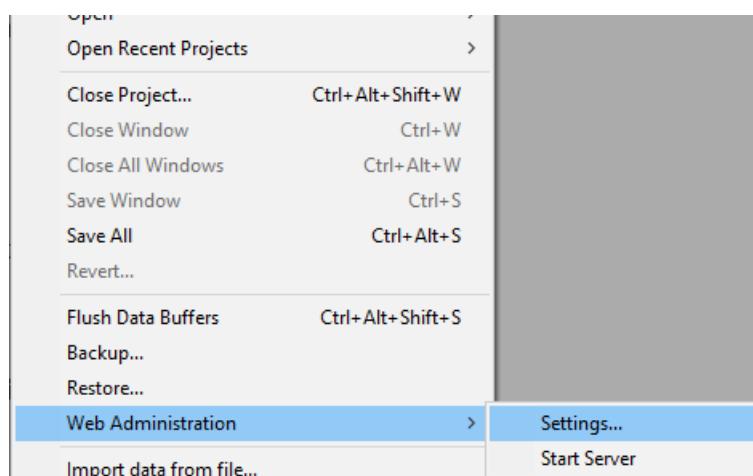
Démarrer le WebAdmin web server

Par défaut, le serveur web `WebAdmin` ne démarre pas automatiquement. Il faut configurer son lancement automatique au démarrage, ou (dans les versions avec une interface) le lancer manuellement via un menu.

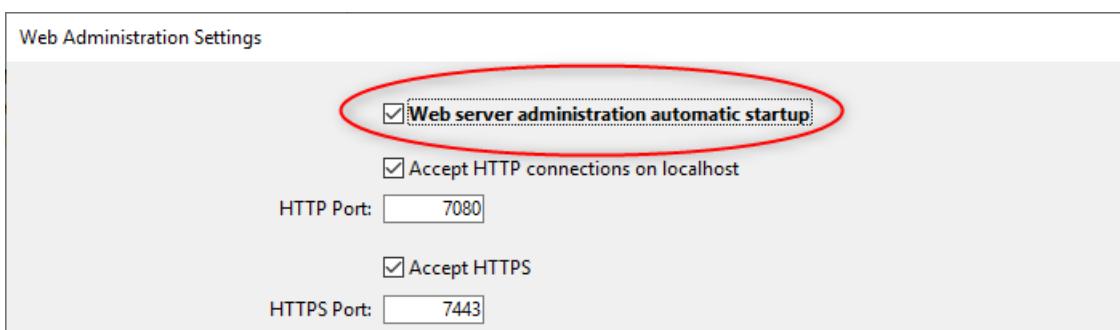
Lancement au démarrage

Vous pouvez configurer le serveur web `WebAdmin` pour qu'il se lance au démarrage de 4D ou 4D Server (avant l'ouverture d'un projet).

- Si vous utilisez une application 4D avec une interface, sélectionnez Fichier > Administration Web > Propriétés....



Cochez l'option Démarrage de l'admin du serveur web dans la fenêtre des paramètres :



- Que vous utilisez une application 4D avec ou sans interface, vous pouvez activer le lancement automatique au démarrage en utilisant l'argument suivant dans *L'interface de ligne de commande* :

```
open ~/Desktop/4D.app --webadmin=auto-start true
```

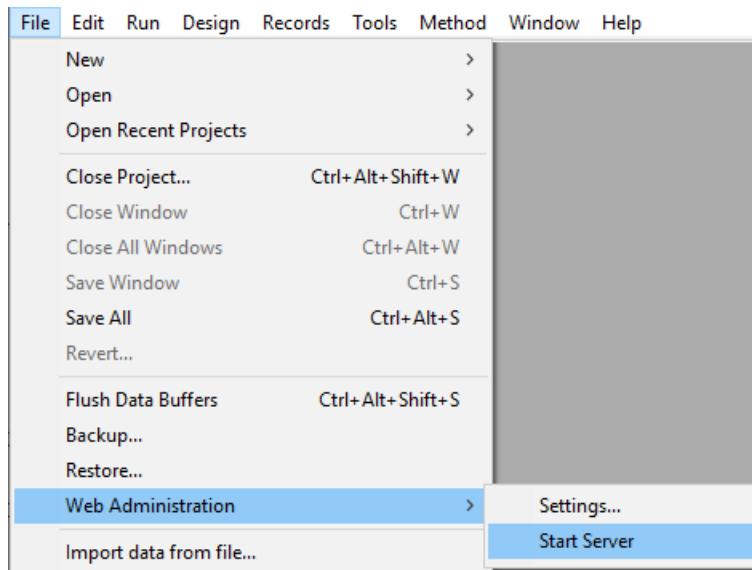
Si le port TCP utilisé par le serveur web `WebAdmin` ([HTTPS](#) ou [HTTP](#) selon les paramètres) n'est pas disponible au démarrage, 4D essaiera avec les 20 ports suivants et utilisera le premier disponible. Si aucun port n'est

disponible, le serveur web ne se lance pas et un message d'erreur s'affiche. Pour les applications sans interface, il apparaît dans la console.

Démarrage et arrêt

Si vous utilisez une application 4D avec une interface, vous pouvez démarrer ou arrêter le serveur web **WebAdmin** de votre projet à tout moment :

Sélectionnez Fichier > Administration web > Démarrer le serveur.



Le menu affiche Arrêter le Server une fois le serveur lancé. Sélectionnez Arrêter le Server pour arrêter le serveur web **WebAdmin**.

Propriétés WebAdmin

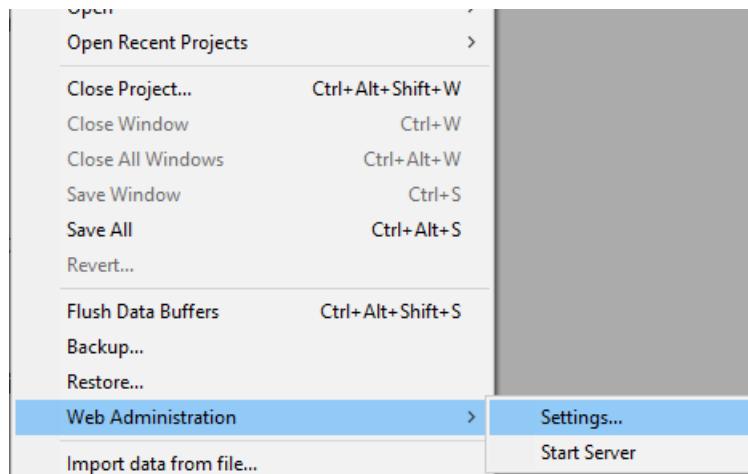
La configuration du composant **WebAdmin** est obligatoire, en particulier pour définir la [clé d'accès](#). Par défaut, quand la clé d'accès n'est pas configurée, les connexions via url ne sont pas autorisées.

Vous pouvez configurer le composant **WebAdmin** dans la [fenêtre de configuration](#)(voir ci-dessous).

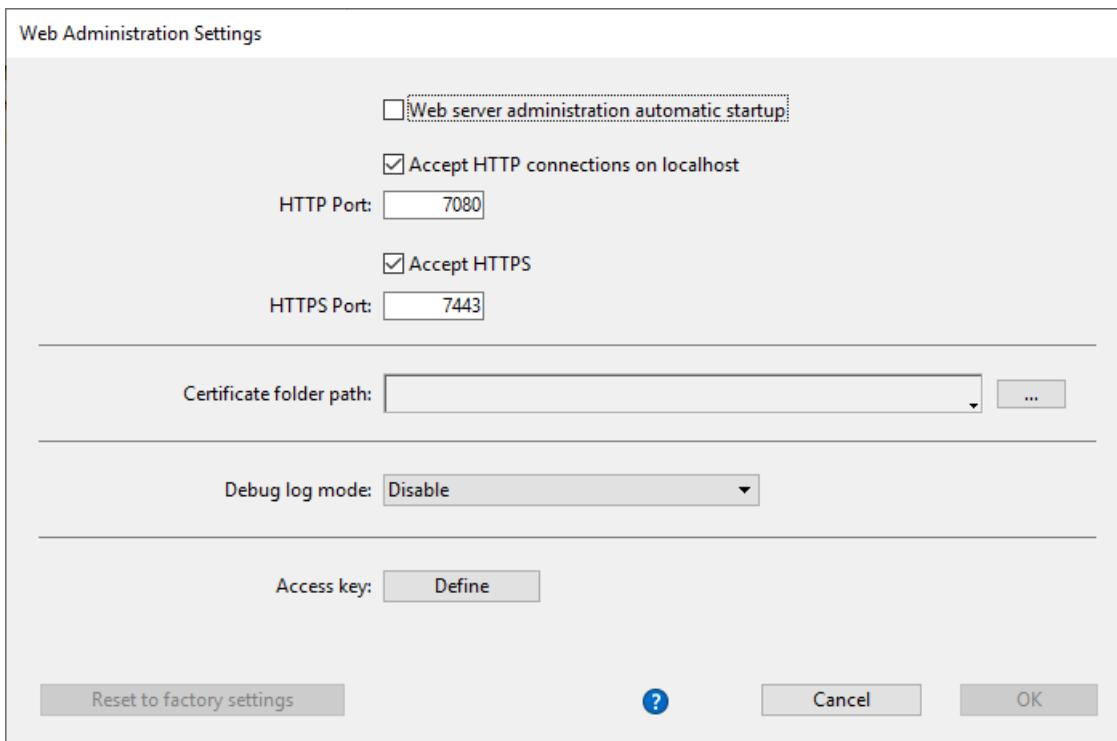
Si vous utilisez une application 4D sans interface, vous pouvez utiliser des [arguments de l'interface de ligne de commande](#) pour définir des paramètres de base. La définition de paramètres avancés se fait via le fichier de paramètres.

Fenêtre de configuration

Pour ouvrir la fenêtre de configuration des paramètres d'administration web, sélectionnez Fichier > Administration web > Propriétés....



La fenêtre suivante s'affiche :



Démarrage automatique de l'admin du serveur web

Cochez cette option pour lancer le serveur web `WebAdmin` automatiquement au démarrage de 4D ou 4D Server (voir [ci-dessus](#)). Cette option n'est pas cochée par défaut.

Connexions HTTP sur localhost acceptées

Quand cette option est cochée, il est possible de se connecter au serveur `WebAdmin` via HTTP sur la même machine que l'application 4D. Cette option est activée par défaut.

Notes :

- Les connexions HTTP autres que sur localhost ne sont jamais acceptées.
- Même si cette option est activée, quand `HTTPS Accepté` est activé et que la configuration TLS est valide, les connexions sur localhost se font via HTTPS.

Port HTTP

Numéro de port utilisé pour les connexions au serveur web `WebAdmin` via HTTP quand Connexions HTTP sur localhost acceptées est activé. La valeur par défaut est 7080.

HTTPS Accepté

Lorsque cette option est activé, vous pourrez vous connecter au serveur web `WebAdmin` via HTTPS. Cette option est activée par défaut.

Port HTTPS

Numéro de port utilisé pour les connexions au serveur web `WebAdmin` via HTTPS quand HTTPS accepté est activé. La valeur par défaut est 7443.

Chemin du dossier de certificat

Chemin du dossier qui contient les fichiers de certificat TLS. Par défaut, le chemin du dossier de certificat est vide, et 4D ou 4D server utilise les fichiers de certificat contenus dans l'application 4D (les certificats personnalisés doivent être stockés au niveau du dossier de projet).

Mode du debug log

Statut ou format du fichier de logs des requêtes HTTP (`HTTPDebugLog_nn.txt`, stocké dans le dossier "Logs" de l'application. `--nn` représente le numéro du fichier). Les options suivantes sont disponibles :

- Désactivé (valeur par défaut)
- Avec tous les body - activé avec toutes les parts des body des requêtes et réponses
- Sans les body - activé sans les parts des body (la taille du body est indiquée)
- Avec les body des requêtes - activé avec les parts des body uniquement dans les requêtes
- Avec la réponse corps - activé avec les parts des body uniquement dans les réponses

Clé d'accès

La configuration d'une clé d'accès est obligatoire pour débloquer l'accès au serveur web `webAdmin` via des URL (l'accès via les menus ne requiert pas de clé d'accès). Lorsque aucune clé d'accès n'est définie, il n'est pas possible pour les clients web d'accéder aux interfaces d'administration web telles que l'[Explorateur de données](#) via des URL. En cas de requête de connexion, une page d'erreur est retournée:



Une clé d'accès est similaire à un mot de passe, mais sans login associé.

- Pour définir une nouvelle clé d'accès, cliquez sur le bouton `Définir`, entrez une chaîne de caractères et cliquez sur `OK`. Une fois fait, le label du bouton devient `Modifier`.
- Pour modifier la clé d'accès, cliquez sur `Modifier`, entrez la nouvelle clé d'accès et cliquez sur `OK`.
- Pour supprimer la clé d'accès, cliquez sur `Modifier`, laissez le champ d'entrée vide et cliquez sur `OK`.

Configuration de WebAdmin sans interface

Pour gérer le contenu du fichier, vous pouvez utiliser la [fenêtre de paramètres WebAdmin](#) de l'application 4D avec une interface, et la lancer sans interface ensuite. Le fichier par défaut `WebAdmin.4DSettings` est alors utilisé.

Dans le cas d'une application 4D ou 4D Server sans interface, vous pouvez configurer et utiliser le fichier `WebAdmin.4DSettings` par défaut, ou désigner un fichier `.4DSettings` personnalisé.

Pour gérer le contenu du fichier, vous pouvez utiliser la [fenêtre de paramètres WebAdmin](#) de l'application 4D avec une interface, et la lancer sans interface ensuite. Par défaut, il existe un fichier `WebAdmin.4DSettings` par application 4D et 4D Server.

Vous pouvez aussi définir un fichier `.4DSettings` (format XML) et l'utiliser à la place du fichier par défaut. Plusieurs arguments dédiés sont disponibles dans [l'interface de ligne de commande](#) pour prendre en charge cette fonctionnalité.

Cette clé d'accès n'est pas stockée de façon transparente dans le fichier `.4DSettings`.

Exemple :

```
"%HOMEPATH%\Desktop\4D Server.exe" MyApp.4DLink --webadmin-access-key  
"my Fabulous AccessKey" --webadmin-auto-start true  
--webadmin-store-settings
```

Authentification et Session

- Lorsqu'on accède à une page de gestion web en entrant une URL et sans identification préalable, une authentification est nécessaire. L'utilisateur doit entrer la **clé d'accès** dans une fenêtre d'authentification. Si aucune clé d'accès n'a été définie dans les propriétés **WebAdmin**, aucun accès via URL n'est possible.
- Quando une page d'administration web est ouverte directement depuis un menu 4D ou 4D Server, tel que Enregistrements> Data Explorer ou Fenêtre> Explorateur de données(4D Server), l'accès est autorisé sans authentification.

Une fois l'accès autorisé, une **session web** est créée avec les priviléges "WebAdmin" sur l'application 4D. Tant que la session courante a le privilège "WebAdmin", le composant **WebAdmin** sert les pages demandées dans les requêtes.

Explorateur de données Web

Aperçu : L'Explorateur de données Web est fourni à titre d'aperçu. L'utilisation de cette fonctionnalité en production n'est pas recommandée. L'implémentation finale pourrait être légèrement différente.

L'Explorateur de données fournit une interface web pour visualiser et rechercher les données de votre datastore. Grâce à cet outil, vous pouvez facilement naviguer parmi toutes vos entités et rechercher, ordonner ou filtrer les valeurs des attributs. Il vous aide à contrôler les données et à identifier rapidement les problèmes rencontrés à chaque étape du développement.

The screenshot shows the 4D Data Explorer web interface. On the left, there's a sidebar with a 'Data Classes' tree containing 'Company', 'Employee' (selected), 'events', and 'Manager'. Below it is an 'Attributes' section with checkboxes for 'ID', 'firstname', 'lastname', 'salary', 'birthdate', 'woman', and 'managerID', all of which are checked. A 'Dark mode' toggle switch is also present. The main area has tabs for 'Company' and 'Employee', with 'Employee' selected. A search bar contains the query 'firstname="h@%"'. The main table lists employees with columns: ID, firstname, lastname, salary, and birthdate. The table shows 38,374 results out of 1,435,602 total. The right panel shows detailed information for employee ID 26: Hermelin SAGELAN, salary 62,409, birthdate 25/04/1975, woman status (unchecked), managerID 24, employerID 1, and a photo of a smiling man with a beard.

Configuration

L'Explorateur de données s'appuie sur le composant serveur web [WebAdmin](#) pour la configuration et les paramètres d'authentification.

- configuration : la configuration de l'Explorateur de données réutilise les [paramètres du serveur web WebAdmin](#),
- authentification : l'accès à l'Explorateur de données est accordé lorsque [l'utilisateur de la session est authentifié](#) et détient le privilège "WebAdmin". Lorsque l'on accède à l'Explorateur de données via l'élément de menu Explorateur de données (voir ci-dessous), une authentification automatique est fournie.

L'accès à l'Explorateur de données peut être désactivé à l'aide de la fonction [.setAdminProtection\(\)](#).

Ouverture de l'Explorateur de données

La page Explorateur de données est automatiquement disponible lorsque [le serveur Web de WebAdmin est lancé](#).

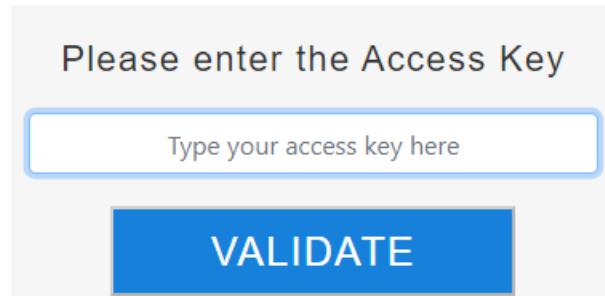
Pour se connecter à la page web de l'Explorateur de données :

- si vous utilisez une application 4D avec interface, sélectionnez la commande Explorateur de données... à partir :

- du menu Enregistrements (en 4D monoposte)
- du menu Window (dans 4D Server)
- que vous utilisez une application 4D headless ou non, vous pouvez ouvrir votre navigateur web et entrer l'adresse suivante :

`IPaddress:HTTPPort/dataexplorer` or `IPaddress:HTTPSPort/dataexplorer`

Dans ce contexte, vous serez invité à saisir la [clé d'accès](#) pour ouvrir une session `WebAdmin` sur le serveur :



Les valeurs `HTTPPort` et `HTTPSPort` sont configurées dans les paramètres de `WebAdmin`.

Utilisation de l'Explorateur de données

En plus d'une vue complète et personnalisable de vos données, l'Explorateur de données vous permet de rechercher et d'ordonner vos données.

Conditions requises

L'Explorateur de données prend en charge les navigateurs Web suivants :

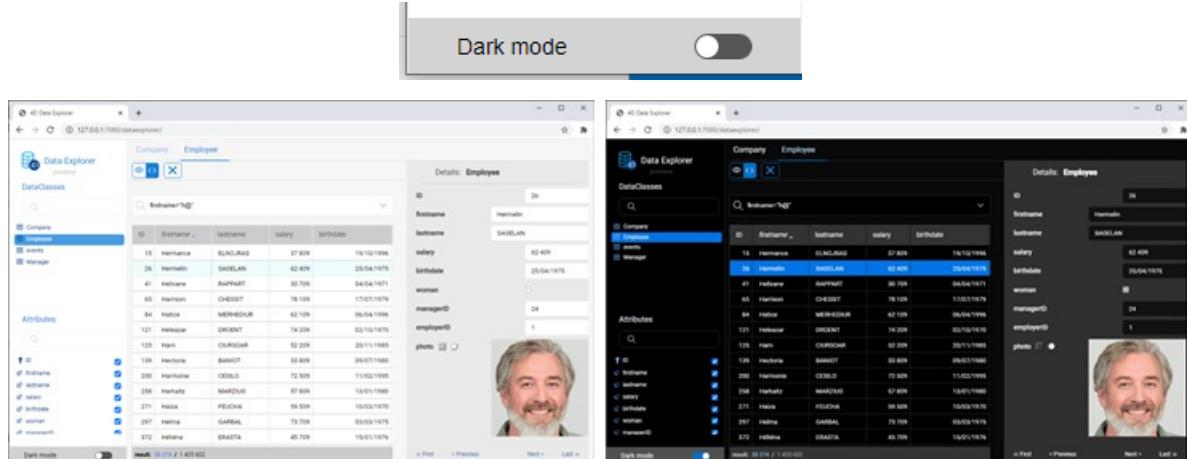
- Chrome
- Safari
- Edge
- FireFox

La résolution minimale pour utiliser l'Explorateur de données est de 1280x720. La résolution recommandée est de 1920x1080.

Principes de base

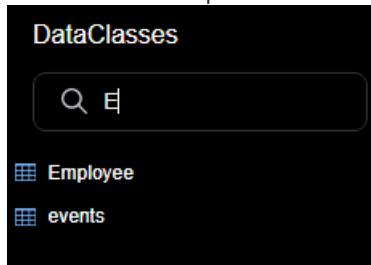
L'Explorateur de données fournit un accès global au modèle de données ORDA conformément aux [règles de mapping ORDA](#).

Vous pouvez passer au thème d'affichage mode sombre à l'aide du sélecteur situé en bas de la page :



La page contient plusieurs zones :

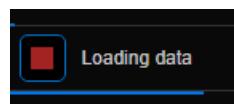
- Sur le côté gauche, se trouvent la zone des Dataclass et la zone des attributs, pour sélectionner les dataclass et les attributs à afficher. Les attributs sont classés selon l'ordre de création de la structure sous-jacente. Les clés primaires et les attributs indexés ont une icône spécifique. Vous pouvez filtrer la liste des noms de dataclasses et d'attributs proposés en utilisant les zones de recherche respectives.



- La partie centrale contient la Zone de recherche et la Grille de données (liste des entités de la dataclass sélectionnée). Chaque colonne de la grille représente un attribut du datastore.
 - Par défaut, toutes les entités sont affichées. Vous pouvez filtrer les entités affichées en utilisant la zone de recherche. Deux modes de requête sont disponibles : [Requête sur les attributs](#) (sélectionné par défaut), et la [Requête avancée avec expression](#). Vous sélectionnez le mode de requête en cliquant sur le bouton correspondant (le bouton X permet de réinitialiser la zone de requête et donc de stopper le filtrage) :



- Le nom de la dataclass sélectionnée est ajouté sous forme d'onglet au-dessus de la grille de données. A l'aide de ces onglets, vous pouvez passer d'une dataclasse à une autre qui a déjà été sélectionnée. Vous pouvez supprimer une dataclasse référencée en cliquant sur l'icône "supprimer" à droite du nom de la dataclass.
- Vous pouvez réduire le nombre de colonnes en décochant les attributs dans la partie gauche. Vous pouvez également changer les colonnes dans la grille de données à l'aide du glisser-déposer. Vous pouvez cliquer sur l'en-tête d'une colonne pour [trier les entités](#) en fonction de ses valeurs (lorsque cela est possible).
- Si une opération nécessite beaucoup de temps, une barre de progression s'affiche. Vous pouvez arrêter l'opération en cours à tout moment en cliquant sur le bouton rouge :



- Sur le côté droit, se trouve la zone de Détails : elle affiche les valeurs des attributs de l'entité courante sélectionnée. Tous les types d'attributs sont affichés, y compris les images et les objets (exprimés en json). Vous pouvez naviguer entre les entités de la dataclass en cliquant sur les liens Premier / Précédent / Suivant / Dernier en bas de la zone.

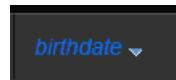
Mise à jour des contenus

Lorsque les données ou le modèle ORDA sont modifiés du côté de la base de données (ajout de table, enregistrement édité ou supprimé, etc.), il suffit de rafraîchir la page de l'Explorateur de données dans le navigateur (à l'aide de la touche F5, par exemple).

Ordonner les entités

Vous pouvez réorganiser la liste des entités affichées en fonction des valeurs des attributs. Tous les types d'attributs peuvent être utilisés pour un tri, sauf l'image et l'objet.

- Cliquez sur un en-tête de colonne pour ordonner les entités en fonction des valeurs d'attribut correspondantes. Par défaut, le tri est ascendant. Cliquez deux fois pour un tri décroissant. Une colonne utilisée pour trier les entités est affichée avec une petite icône et son nom est en *italique*.



- Vous pouvez trier les attributs sur plusieurs niveaux. Par exemple, vous pouvez trier les employés par ville, puis par salaire. Pour ce faire, maintenez la touche Maj enfoncée et cliquez successivement sur chaque en-tête de colonne à inclure dans l'ordre de tri.

Requête sur les attributs

Dans ce mode, vous pouvez filtrer les entités en saisissant les valeurs à rechercher (ou à exclure) dans les zones situées au-dessus de la liste des attributs. Vous pouvez filtrer un ou plusieurs attributs. La liste des entités est automatiquement mise à jour lors de la saisie.

The screenshot shows a search interface with three buttons at the top: a magnifying glass, a refresh arrow, and a close X. Below is a search input field containing '= ID Flo'. A table below the input shows three rows: the first row has columns 'ID' and 'firstname'; the second row contains '1' and 'Florette'; the third row contains '29' and 'Flora'. The fourth row contains '1 200' and 'Floria'. The entire interface is dark-themed with blue highlights.

ID	firstname
1	Florette
29	Flora
1 200	Floria

Si vous saisissez plusieurs attributs, un "ET" est automatiquement appliqué. Par exemple, le filtre suivant affiche les entités dont l'attribut *prénom* commence par "flo" ET la valeur de l'attribut *salaire* > 50000 :

The screenshot shows a search interface with three buttons at the top: a magnifying glass, a refresh arrow, and a close X. Below is a search input field containing 'Flo > 50000 ='. A table below the input shows four rows: the first row has columns 'firstname' and 'salary'; the second row contains 'Flora' and '76 109'; the third row contains 'Floria' and '58 009'; the fourth row contains 'Florent' and '79 409'. The entire interface is dark-themed with blue highlights.

firstname	salary
Flora	76 109
Floria	58 009
Florent	79 409

Le bouton X permet de supprimer les attributs saisis et de stopper ainsi le filtrage.

Différents opérateurs et options de requête sont disponibles, en fonction du type de données de l'attribut.

Vous ne pouvez pas filtrer sur les attributs d'image ou d'objet.

Opérateurs numériques

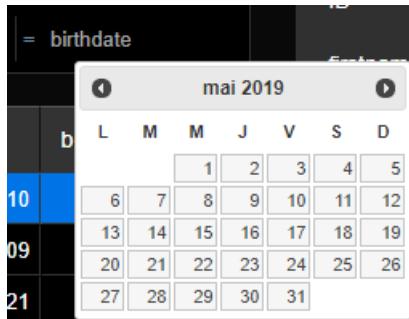
Avec les attributs numériques, de date et d'heure, l'opérateur "=" est sélectionné par défaut. Toutefois, vous pouvez sélectionner un autre opérateur dans la liste des opérateurs (cliquez sur l'icône "=" pour afficher la liste) :

The screenshot shows a dropdown menu for an operator. The menu items are: '=' (selected), '<', '<=' (disabled), '>', '>=' (disabled), '≠' (disabled), and '29 309' (disabled). Below the menu is a table with two rows: the first row has columns 'salary' and '29 309'; the second row has columns 'salary' and '73 109'. The entire interface is dark-themed with blue highlights.

=	salary
<	

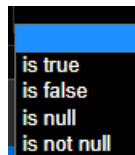
Dates

Avec les attributs de date, vous pouvez saisir la date à utiliser à l'aide d'un widget de sélection de date (cliquez sur la zone de la date pour afficher le calendrier) :



Booléens

Lorsque vous cliquez sur une zone d'attribut booléen, vous pouvez filtrer sur les valeurs true/false mais aussi sur les valeurs null/not null :



- null indique que la valeur de l'attribut n'a pas été définie
- not null indique que la valeur de l'attribut est définie (donc true ou false).

Text

Les filtres texte ne sont pas diacritiques (a = A).

Le filtre est du type "commence par". Par exemple, si vous saisissez "Jim", vous obtiendrez les valeurs "Jim" et "Jimmy".

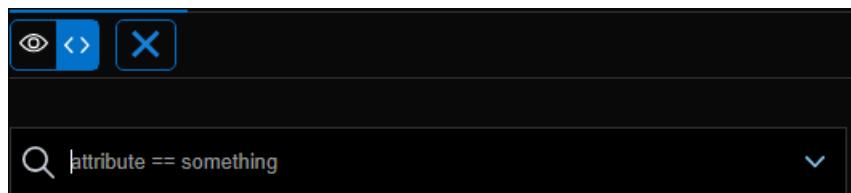
Vous pouvez également utiliser le caractère générique (@) pour remplacer un ou plusieurs caractères de départ. Par exemple :

Un filtre avec	Résultats
Bel	Toutes les valeurs qui commencent par "Bel"
@do	Toutes les valeurs contenant "do"
Bel@do	Toutes les valeurs commençant par "Bel" et contenant "do"

Si vous souhaitez créer des requêtes plus spécifiques, telles que "est exactement", vous devrez utiliser la fonction de requêtes avancées.

Requêtes avancées avec des expressions

Lorsque vous sélectionnez cette option, une zone de requête s'affiche au-dessus de la liste des entités, et vous permet de saisir une expression à utiliser pour filtrer le contenu :



Vous pouvez saisir des requêtes avancées qui ne sont pas disponibles en tant que requêtes d'attributs. Par exemple, si vous voulez trouver des entités dont l'attribut *firstname* contient "Jim" mais pas "Jimmy", vous pouvez écrire :

```
firstname=="Jim"
```

Vous pouvez utiliser n'importe quelle expression de requête ORDA, comme [documenté avec la fonction query\(\)](#), avec les limitations ou différences suivantes :

- Par sécurité, vous ne pouvez pas exécuter de formules en utilisant `eval()`.
- Les placeholders ne peuvent pas être utilisés ; vous devez saisir une `queryString` avec des valeurs.
- Les valeurs de chaîne contenant des caractères d'espacement doivent être intégrées dans des guillemets doubles ("").

Par exemple, avec la dataclass Employee, vous pouvez écrire :

```
firstname = "Marie Sophie" ET manager.lastname = "@th"
```

Vous pouvez cliquer sur l'icône  pour afficher à la fois `queryPlan` et `queryPath`. Dans cette zone, vous pouvez survoler les blocs de sous-requêtes pour avoir des informations détaillées par sous-requête :



Faites un clic droit dans la zone des requêtes pour afficher les précédentes requêtes valides :

ID	firstname	lastname	salary
2	isabelle	smith	30000
3	Narime	smith	30000

Details of the valid queries shown in the background:

```
firstname = "h@%" AND lastname == "sm@%" OR salary > 30000
firstname = "h@%" AND lastname == "sm@%" OR salary > 30000
firstname = "h@"
firstname == "Jim"
firstname = "M" and lastname = "@th"
firstname = "M@%" and lastname = "@th"
firstname = "M@%" or lastname = "@th"
firstname = "h@%" AND lastname == "smith" OR salary > 30000
```

Interface de ligne de commande

Vous pouvez utiliser le terminal macOS ou la console Windows pour piloter vos applications 4D (4D et 4D Server) à l'aide de lignes de commande. Cette fonctionnalité vous permet notamment :

- de lancer une base de données à distance, ce qui peut être particulièrement utile pour administrer des serveurs Web.
- d'exécuter des tests automatiques pour vos applications.

Informations de base

Vous pouvez exécuter des lignes de commande pour les applications 4D à l'aide du terminal macOS ou de la console Windows.

- Sous macOS, vous devez utiliser la commande `open`.
- Sous Windows, vous pouvez simplement passer les arguments directement.

Sous macOS, vous pouvez passer les arguments directement en allant dans le dossier contenant l'application, à l'intérieur du package (Contents/MacOS), ce qui permet d'adresser le flux stderr. Par exemple, si le package 4D se trouve dans le dossier `MyFolder`, vous devez écrire la ligne de commande comme suit :
`/MyFolder/4D.app/Contents/MacOS/4D`. Nous vous recommandons cependant d'utiliser la commande `open` chaque fois que vous n'avez pas besoin d'accéder au flux stderr.

Lancer une application 4D

Voici une description des lignes de commande et des arguments pris en charge pour le lancement d'applications 4D.

Syntaxe :

```
<applicationPath> [--version] [--help] [--project] [<projectPath | packagePath | 4dlinkPath> [--data <dataPath>] [--opening-mode interpreted | compiled] [--create-data] [--user-param <user string>] [--headless] [--datadir <datadir>] [--webadmin-settings-file] [--webadmin-access-key] [--webadmin-auto-start] [--webadmin-store-settings]
```

Argument	Valeur	Description
<code>applicationPath</code>	Chemin de 4D, 4D Server ou de l'application fusionnée	Lance l'application. Identique au double-clic sur l'application 4D. Lorsqu'elle est appelée sans argument de fichier de structure, l'application est exécutée et la boîte de dialogue «sélectionner la base de données» apparaît.
<code>--version</code>		Affiche la version de l'application et quitte
<code>--help</code>		Affiche le message d'aide et quitte. Autres arguments : <code>-?</code> , <code>-h</code>
<code>--project</code>	<code>projectPath packagePath 4dlinkPath</code>	Fichier de projet à ouvrir avec le fichier de données courant. Aucune boîte de dialogue n'apparaît.
<code>--data</code>	<code>dataPath</code>	Fichier de données à ouvrir avec le fichier de projet désigné. S'il n'est pas indoqué, 4D utilise le dernier fichier de données ouvert.
<code>--opening-mode</code>	<code>interpreted compiled</code>	Base de données de requêtes à ouvrir en mode interprété ou compilé. Aucune erreur n'est générée si le mode demandé n'est pas disponible.
<code>--create-data</code>		Crée automatiquement un nouveau fichier de données si aucun

Argument	Valeur	Description
		<p>Création automatiquement d'un nouveau fichier de données standard.</p> <p>Si aucun argument de données valide n'est trouvé. Aucune boîte de dialogue n'apparaît. 4D utilise le nom de fichier passé dans l'argument "--data" s'il en existe un (génère une erreur si un fichier du même nom existe déjà).</p>
--user-param	Chaîne utilisateur personnalisée	<p>Une chaîne qui sera disponible dans l'application 4D via la commande Get database parameter (la chaîne ne doit pas commencer par un caractère "-", qui est réservé).</p>
--headless		<p>Lance 4D, 4D Server ou l'application fusionnée sans interface (mode headless). In this mode:</p> <ul style="list-style-type: none">• Le mode Développement n'est pas disponible, la base de données démarre en mode Application• Aucune barre d'outils, barre de menus, fenêtre MDI ou écran d'accès direct ne s'affiche• Aucune icône n'est affichée dans le dock ou la barre des tâches• La base de données ouverte n'est pas enregistrée dans le menu "Bases de données récentes"• Le journal de diagnostic est automatiquement lancé (voir [SET DATABASE PARAMETER] (https://doc.4d.com/4dv19/help/command/en/page642.html) sélecteur 79)• Chaque appel à une boîte de dialogue est intercepté et une réponse automatique est fournie (par exemple OK pour la commande [ALERT] (https://doc.4d.com/4dv19/help/command/en/page41.html), Abort pour un boîte de dialogue d'erreur, etc.). Toutes les commandes interceptées (*) sont enregistrées dans le journal de diagnostic. <p>Pour les besoins de maintenance, vous pouvez envoyer n'importe quel texte aux flux de sortie standard à l'aide de la commande LOG EVENT. A noter que les applications 4D headless ne peuvent être fermées qu'en appelant QUIT 4D ou en utilisant le gestionnaire de tâches du système d'exploitation</p>
--dataless		<p>Lance 4D, 4D Server ou une application fusionnée en mode headless. Le mode Dataless est utile lorsque 4D exécute des tâches sans données (compilation de projet par exemple). In this mode:</p> <ul style="list-style-type: none">• Aucun fichier contenant des données n'est ouvert, même s'il est spécifié dans la ligne de commande ou le fichier .4DLink, ou lors de l'utilisation des commandes CREATE DATA FILE et OPEN DATA FILE.• Les commandes qui manipulent les données généreront une erreur. Par exemple : «CREATE RECORD» lance «aucune table à laquelle appliquer la commande». <p>Note:</p> <ul style="list-style-type: none">• S'il est passé en ligne de commande, le mode dataless s'applique à toutes les bases de données ouvertes dans 4D, tant que l'application n'est pas fermée.• S'il est passé à l'aide du fichier .4DLink, le mode dataless ne s'applique qu'à la base de données spécifiée dans le fichier .4DLink. Pour plus d'informations sur les fichiers .4DLink, voir [Raccourcis d'ouverture de projet] (..../GettingStarted/creating.md#project-opening-shortcuts).
--webadmin-settings-file	Chemin de fichier	Chemin du fichier WebAdmin .4DSettings personnalisé pour le serveur Web WebAdmin
--webadmin-access-key	Chaine	Clé d'accès au serveur Web WebAdmin

WebAdmin Access Key	Champ	Clé d'accès au serveur Web WebAdmin
Argument --webadmin-auto-start	Valeur Booléen	Description Statut du démarrage automatique du serveur Web WebAdmin
--webadmin-store-settings		Stocke la clé d'accès et les paramètres de démarrage automatique dans le fichier de paramètres courant (c'est-à-dire le fichier WebAdmin.4DSettings par défaut ou un fichier personnalisé désigné par le paramètre <code>--webadmin-settings-path</code>). Utilisez l'argument <code>--webadmin-store-settings</code> pour sauvegarder ces paramètres si nécessaire

[fichier journal de diagnostic](#) (alerte de licence, boîte de dialogue de conversion, sélection de la base de données, sélection du fichier de données). Dans ce cas, un message d'erreur est envoyé à la fois dans le flux stderr et dans le journal d'événements système, puis l'application se ferme.

Exemples

Ces exemples supposent que votre application 4D est stockée sur le bureau et que la base de données à ouvrir se trouve dans le dossier "Documents".

Le dossier courant de l'utilisateur est atteint à l'aide de la commande "~" sous macOS et de la commande "%HOMEPATH%" sous Windows.

Ces exemples supposent que votre application 4D est stockée sur le bureau et que la base de données à ouvrir se trouve dans le dossier "Documents".

- Sous macOS :

```
open ~/Desktop/4D.app
```

- Sous Windows :

```
%HOMEPATH%\Desktop\4D\4D.exe
```

Lancer l'application :

```
yarn open ~/Desktop/4D.app --args ~/Documents/myDB.4dbase
```

Lancer l'application avec un fichier de package sur macOS :

- Sous macOS :

```
yarn open ~/Desktop/4D.app --args ~/Documents/myProj/Project/myProj.4DProject
```

- Sous Windows :

```
%HOMEPATH%\Desktop\4D\4D.exe %HOMEPATH%\Documents\myProj\Project\myProj.4DProject
```

Lancer l'application avec un fichier projet :

- Sous macOS :

```
open ~/Desktop/4D.app --args --project ~/Documents/myProj/Project/myProj.4DProject --data ~/Documents/da
```

- Sous Windows :

```
%HOMEPATH%\Desktop\4D\4D.exe --project %HOMEPATH%\Documents\myProj\Project\myProj.4DProject --data %HOME  
ou :  
%HOMEPATH%\Desktop\4D\4D.exe /project %HOMEPATH%\Documents\myProj\Project\myProj.4DProject /data %HOMEP
```

Lancer l'application avec un fichier projet et un fichier de données :

- Sous macOS :

```
open ~/Desktop/4D.app MyDatabase.4DLink
```

```
open "~/Desktop/4D Server.app" MyDatabase.4DLink
```

- Sous Windows :

```
%HOMEPATH%\Desktop\4D.exe MyDatabase.4DLink
```

```
%HOMEPATH%\Desktop\4D Server.exe" MyDatabase.4DLink
```

Lancer l'application avec un fichier .4DLink :

- Sous macOS :

```
open ~/Desktop/4D.app ~/Documents/myBase.4dbase --args --opening-mode compiled --create-data true
```

- Sous Windows :

```
%HOMEPATH%\Desktop\4D\4D.exe %HOMEPATH%\Documents\myBase.4dbase\myDB.4db --opening-mode compiled --creat
```

Lancez l'application en mode compilé et créer un fichier de données s'il n'est pas disponible :

- Sous macOS :

```
open ~/Desktop/4D.app --args --project ~/Documents/myProj/Project/myProj.4DProject --data ~/Documents/da
```

- Sous Windows :

```
%HOMEPATH%\Desktop\4D\4D.exe --project %HOMEPATH%\Documents\myProj\Project\myProj.4DProject --data %HOME
```

Lancer l'application avec un fichier projet et un fichier de données et passer une chaîne comme paramètre utilisateur :

- Sous macOS :

```
open ~/Desktop/4D.app --args --project ~/Documents/myProj/Project/myProj.4DProject --data ~/Documents/da
```

```
open ~/Desktop/MyBuiltRemoteApp --headless
```

- Sous Windows :

```
%HOMEPATH%\Desktop\4D\4D.exe --project %HOMEPATH%\Documents\myProj\Project\myProj.4DProject --data %HOME%  
%HOMEPATH%\Desktop\4D\MyBuiltRemoteApp.exe --headless
```

Protocole TLS (HTTPS)

Tous les serveurs 4D peuvent communiquer en mode sécurisé via le protocole TLS (Transport Layer Security) :

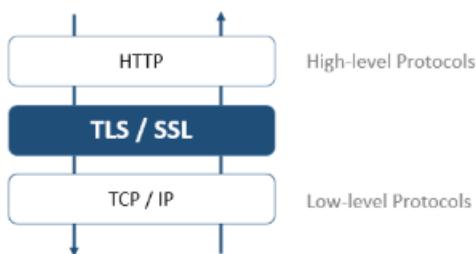
- le serveur web
- le serveur d'application (application de bureau client-serveur)
- le serveur SQL

Aperçu

Le protocole TLS (successeur du SSL) a pour but de sécuriser les informations de données entre deux applications— principalement entre un serveur web et un navigateur. Ce protocole est largement répandu et compatible avec la plupart des navigateurs web.

Au niveau de l'architecture réseau, le protocole de sécurité s'insère entre la couche TCP/IP (bas niveau) et le protocole de haut niveau HTTP, pour lequel il est principalement destiné.

Configuration réseau avec TSL :



Le protocole TLS a pour but d'authentifier l'émetteur et le récepteur et de garantir la confidentialité et l'intégrité des informations échangées :

- Authentification : l'identité de l'émetteur et du récepteur sont confirmées.
- Confidentialité: les données envoyées sont cryptées afin de les rendre inintelligibles pour les tiers non autorisés.
- Intégrité: les données reçues n'ont pas été altérées, frauduleusement ou accidentellement.

Les principes de sécurisation utilisés par TLS sont basés sur l'emploi d'un algorithme de cryptage utilisant une paire de clés : une clé privée et une clé publique. La clé privée est utilisée pour crypter les données. Elle est conservée par l'émetteur (le site Web). La clé publique est utilisée pour décrypter les données. Elle est diffusée auprès des récepteurs (les navigateurs web) via le certificat. L'emploi du TLS dans le cadre d'Internet requiert en effet l'entremise d'une Autorité de Certification telle que Verisign®. Moyennant une participation financière du site Web demandeur, cet organisme délivre un certificat, garantissant l'identité du serveur et contenant la clé publique permettant la communication en mode sécurisé.

Pour plus d'informations sur les principes généraux de cryptage et d'emploi de clés publiques/clés privées, voir la description de la commande `CRYPTER BLOB`.

Version minimale

Par défaut, la version minimale acceptée par le serveur est TLS 1.2. Vous pouvez modifier cette valeur via le sélecteur `Min version TLS` de la commande `FIXER PARAMETRE BASE`.

Vous pouvez définir le niveau de sécurité de votre serveur web en définissant la [version TLS minimale](#) acceptée pour les connexions.

Obtenir un certificat

La mise en place d'un serveur fonctionnant en TLS nécessite un certificat numérique délivré par un opérateur de certification. Ce certificat renferme diverses informations dont la carte d'identité du site ainsi que la clé publique utilisée pour communiquer avec lui. Il est transmis aux clients (navigateurs Web) se connectant au site. Une fois qu'il est accepté, la communication en mode sécurisé s'établit.

Les navigateurs Web autorisent uniquement les certificats émis par une autorité de certification référencée dans leurs propriétés.



Le choix de l'autorité de certification dépend de plusieurs facteurs. Plus l'autorité est "connue", plus le nombre de navigateurs acceptant les certificats qu'elle délivre sera important, mais plus le prix à payer sera élevé.

Pour obtenir un certificat numérique :

1. Générez une "clé privée" à l'aide de la commande `GENERER CLES CRYPTAGE`.

Attention : Pour des raisons de sécurité, la clé privée ne doit jamais être diffusée sur un réseau. En fait, elle ne doit pas quitter le poste serveur. Pour le serveur Web, le fichier Key.pem doit être placé dans le dossier de la structure du projet.

2. Etablissez une demande de certificat à l'aide de la commande `GENERER DEMANDE CERTIFICAT`.

3. Envoyez la demande de certificat à l'autorité de certification que vous avez choisie.

Pour remplir la demande de certificat, il vous sera peut-être nécessaire de contacter l'autorité de certification. Les autorités de certification vérifient la réalité des informations qui leur ont été transmises. La demande de certificat est générée dans un BLOB au format PKCS encodé en base64 (format PEM). Ce principe autorise le copier-coller des clés sous forme de texte et leur envoi par E-mail en toute sécurité, sans risque d'altération de leur contenu. Vous pouvez donc par exemple sauvegarder le BLOB contenant la demande de certificat dans un document texte (à l'aide de `BLOB VERS DOCUMENT`), puis l'ouvrir et copier-coller son contenu dans un E-mail ou un formulaire Web destiné à l'autorité de certification.

4. Une fois que vous avez reçu votre certificat, créez un fichier texte que vous nommerez "cert.pem" et copiez dans ce fichier le contenu du certificat.

Vous pouvez recevoir votre certificat sous plusieurs formes (généralement via un E-mail ou un formulaire HTML). 4D accepte la plupart des formats de texte (macOS, PC, Linux...) pour les certificats. En revanche, le certificat doit être au format PEM, c'est-à-dire PKCS encodé en base64.

Les caractères de fins de ligne CR ne sont pas pris en charge. Vous devez utiliser CRLF ou LF.

5. Placez le fichier "cert.pem" à [l'emplacement adéquat](#).

Le serveur Web peut dès lors fonctionner en mode sécurisé. La durée de validité d'un certificat varie généralement entre trois mois et un an.

Installation et activation

Installer des fichiers `key.pem` et `cert.pem`

Pour pouvoir utiliser le protocole TLS avec le serveur, vous devez installer `key.pem` (document contenant la clé de chiffrement privée) et `cert.pem` (document contenant le certificat) au(x) emplacement(s) approprié(s). Différents emplacements sont nécessaires en fonction du serveur sur lequel vous souhaitez utiliser TLS.

Des fichiers `key.pem` et `cert.pem` par défaut sont fournis avec 4D. Pour un niveau de sécurité plus élevé, nous vous recommandons fortement de remplacer ces fichiers avec vos propres certificats.

Avec le serveur Web

Pour être utilisés par le serveur web de 4D, les fichiers `key.pem` et `cert.pem` doivent être placés :

- avec 4D en mode local ou 4D Server, à côté du [dossier du projet](#)
- avec 4D en mode distant, dans le dossier de la base de données cliente sur la machine distante (pour plus d'informations sur l'emplacement de ce dossier, consultez la commande [Get 4D folder](#)).

Vous devez copier ces fichiers manuellement sur la machine distante.

Avec le serveur d'applications (applications de bureau client-serveur)

Pour être utilisés par le serveur d'applications de 4D, les fichiers `key.pem` et `cert.pem` doivent être placés :

- dans le dossier [Resources](#) de l'application 4D Server
- et dans le dossier Resources de chaque application 4D distante (pour plus d'informations sur l'emplacement de ce dossier, consultez la commande [Get 4D folder](#)).

Avec le serveur SQL

Pour être utilisés par le serveur SQL de 4D, les fichiers `key.pem` et `cert.pem` doivent être placés à côté du [dossier du projet](#).

Activation du TLS

L'installation de fichiers `key.pem` et `cert.pem` permet d'utiliser TLS avec le 4D Server. Cependant, pour que les connexions TLS soient acceptées par le server, il est nécessaire de les activer :

- Avec le serveur web de 4D, vous devez [autoriser le HTTPS](#). Vous pouvez activer l'[option HSTS](#) pour rediriger les navigateurs qui tentent de se connecter via HTTP.
- Avec le serveur d'applications, vous devez sélectionner l'option Crypter les communications Client-Serveur dans la page Client-Serveur des Propriétés.
- Avec le serveur SQL, vous devez sélectionner l'option Activer TLS dans la page "SQL" des Propriétés.

Le serveur web 4D prend également en charge l'[option HSTS](#) pour déclarer que les navigateurs doivent interagir avec lui uniquement via des connexions HTTPS sécurisées.

Perfect Forward Secrecy (PFS)

Le [PFS](#) ajoute une couche de sécurité supplémentaire à vos communications. Plutôt que d'utiliser des clés d'échanges préétablies, la PFS crée des clés de session de manière coopérative entre les parties en communication en utilisant des algorithmes Diffie-Hellman (DH). Le mode conjoint de production des clés crée un "secret partagé" qui empêche des éléments externes de les compromettre.

Lorsque TLS est activé sur le serveur Web de 4D Web, PFS est automatiquement activé. Si le fichier `dhpamrs.pem` (document contenant la clé DH privée du serveur) n'existe pas déjà, 4D le génère automatiquement avec une taille de clé de 2048. La génération initiale de ce fichier peut prendre plusieurs minutes. Le fichier est placé avec les fichiers `key.pem` et `cert.pem`.

Si vous utilisez une [liste de chiffrement personnalisée](#) et souhaitez activer PFS, vérifiez que votre liste contient des entrées avec des algorithmes DH ou ECDH (courbes elliptiques de Diffie–Hellman).

Gestion des licences 4D

Une fois installés sur votre disque, les produits 4D doivent être activés pour que vous puissiez les utiliser. Habituellement, l'activation est automatique si vous [vous connectez à l'aide de votre compte 4D](#) dans l'assistant de bienvenue.

Cependant, dans des cas spécifiques, vous pourriez avoir besoin d'activer vos licences manuellement, si par exemple :

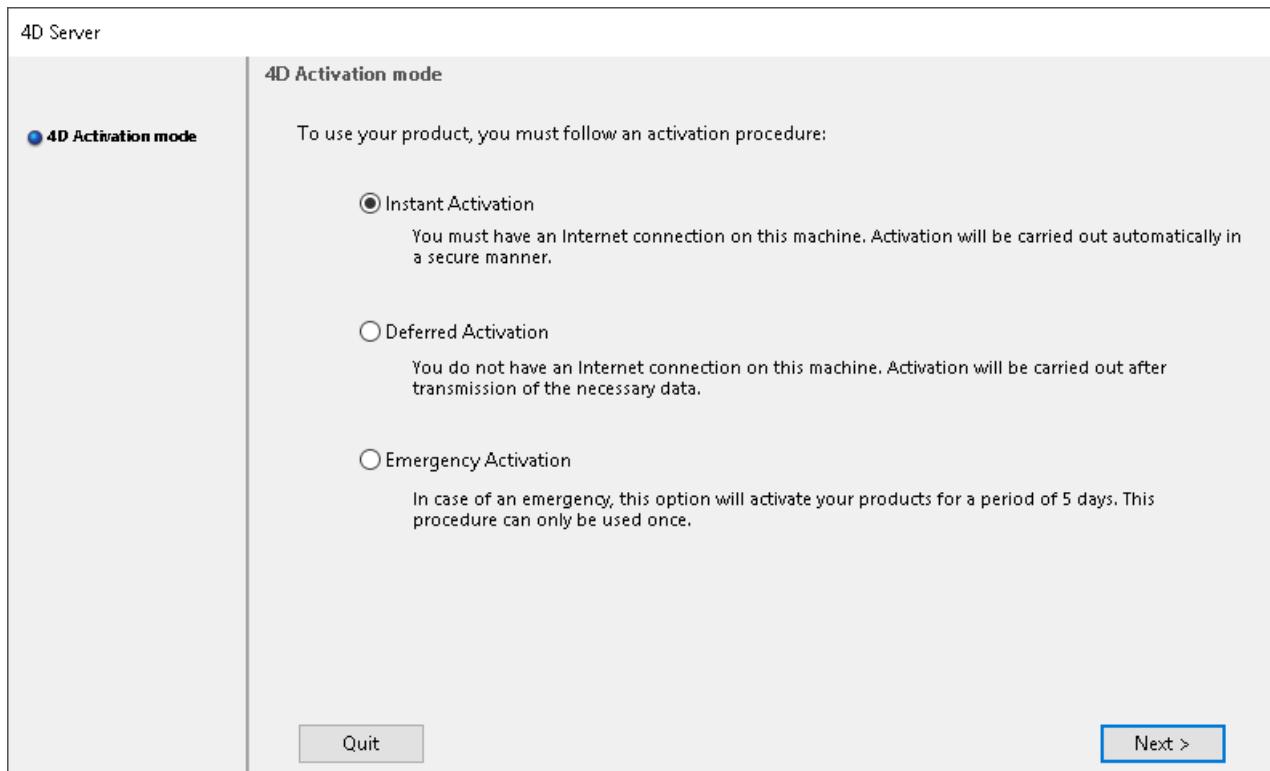
- votre configuration ne permet pas l'activation automatique,
- vous avez acheté des licences supplémentaires.

Aucune activation n'est requise pour les usages suivants :

- 4D utilisé en mode distant (connexion à un 4D Server)
- 4D utilisé en mode local avec un projet d'application interprété sans accès au mode Développement.

Première activation

Pour activer 4D, sélectionnez la commande Gestionnaire de licences... du menu Aide. Pour activer 4D Server, lancez l'application 4D Server. La boîte de dialogue de choix du [mode d'activation](#) apparaît.

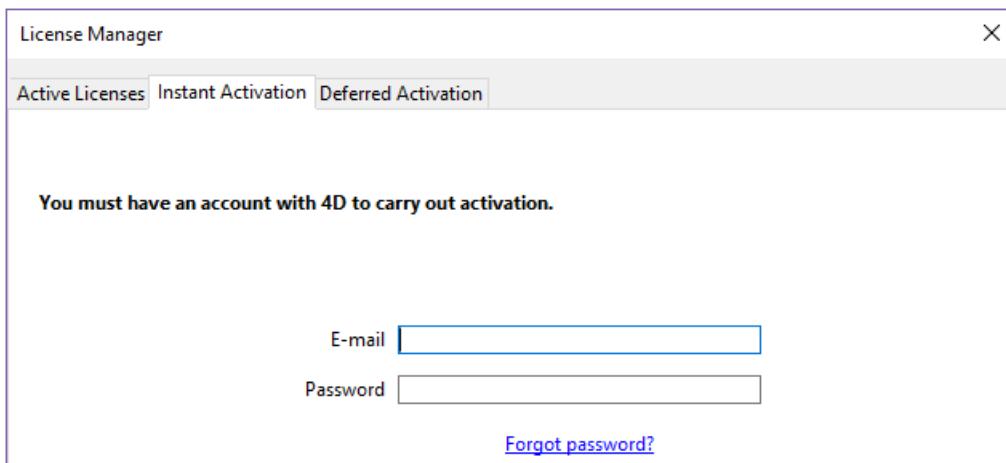


4D vous propose trois modes d'activation. L'activation immédiate est recommandée.

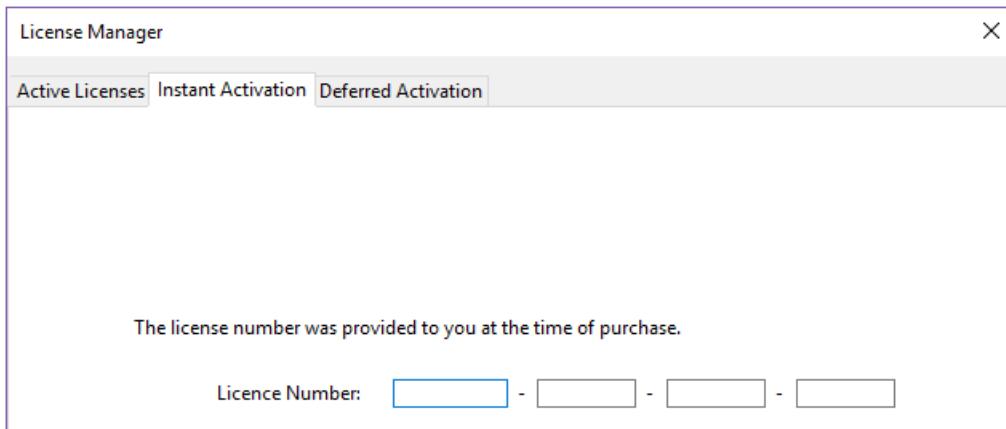
Activation immédiate

Saisissez votre identifiant utilisateur (e-mail ou compte 4D) ainsi que votre mot de passe. Si vous n'avez pas encore de compte client chez 4D, vous devez en créer un à l'adresse suivante :

<https://account.4d.com/us/login.shtml>



Entrez ensuite le numéro de licence du produit à activer. Ce numéro se trouve dans l'e-mail de livraison ou le certificat d'authenticité reçu par courrier.



Activation différée

Si vous ne pouvez pas utiliser [l'activation immédiate](#) parce que votre ordinateur n'a pas d'accès Internet, vous pouvez effectuer une activation différée comme décrit dans les étapes suivantes.

1. Dans la fenêtre du Gestionnaire de licences de 4D accessible depuis le menu Aide, sélectionnez l'onglet Activation différée.
2. Entrez votre Numéro de licence ainsi que votre adresse E-mail, puis cliquez sur [Générer le fichier...](#) afin de créer le fichier d'ID (*reg.txt*).

License Manager X

Active Licenses Instant Activation Deferred Activation

Step 1 out of 3

I want to generate an ID file that I will send to 4D in order to get an activation key in return.

→ Licence Number: - - -

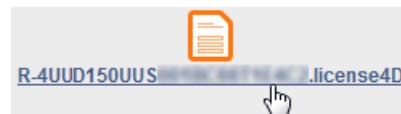
→ E-mail (mandatory):

→ Generate file...

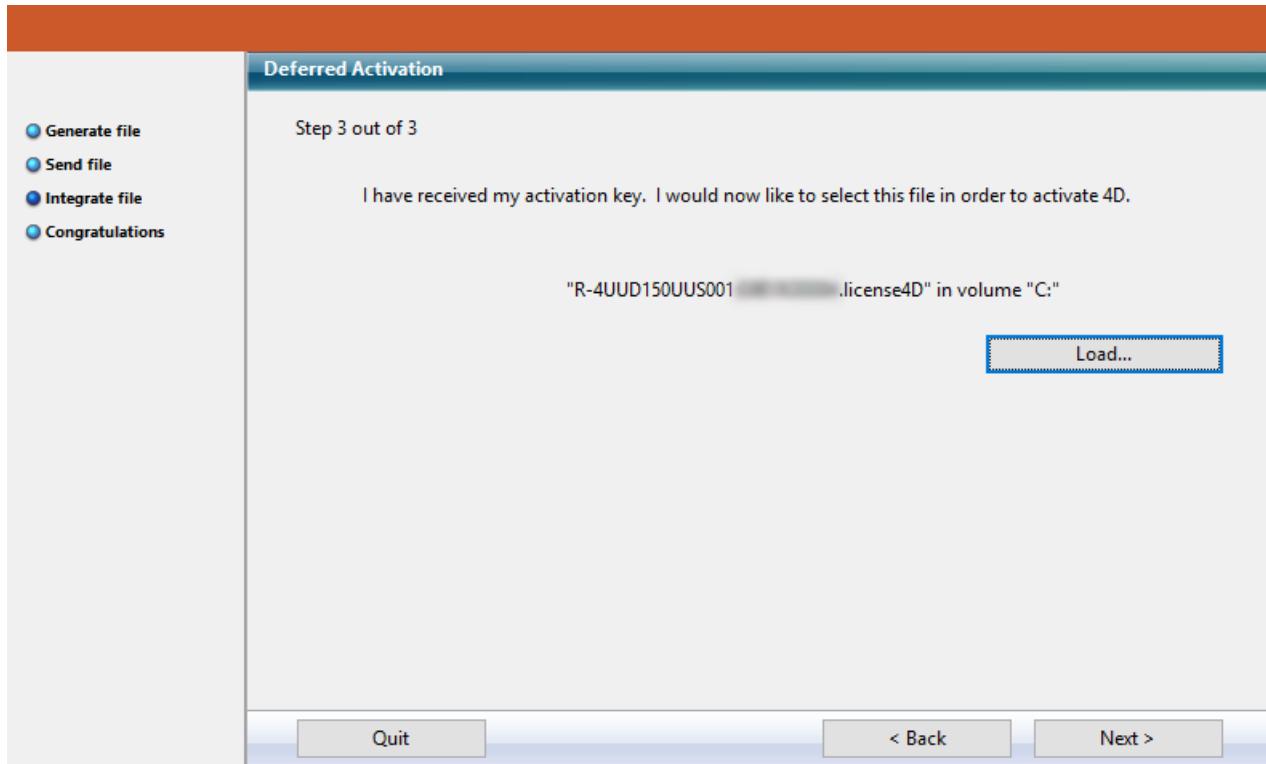
I have received my activation key. I would now like to select this key in order to activate 4D.

[< Back](#) [Next >](#) [Done](#)

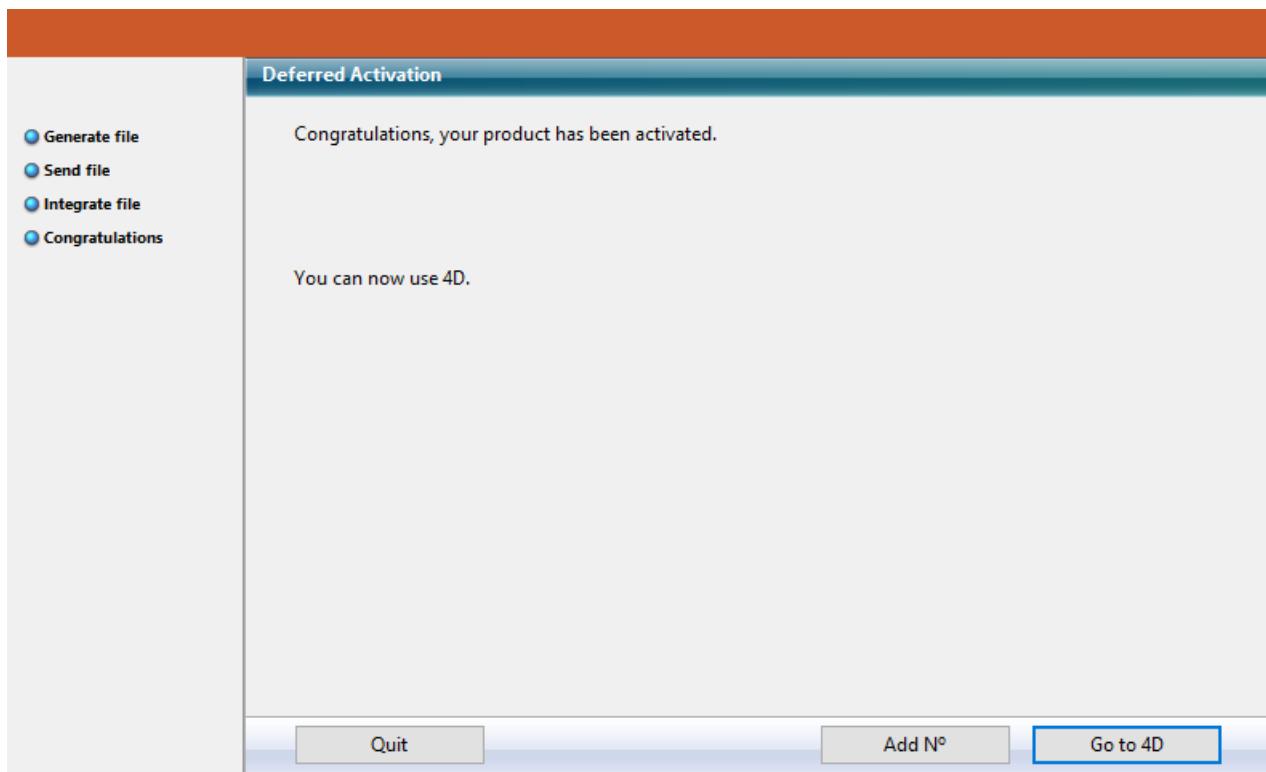
3. Enregistrez le fichier *reg.txt* sur un support USB puis connectez ce support à un ordinateur qui a un accès Internet.
4. Depuis la machine qui a un accès Internet, connectez-vous sur <https://activation.4d.com>.
5. Dans la page Web, cliquez sur le bouton Parcourir... et sélectionnez le fichier *reg.txt* généré lors des étapes 3 et 4 ; puis cliquez sur le bouton Activer.
6. Téléchargez le(s) fichier(s) de licence.



7. Enregistrez le ou les fichier(s) *license4d* sur un support partagé et transférez-le(s) sur la machine 4D utilisée lors de l'étape 1.
8. De retour sur la machine avec 4D, toujours dans l'écran Activation différée, cliquez sur le bouton Suivant ; puis cliquez sur le bouton Charger... et sélectionnez un fichier *license4d* depuis le media partagé utilisé à l'étape 7.



Une fois le fichier de licence chargé, cliquez sur le bouton Suivant.



9. Cliquez sur le bouton Ajouter Nº pour ajouter une autre licence. Répétez ces étapes jusqu'à ce que toutes les licences téléchargées à l'étape 6 aient été intégrées.

Votre application 4D est désormais activée.

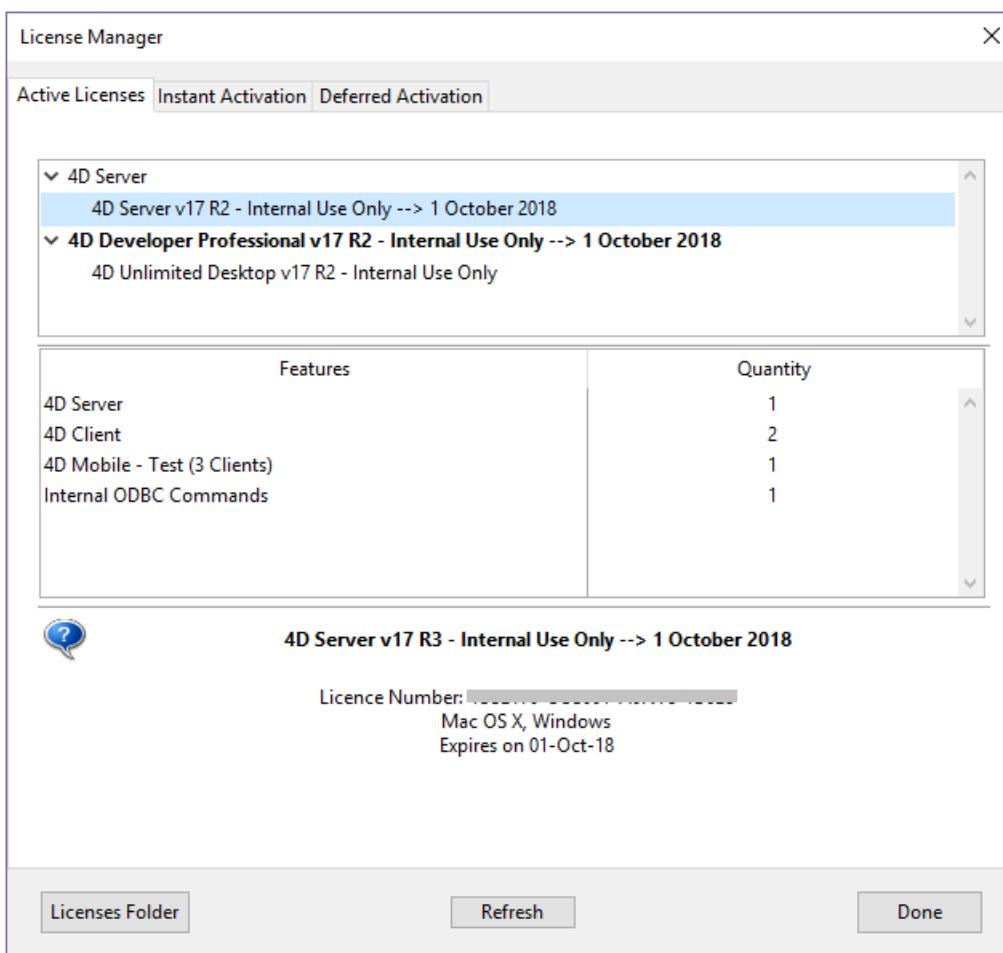
Activation d'urgence

Ce mode permet l'activation exceptionnelle et temporaire de l'application 4D (5 jours maximum) sans connexion au site Internet de 4D. Cette activation ne peut être utilisée qu'une seule fois.

Ajouter des licences

Vous pouvez à tout moment ajouter de nouvelles licences, par exemple pour étendre les capacités de votre application.

Choisissez la commande Gestionnaire de licences... dans le menu Aide de l'application 4D ou 4D Server puis cliquez sur le bouton Actualiser :



Ce bouton vous connecte à notre base clients et active automatiquement toutes les licences nouvelles ou mises à jour liées à la licence courante (la licence courante est affichée en gras dans la liste des Licences actives). Vous devrez simplement saisir vos identifiants 4D (compte et mot de passe). Vous devrez simplement saisir vos identifiants 4D (compte et mot de passe).

- Si vous avez acheté des expansions supplémentaires pour un 4D Server, vous n'avez pas besoin de saisir de numéro -- cliquez simplement sur Actualiser.
- A la première activation d'un 4D Server, vous devez uniquement saisir le numéro du serveur et toutes les licences d'expansion associées sont automatiquement affectées.

Vous pouvez utiliser le bouton Actualiser dans les contextes suivants :

- Lorsque vous avez acquis une expansion supplémentaire et souhaitez l'activer,
- Lorsque vous voulez mettre à jour un numéro de licence temporaire ayant expiré (Partenaires ou évolutions).

4D Online Store

Sur le site web 4D Store, vous pouvez commander, mettre à jour, étendre et gérer vos produits 4D. Vous pouvez vous connecter au store à l'adresse suivante : <https://store.4d.com/fr/> (veuillez sélectionner votre pays).

Cliquez sur Se connecter pour vous identifier à l'aide de votre compte existant ou sur Nouveau compte pour en créer un nouveau, puis suivez les instructions à l'écran.

Gestion des licences

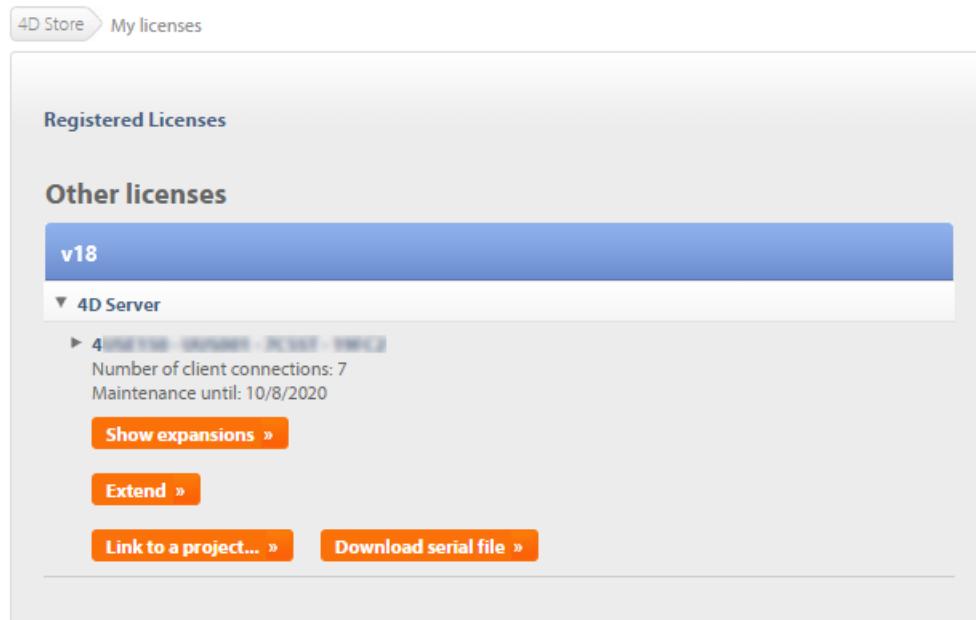
Après vous être identifié, vous pouvez cliquer sur le lien Liste de mes licences en haut de la partie droite de la fenêtre :

MY LICENSES

[License list »](#)
[License Registration »](#)
[Purchase an Upgrade »](#)
[Upgrade Under Maintenance »](#)

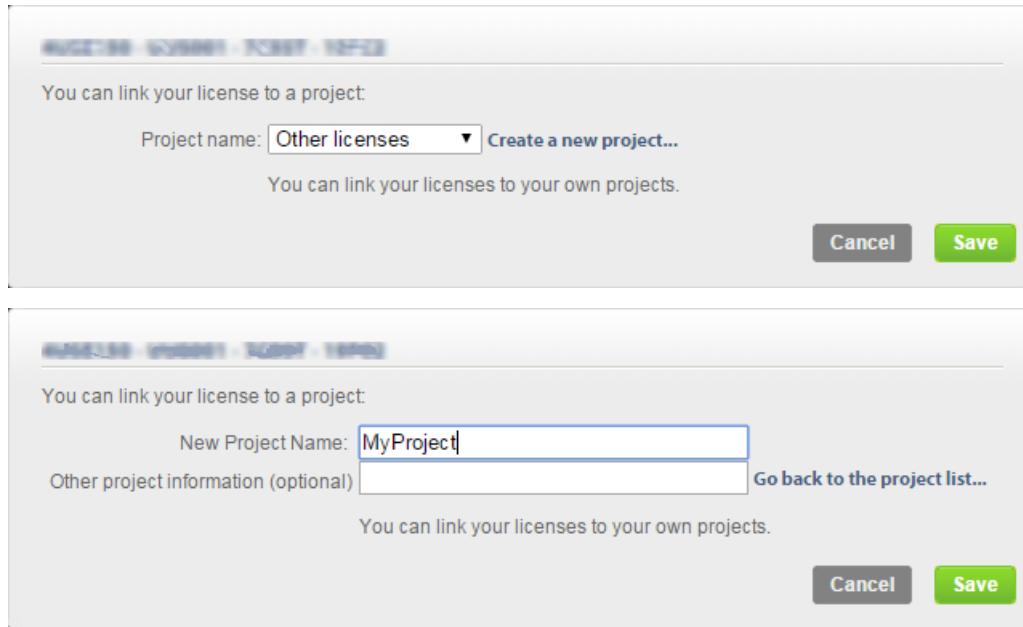
Vous pouvez ensuite gérer vos licences en les affectant à des projets.

Sélectionnez la licence que vous souhaitez dans la liste, puis cliquez sur **Lier à un projet:



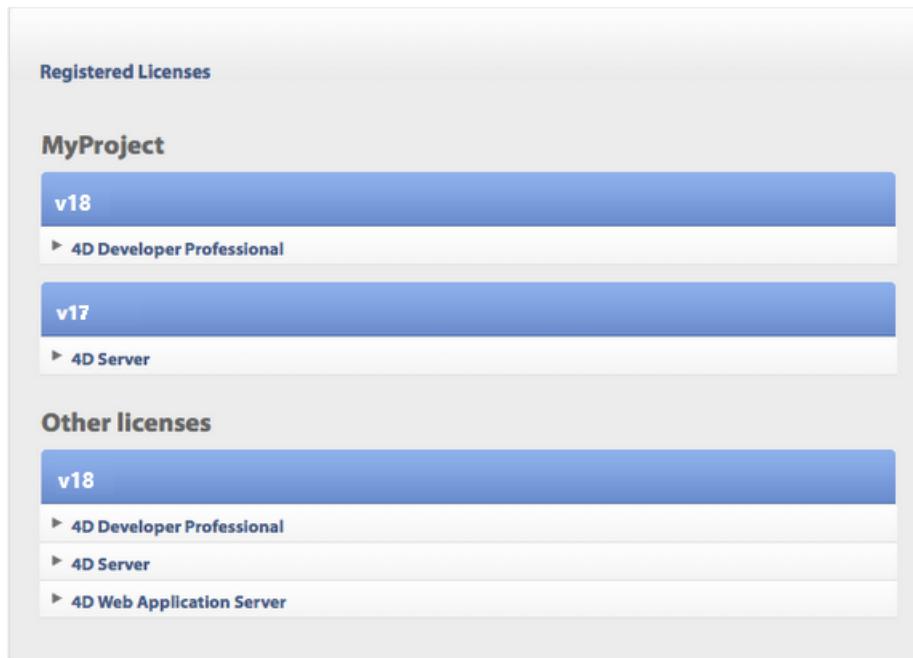
The screenshot shows the 'My licenses' interface. At the top, there's a navigation bar with '4D Store' and 'My licenses'. Below it, a section titled 'Registered Licenses' lists a single entry: 'v18'. Under 'v18', there's a section for '4D Server' with the following details:
Number of client connections: 7
Maintenance until: 10/8/2020
Buttons below this section include 'Show expansions »', 'Extend »', 'Link to a project... »', and 'Download serial file »'.

Vous pouvez sélectionner un projet existant ou créer un nouveau :



The image contains two screenshots of a modal dialog box for linking a license to a project.
The top screenshot shows a dropdown menu labeled 'Project name' with 'Other licenses' selected. There is also a link 'Create a new project...'. Below the dropdown, a message says 'You can link your licenses to your own projects.' At the bottom are 'Cancel' and 'Save' buttons.
The bottom screenshot shows a 'New Project Name' input field containing 'MyProject'. Below it is a field for 'Other project information (optional)'. A link 'Go back to the project list...' is visible. The same message 'You can link your licenses to your own projects.' is present at the bottom, along with 'Cancel' and 'Save' buttons.

Les projets vous permettent d'organiser vos licences comme vous le souhaitez :



Dépannage

En cas d'échec du processus d'installation ou d'activation, veuillez consulter le tableau suivant, présentant les causes de dysfonctionnements les plus fréquentes :

Symptômes	Causes possibles	Solution(s)
Impossible de télécharger le produit depuis le site Internet de 4D	Site Internet indisponible, antivirus, firewall	1- Réessayez ultérieurement OU 2- Désactivez temporairement votre antivirus ou votre firewall.
Impossible d'installer le produit sur le disque (installation refusée).	Droits d'accès utilisateur insuffisants	Ouvrez une session avec des droits d'accès permettant l'installation d'applications (accès administrateur)
Echec de l'activation en ligne	Antivirus, firewall, proxy	1- Désactivez temporairement votre antivirus ou votre firewall OU 2- Utilisez l'activation différée (non disponible avec les licences des versions "R")

Si ces informations ne vous permettent pas de résoudre votre problème, veuillez contacter 4D ou votre distributeur local.

Contacts

Pour toute question relative à l'installation ou l'activation de votre produit, veuillez contacter 4D Sas ou votre distributeur local.

Pour la France :

- Web : <http://www.4d.com/fr/>
- Téléphone : 0892 68 09 97 (0,34 Euro Ttc/Min)

Pour le Royaume-Uni :

- Web: <https://uk.4d.com/4d-technical-support>
- Téléphone : 01625 536178

Gestion des groupes et utilisateurs 4D

Dans les applications multi-utilisateurs, 4D fournit aux utilisateurs certains privilèges d'accès standard et certains pouvoirs. Une fois qu'un système d'utilisateurs et de groupes a été créé, ces privilèges standard prennent effet.

Utilisateurs et groupes dans les projets

Dans les applications projet (fichiers .4DProject ou .4dz), les utilisateurs et groupes 4D peuvent être configurés à la fois en monoposte ou en multi-utilisateurs. Toutefois, le contrôle des accès est effectif uniquement avec 4D Server. Le tableau suivant liste les principales fonctionnalités des utilisateurs et groupes ainsi que leur disponibilité :

	4D (monoposte)	4D Server
Ajouter/modifier des utilisateurs et groupes	oui	oui
Affecter l'accès des utilisateurs/groupes aux serveurs	oui	oui
Identification de l'utilisateur	non (tous les utilisateurs sont des Super_Utilisateur)	oui
Contrôle d'accès une fois qu'un mot de passe a été affecté au Super_Utilisateur	non (tous les accès sont accordés au Super_Utilisateur)	oui

Pour obtenir des informations sur l'identification des utilisateurs et le contrôle des accès pour les déploiements monoposte, reportez-vous à [ce paragraphe](#).

Super_Utilisateur et Administrateur

L'utilisateur le plus puissant est le Super_Utilisateur. Aucune partie de l'application n'est inaccessible au Super_Utilisateur. Le Super_Utilisateur peut :

- accéder à tous les serveurs de l'application sans restrictions,
- créer des utilisateurs et des groupes,
- affecter des privilèges d'accès aux groupes,
- utiliser le mode Développement. En monoposte, les droits d'accès du Super_Utilisateur sont toujours utilisés. En mode client/serveur, l'affectation d'un mot de passe au Super_Utilisateur affiche la boîte de dialogue de connexion. L'accès au mode Développement est en lecture seule.

Après le Super_Utilisateur, le second plus puissant utilisateur est l'Administrateur, qui est en général responsable de la gestion du système d'accès et des fonctionnalités d'administration.

L'Administrateur peut :

- créer des utilisateurs et des groupes,
- accéder au moniteur et à la fenêtre d'administration de 4D Server
- accéder à la fenêtre CSM pour gérer les sauvegardes, restitutions ou le serveur.

L'Administrateur ne peut pas :

- modifier l'utilisateur Super_Utilisateur
- par défaut, accéder à des objets protégés de l'application. En particulier, l'Administrateur ne peut pas accéder au mode Développement s'il est restreint. L'Administrateur doit faire partie d'un ou plusieurs groupes pour avoir des privilèges d'accès dans l'application. Il est placé dans tous les nouveaux groupes, mais vous pouvez cependant l'exclure de ces groupes.

Par défaut, le Super_Utilisateur et l'Administrateur se trouvent dans toutes les applications. Dans la [boîte de dialogue de gestion des utilisateurs](#), les icônes du Super_Utilisateur et de l'Administrateur ont des icônes respectivement rouge et

verte :

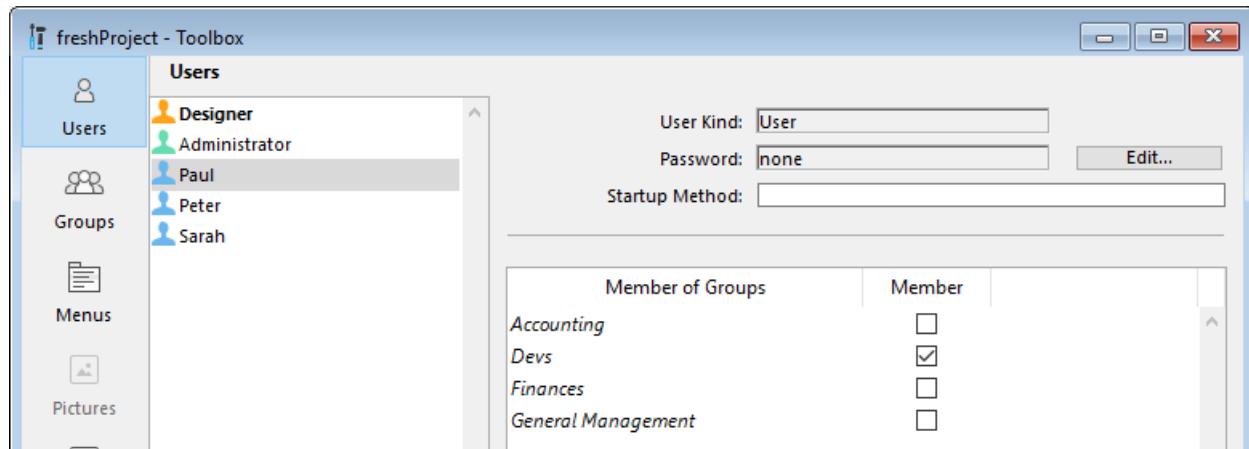
- Icône du Super_Utilisateur :
- Icône de l'Administrateur :

Vous pouvez renommer les utilisateurs Super_Utilisateur et Administrateur. Dans le langage, Super_Utilisateur porte toujours l'identifiant 1 et l'Administrateur l'identifiant 2.

Le Super_Utilisateur et l'Administrateur peuvent créer chacun 16 000 groupes et 16 000 utilisateurs.

Éditeur d'utilisateurs

L'éditeur des utilisateurs se trouve dans la boîte à outils de 4D.



L'éditeur des utilisateurs et des groupes peut être affiché au moment de l'exécution à l'aide de la commande [EDIT ACCESS](#). Toute la configuration des utilisateurs et des groupes peut également être modifiée pendant l'exécution de l'application à l'aide des commandes du langage 4D du thème [Utilisateurs et groupes](#).

Ajouter et modifier des utilisateurs

Vous utilisez l'éditeur d'utilisateurs pour créer des comptes utilisateurs, définir leurs propriétés et leur appartenance aux différents groupes.

Pour ajouter un utilisateur depuis la boîte à outils :

1. Sélectionnez Boîte à outils > Utilisateurs dans le menu Développement ou cliquez sur le bouton Boîte outils de la barre d'outils de 4D. 4D affiche la fenêtre d'édition des utilisateurs.

La liste des utilisateurs affiche tous les utilisateurs, y compris [le Super_Utilisateur et l'Administrateur](#).

2. Cliquez sur le bouton d'ajout situé au-dessous de la Liste des utilisateurs. OU Cliquez avec le bouton droit de la souris dans la Liste des utilisateurs et choisissez la commande Ajouter ou Dupliquer dans le menu contextuel.

La commande Dupliquer permet de créer rapidement plusieurs utilisateurs ayant des caractéristiques communes.

4D ajoute un nouvel utilisateur à la liste, nommé par défaut "Nouvel utilisateurN".

3. Saisissez le nom du nouvel utilisateur. Ce nom sera utilisé par l'utilisateur pour ouvrir l'application. Vous pouvez renommer un utilisateur à tout moment en utilisant la commande Renommer du menu contextuel, ou en utilisant la combinaison Alt+clic (Windows) ou Option+clic (macOS) ou en cliquant deux fois sur un nom.
4. Pour saisir le mot de passe de l'utilisateur, cliquez sur le bouton Modifier... dans la zone des propriétés de l'utilisateur et saisissez deux fois le mot de passe dans la boîte de dialogue. Vous pouvez saisir jusqu'à 15 caractères alphanumériques. L'éditeur de mots de passe tient compte de la casse des caractères (majuscules ou minuscules).

Les utilisateurs peuvent modifier leur mot de passe à tout moment en fonction des options de la page Sécurité

des propriétés de la structure, ou à l'aide de la commande `CHANGE PASSWORD`.

- Définissez le ou les groupe(s) d'appartenance de l'utilisateur à l'aide du tableau "Membre des groupes". Vous pouvez ajouter l'utilisateur sélectionné à un groupe en cochant l'option correspondante dans la colonne Membre.

L'appartenance des utilisateurs aux groupes peut également être définie par groupe dans la [page Groupes](#).

Supprimer un utilisateur

Pour supprimer un utilisateur, sélectionnez-le puis cliquez sur le bouton de suppression ou utilisez la commande Supprimer du menu contextuel.



Les utilisateurs supprimés n'apparaissent plus dans la liste de l'éditeur d'utilisateurs. A noter que les numéros des utilisateurs supprimés peuvent être réattribués lors de la création de nouveaux comptes.

Propriétés des utilisateurs

- Le champ Type d'utilisateur : le champ Type d'utilisateur contient "Super_Utilisateur", "Administrateur", ou (pour tous les autres utilisateurs) "Utilisateur".
- Méthodes de démarrage : Nom d'une méthode associée qui sera automatiquement associée lorsque l'utilisateur ouvre l'application (facultatif). Cette méthode peut être utilisée par exemple pour charger les préférences utilisateur.

Éditeur de groupes

L'éditeur de groupes se trouve dans la boîte à outils de 4D.

Configurer des groupes

Vous utilisez l'éditeur de groupes pour définir les éléments qu'ils contiennent (utilisateurs et/ou autres groupes) et pour répartir les accès aux plug-ins.

Attention, une fois créé, un groupe ne peut pas être supprimé. Si vous souhaitez désactiver un groupe, il vous suffit de retirer tous les utilisateurs qu'il contient.

Pour créer un groupe :

- Sélectionnez Boîte à outils > Groupes dans le menu Développement ou cliquez sur le bouton Boîte outils de la barre d'outils de 4D puis cliquez sur le bouton Groupes. 4D affiche la fenêtre d'édition des groupes. La liste des groupes affiche tous les groupes du projet d'application.
- Cliquez sur le bouton



situé en-dessous de la liste des groupes.

OU

Faites un clic droit sur la liste de groupes et sélectionnez la commande Ajouter ou Dupliquer dans le menu contextuel.

La commande Dupliquer permet de créer rapidement plusieurs groupes ayant des caractéristiques communes.

4D ajoute un nouveau groupe à la liste, nommé par défaut "Nouveau groupe1".

- Saisissez le nom du nouveau groupe. Le nom du groupe peut avoir une longueur maximale de 15 caractères. Vous pouvez renommer un groupe à tout moment en utilisant la commande Renommer du menu contextuel, ou en utilisant la combinaison Alt+clic (Windows) ou Option+clic (macOS) ou en cliquant deux fois sur un nom.

Placer des utilisateurs ou des groupes dans des groupes

Vous pouvez placer tout utilisateur ou tout groupe dans un groupe et vous pouvez aussi le placer dans plusieurs groupes. Il n'est pas obligatoire de placer un utilisateur dans un groupe.

Pour placer un utilisateur ou un groupe dans un groupe, il suffit de sélectionner le groupe dans la liste puis de cocher l'option "Membre" pour chaque utilisateur ou groupe dans la zone d'attribution des membres :

	User / Group	Member
	Administrator	<input checked="" type="checkbox"/>
	Designer	<input type="checkbox"/>
	New user	<input type="checkbox"/>
	Paul	<input type="checkbox"/>
	Peter	<input checked="" type="checkbox"/>
	Sarah	<input type="checkbox"/>
	Finances	<input checked="" type="checkbox"/>
	General Management	<input checked="" type="checkbox"/>
	Devs	<input type="checkbox"/>
	Admins	<input type="checkbox"/>

Si vous cochez le nom d'un utilisateur, l'utilisateur est ajouté au groupe. Si vous cochez un nom de groupe, tous les utilisateurs du groupe sont ajoutés au nouveau groupe. L'utilisateur ou le groupe affilié dispose alors des priviléges d'accès affectés au nouveau groupe.

Placer des groupes dans d'autres groupes permet de créer une hiérarchie d'utilisateurs. Les utilisateurs d'un groupe placé dans un autre groupe disposent des autorisations d'accès des deux groupes. Reportez-vous au paragraphe [Un schéma d'accès hiérarchique](#) ci-dessous.

Pour supprimer un utilisateur ou un groupe d'un autre groupe, il suffit de désélectionner l'option correspondante dans la liste des membres.

Affecter un groupe à un plug-in ou à un serveur

Pour affecter un groupe à un plug-in, il suffit de cocher l'option correspondante. Les plug-ins comprennent tous les plug-ins de 4D ainsi que tout plug-in développés par une société tierce.

Répartir les accès aux plug-ins vous permet de contrôler l'utilisation des licences dont vous disposez pour ces plug-ins. Tout utilisateur n'appartenant pas au groupe d'accès à un plug-in ne pourra pas charger ce plug-in.

Les licences utilisées demeurent associées aux comptes utilisateurs 4D dans le groupe, durant toute la session 4D.

La zone "Plug-ins" de la page Groupes de la boîte à outils liste tous les plug-ins chargés par l'application 4D. Pour affecter un groupe à un plug-in, il suffit de cocher l'option correspondante.

Plug-in	Access
4D Client Web Server	<input type="checkbox"/>
4D Client SOAP Server	<input type="checkbox"/>
4D Write PRO	<input checked="" type="checkbox"/>
4D View PRO	<input type="checkbox"/>

Les lignes 4D Client Web Server et 4D Client SOAP Server permettent contrôler la possibilité de publication Web et SOAP (Web Services) de chaque 4D en mode distant. En effet, ces licences sont considérées par 4D Server comme des licences de plug-ins. Ainsi, comme pour un plug-in, vous pouvez restreindre le droit d'utiliser ces licences à un groupe d'utilisateurs spécifique.

Un schéma d'accès hiérarchique

Le meilleur moyen d'assurer la sécurité de votre application et de proposer différents niveaux d'accès aux utilisateurs est d'utiliser un schéma hiérarchique des accès. Les utilisateurs peuvent être affectés à différents groupes et les groupes peuvent être hiérarchisés pour créer des niveaux de droits d'accès. Cette section décrit différentes approches de ce thème.

Dans cet exemple, un utilisateur appartient à l'un des trois groupes définis suivant son niveau de responsabilité. Les utilisateurs du groupe Comptabilité sont responsables de la saisie de données. Les utilisateurs du groupe Dir. finance

sont responsables de la mise à jour des données, comme la mise à jour d'enregistrements ou la suppression d'enregistrements obsolètes. Les utilisateurs du groupe Direction générale sont responsables de l'analyse de données, ce qui inclut la réalisation de recherches et l'impression d'états.

Les groupes sont hiérarchisés afin que les priviléges soient correctement affectés aux utilisateurs de chacun des groupes.

- Le groupe Direction générale ne contient que les utilisateurs de "haut niveau".

The screenshot shows the 'freshProject - Toolbox' window. On the left, there's a sidebar with icons for Users, Groups, Menus, Pictures, Help Tips, Lists, Style Sheets, Filters, and Resources. The 'Groups' icon is highlighted. In the main area, under 'Groups', the 'General Management' group is selected. To the right, there are two tables. The top table, titled 'User / Group', lists users and their group membership. The bottom table, titled 'Plug-in', lists plug-ins and their access levels.

User / Group	Member
Administrator	<input checked="" type="checkbox"/>
Designer	<input type="checkbox"/>
Paul	<input type="checkbox"/>
Peter	<input type="checkbox"/>
Sarah	<input type="checkbox"/>
Accounting	<input type="checkbox"/>
Finances	<input type="checkbox"/>
Devs	<input type="checkbox"/>
Admins	<input type="checkbox"/>

Plug-in	Access
4D Client Web Server	<input type="checkbox"/>
4D Client SOAP Server	<input type="checkbox"/>
4D Write PRO	<input type="checkbox"/>
4D View PRO	<input type="checkbox"/>

- Le groupe Dir. finance contient les utilisateurs du groupe Direction générale.

The screenshot shows the 'Groups' section of the 4D Toolbox. On the left, a sidebar lists various tools: Users, Groups (selected), Menus, Pictures, Help Tips, Lists, Style Sheets, Filters, and Resources. Below these are buttons for adding (+), removing (-), and filtering (*).

Groups:

- Accounting
- Admins
- Devs
- Finances** (highlighted)
- General Management

User / Group Member:

User / Group	Member
Administrator	<input checked="" type="checkbox"/>
Designer	<input type="checkbox"/>
Paul	<input type="checkbox"/>
Peter	<input type="checkbox"/>
Sarah	<input type="checkbox"/>
Accounting	<input type="checkbox"/>
General Management	<input checked="" type="checkbox"/>
Devs	<input type="checkbox"/>
Admins	<input type="checkbox"/>

Plug-in Access:

Plug-in	Access
4D Client Web Server	<input type="checkbox"/>
4D Client SOAP Server	<input type="checkbox"/>
4D Write PRO	<input type="checkbox"/>
4D View PRO	<input type="checkbox"/>

- Le groupe Comptabilité contient des opérateurs de saisie mais aussi les utilisateurs des groupes Dir. finance et donc Direction générale.

This screenshot is identical to the one above, showing the same groups, member assignments, and plug-in access settings in the 4D Toolbox Groups panel.

Vous pouvez ensuite décider des priviléges affectés à chaque groupe suivant le niveau de responsabilité des utilisateurs qu'il contient.

Un tel système hiérarchique rend aisée l'affectation d'un utilisateur à un groupe. Il suffit de placer chaque utilisateur

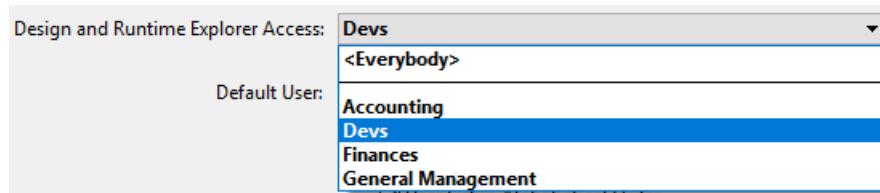
dans un groupe et d'utiliser la hiérarchie des groupes pour déterminer les accès.

Définition des accès aux groupes

Les groupes se voient attribuer des priviléges d'accès à des parties ou des fonctionnalités spécifiques de l'application :

- Accès à l'environnement de Développement et à l'Explorateur d'exécution,
- Serveur HTTP,
- Serveur REST,
- Serveur SQL..

Ces accès sont définis dans la boîte de dialogue Paramètres. L'exemple suivant présente les droits d'accès à l'explorateur d'exécution et au Développement assignés au groupe "Devs" :



Vous utilisez également des groupes pour [distribuer les licences disponibles](#). Cette distribution est définie dans l'éditeur Groupes.

Fichier directory.json

Les utilisateurs, les groupes ainsi que leurs droits d'accès sont stockés dans un fichier spécifique du projet nommé directory.json.

Ce fichier peut être stocké aux emplacements suivants, en fonction de vos besoins :

- Si vous souhaitez utiliser le même répertoire pour tous les fichiers de données (ou si vous utilisez un seul fichier de données), stockez le fichier directory.json dans le dossier des paramètres utilisateur, c'est-à-dire dans le dossier "Settings" [au même niveau que le dossier "Project"](#) (emplacement par défaut).
- Si vous souhaitez utiliser un fichier répertoire spécifique par fichier de données, stockez le fichier directory.json dans le dossier des paramètres des données, c'est-à-dire dans le dossier "[Settings](#)" du dossier "Data". Si un fichier directory.json se trouve à cet emplacement, il est prioritaire par rapport au fichier du dossier Settings utilisateur. Cette configuration personnalisée/locale des utilisateurs et des groupes ne sera pas modifiée par une mise à niveau de l'application.

To allow for safe changes of passwords and group memberships in a deployed environment, you can include your directory.json file in the server application during the build, using the [corresponding build application option](#).

Page informations

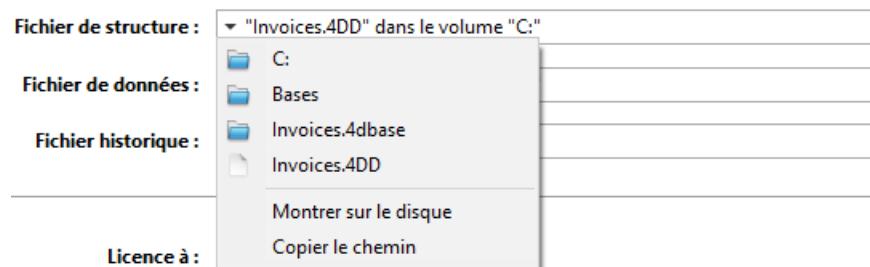
La page "Informations" fournit diverses informations relatives à l'environnement 4D et l'environnement système, la base de données et les fichiers de l'application. Chaque page d'information est accessible via des onglets situés dans la partie supérieure de la fenêtre.

Programme

Cette page affiche le nom, la version et l'emplacement de l'application ainsi que du dossier 4D actif (pour plus d'informations sur le dossier 4D actif, reportez-vous à la description de la commande `Get 4D folder` du manuel *Langage 4D*).

La partie centrale de la fenêtre affiche le nom et l'emplacement du projet et des fichiers de données ainsi que, le cas échéant, du fichier d'historique. La partie inférieure de la fenêtre indique le nom du propriétaire de la licence 4D, le type de licence et le nom de l'utilisateur 4D courant.

- Affichage et sélection des chemins d'accès : dans la page Programme, les chemins d'accès sont affichés sous forme de pop up menus contenant l'enchaînement des dossiers à partir du disque :



Si vous sélectionnez un élément du menu (disque ou dossier), il s'affiche dans une nouvelle fenêtre système. La commande Copier copie le chemin d'accès complet dans le Presse-papiers sous forme de texte, et en utilisant les séparateurs de la plate-forme courante.

- Dossier "Licenses" Le bouton Dossier "Licenses" permet d'afficher le contenu du dossier Licenses actif dans une nouvelle fenêtre système. Tous les fichiers de licence installés dans votre environnement 4D sont regroupés dans ce dossier, placé sur votre disque dur. Lorsqu'ils sont ouverts avec un navigateur Web, ces fichiers affichent des informations relatives aux licences qu'ils contiennent et à leurs caractéristiques. L'emplacement du dossier "Licenses" peut varier en fonction de la version ou de la langue de votre système d'exploitation. For more information about the location of this folder, refer to the `Get 4D folder` command. Note : Vous pouvez également accéder à ce dossier depuis la boîte de dialogue "Mise à jour des licences" (accessible depuis le menu Aide).

Tables

Cette page propose une vue d'ensemble de la base :

Centre de Sécurité et de Maintenance

Informations

Programme Tables Données Structure

No.	Tables	Enregistrem...	Champs	Index	Cryptable	Cryptée	Taille table adresse:
1	SOCIETES	600	18	1	<input type="checkbox"/>	<input type="checkbox"/>	600
2	INVOICES	12 526	14	4	<input type="checkbox"/>	<input type="checkbox"/>	12 526
3	INVOICE_LINES	68 903	11	3	<input type="checkbox"/>	<input type="checkbox"/>	68 903
4	PRODUCTS	721	6	2	<input type="checkbox"/>	<input type="checkbox"/>	721
5	SETTINGS	6	2	0	<input type="checkbox"/>	<input type="checkbox"/>	6
6	Process_Temps	0	4	1	<input type="checkbox"/>	<input type="checkbox"/>	0
7	ECHANTILLONS	0	4	1	<input type="checkbox"/>	<input type="checkbox"/>	0
8	Employes	0	10	2	<input checked="" type="checkbox"/>	<input type="checkbox"/>	0
9	manager	0	2	1	<input type="checkbox"/>	<input type="checkbox"/>	0
Total		9	82 756	71	15	1	0
							82 756

Les informations présentes dans cette page sont disponibles en mode standard et en mode maintenance.

La page liste toutes les tables de la base (y compris les tables invisibles) ainsi que leurs caractéristiques :

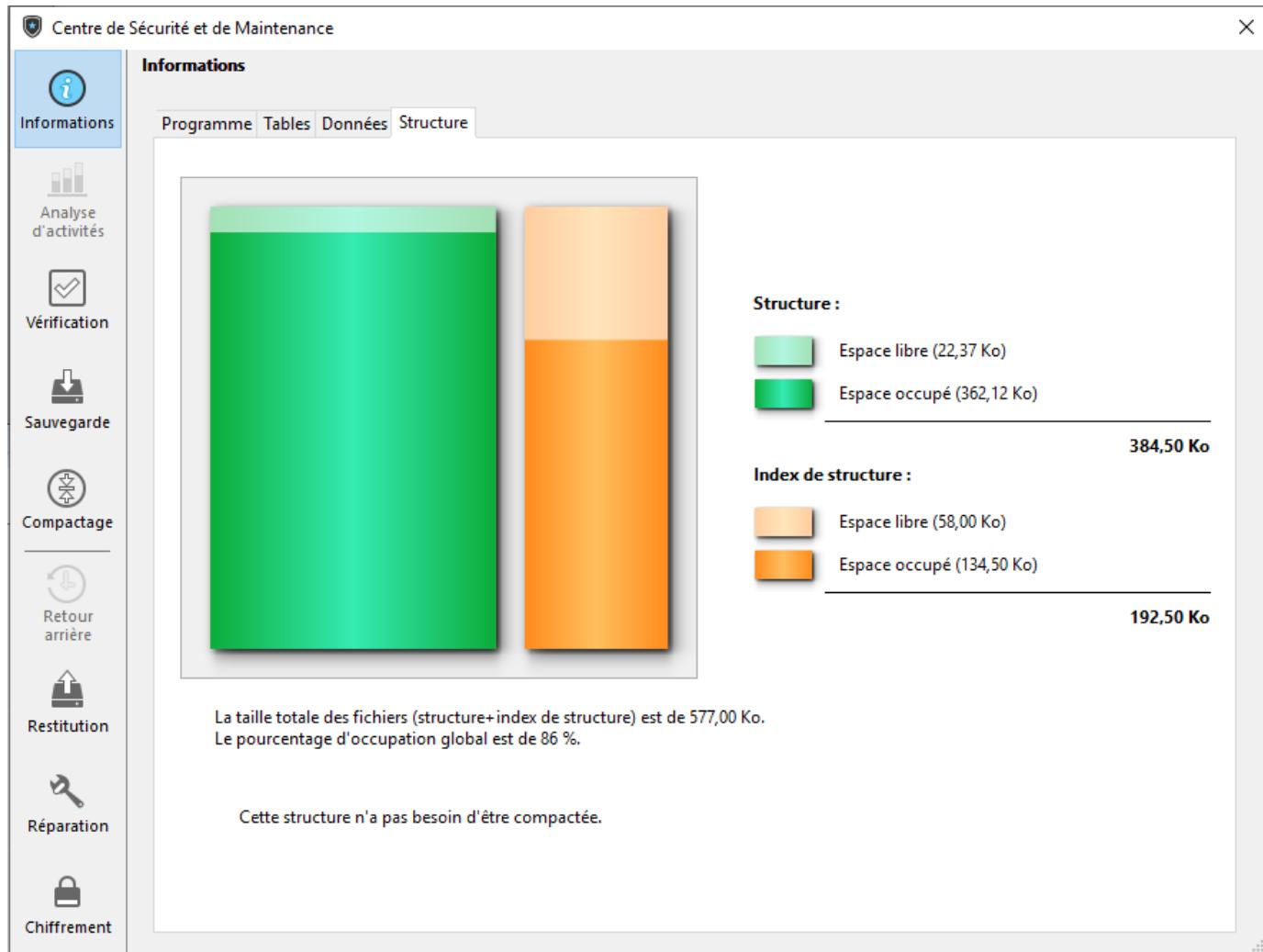
- No. : Numéros internes des tables.
- Tables : Noms des tables. Les noms des tables supprimées sont affichés entre parenthèses (lorsqu'elles sont dans la corbeille).
- Enregistrements : Nombre total d'enregistrements de chaque table. Si un enregistrement est endommagé ou ne peut pas être lu, le mot *Erreur* est affiché à la place du total. Dans ce cas, vous devez envisager d'utiliser les outils de vérification et de réparation.
- Champs : Nombre de champs dans la table. Les champs invisibles sont comptés, en revanche les champs supprimés ne le sont pas.
- Index : Nombre d'index de tout type associés à la table
- Chiffrable : Si l'attribut Chiffrable est coché, il est sélectionné pour la table au niveau de la structure (voir le paragraphe "Chiffrable" dans le manuel Développement).
- Chiffrée : Si cet attribut est coché, les enregistrements de la table sont chiffrés dans le fichier de données. *Note : Toute incohérence entre les options Chiffrable et Chiffrée nécessite une vérification de l'état de chiffrement du fichier de données dans la Page Chiffrement du CSM.*
- Taille table adresses : Taille de la table d'adresses pour chaque table. La table d'adresses est une table interne qui stocke un élément par enregistrement créé dans la table. Elle établit le lien entre les enregistrements et leur adresse physique. Pour des raisons de performance, elle n'est pas redimensionnée lorsque des enregistrements sont supprimés. Sa taille peut donc être différente du nombre d'enregistrements actuel de la table. À noter que si cette différence est significative, il peut être intéressant de compacter les données avec l'option "Compacter la table d'adresses" afin d'optimiser la taille de la table d'adresses (voir [Page Compactage](#)). *Note : Des différences entre la taille de la table d'adresses et le nombre d'enregistrements peuvent également résulter d'un incident durant l'écriture du cache sur le disque. *

Données

La page Données fournit des informations sur l'espace de stockage libre et occupé dans les fichiers de données et la structure de la base.

Cette page n'est pas accessible en mode maintenance

Ces informations sont fournies sous forme de valeurs en octets et sont également représentées sous forme graphique :



La page Données ne tient pas compte de la taille des données éventuellement stockées à l'extérieur du fichier de données (cf. section Stockage externe des données).

Des fichiers trop fragmentés réduisent les performances du disque dur et donc de la base. Si le taux d'occupation est trop faible, 4D vous le signale par une icône d'avertissement (qui apparaît dans le bouton Informations et dans l'onglet du type de fichier correspondant) et indique qu'un compactage est requis :



Une icône d'avertissement est également affichée sur le bouton de la Page [Compactage](#) :



Page Analyse d'activités

La page Analyse d'activités du CSM permet de visualiser le contenu du fichier d'historique courant. This function is useful for parsing the use of an application or detecting the operation(s) that caused errors or malfunctions. In the case of an application in client-server mode, it allows verifying operations performed by each client machine.

Il est également possible de revenir en arrière parmi les opérations effectuées sur les données de la base. Pour plus d'informations sur ce point, reportez-vous à la section [Page Retour](#) arrière.

Maintenance and Security Center

Activity analysis

The list below shows all the performed operations recorded in the log file since the last backup.

Operation	Action	Table	Primary...	Process	Size	Date	Hour	System User	4D User	Values	Record
792	Addition	Parts	198	26	50	16/12/2019	16:37	aschmitt	Main user	198 ;	197
793	Sequence	Invoices		26		16/12/2019	16:37	aschmitt	Main user	200	
794	Addition	Invoices	199	26	54	16/12/2019	16:37	aschmitt	Main user	199 ; ;	198
795	Sequence	Parts		26		16/12/2019	16:37	aschmitt	Main user	200	
796	Addition	Parts	199	26	52	16/12/2019	16:37	aschmitt	Main user	199 ;	198
797	Sequence	Invoices		26		16/12/2019	16:37	aschmitt	Main user	201	
798	Addition	Invoices	200	26	54	16/12/2019	16:37	aschmitt	Main user	200 ; ;	199
799	Sequence	Parts		26		16/12/2019	16:37	aschmitt	Main user	201	
800	Addition	Parts	200	26	50	16/12/2019	16:37	aschmitt	Main user	200 ;	199
801	Sequence	Invoices		26		16/12/2019	16:37	aschmitt	Main user	202	
802	Addition	Invoices	201	26	52	16/12/2019	16:37	aschmitt	Main user	201 ; ;	200
803	Sequence	Parts		26		16/12/2019	16:37	aschmitt	Main user	202	
804	Addition	Parts	201	26	48	16/12/2019	16:37	aschmitt	Main user	201 ;	200
805	Sequence	Invoices		26		16/12/2019	16:37	aschmitt	Main user	203	
806	Addition	Invoices	202	26	54	16/12/2019	16:37	aschmitt	Main user	202 ; ;	201
807	Sequence	Parts		26		16/12/2019	16:37	aschmitt	Main user	203	
808	Addition	Parts	202	26	50	16/12/2019	16:37	aschmitt	Main user	202 ;	201
809	Sequence	Invoices		26		16/12/2019	16:37	aschmitt	Main user	204	
810	Addition	Invoices	203	26	52	16/12/2019	16:37	aschmitt	Main user	203 ; ;	202
811	Sequence	Parts		26		16/12/2019	16:37	aschmitt	Main user	204	
812	Addition	Parts	203	26	48	16/12/2019	16:37	aschmitt	Main user	203 ;	202

Analyze Browse Export...

Chaque opération élémentaire enregistrée dans le fichier d'historique apparaît sous forme d'une ligne. Les colonnes fournissent diverses informations concernant l'opération. Vous pouvez réorganiser les colonnes comme vous le souhaitez en cliquant sur leur en-tête.

Les informations affichées permettent d'identifier la source et le contexte de chaque opération :

- Opération : numéro de séquence de l'opération dans le fichier d'historique.
- Action : type d'opération effectuée. Cette colonne peut contenir les opérations suivantes :
 - Ouverture du fichier de données : ouverture d'un fichier de données.
 - Fermeture du fichier de données : fermeture du fichier de données ouvert.
 - Création d'un contexte : création d'un process définissant un contexte d'exécution.
 - Fermeture d'un contexte : fermeture d'un process.
 - Ajout : création et stockage d'un enregistrement.
 - Ajout d'un BLOB : stockage d'un BLOB dans un champ BLOB.
 - Suppression : suppression d'un enregistrement.
 - Modification : modification d'un enregistrement.

- Début de transaction : transaction démarrée.
- Validation de transaction : transaction validée.
- Annulation de transaction : transaction annulée.
- Contexte de mise à jour : modification des données supplémentaires (ex : un appel à `CHANGE CURRENT USER` ou `SET USER ALIAS`).
- Table : table à laquelle appartient l'enregistrement ajouté/supprimé/modifié ou le BLOB.
- Clé primaire/BLOB : contenu de la clé primaire de l'enregistrement (lorsque la clé primaire est composée de plusieurs champs, les valeurs sont séparées par des points-virgules), ou numéro de séquence du BLOB impliqué dans l'opération.
- Process : numéro interne du process dans lequel l'opération a été effectuée. Ce numéro interne correspond au contexte de l'opération. Taille : taille en octets des données traitées par l'opération.
- Taille : taille (en octets) des données traitées par l'opération.
- Date et Heure : date et heure à laquelle l'opération a été effectuée.
- 4D User: 4D user name of the user that performed the operation. Si un alias est défini pour l'utilisateur, l'alias s'affiche à la place du nom d'utilisateur 4D.
- System User: System name of the user that performed the operation. Si un alias est défini pour l'utilisateur, l'alias s'affiche à la place du nom d'utilisateur 4D.
- Valeurs : valeurs des champs de l'enregistrement en cas d'ajout ou de modification. Les valeurs sont séparées par des ";". Seules les valeurs représentables sous forme alphanumérique sont affichées.
Note : Si la base est chiffrée et si aucune clé de données valide correspondant au fichier d'historique n'a été fournie, les valeurs chiffrées ne sont pas affichées dans cette colonne.
- Enregistrements : Numéro de l'enregistrement.

Click on Analyze to update the contents of the current log file of the selected application (named by default `dataname.journal`). The Browse button can be used to select and open another log file for the application. Le bouton Exporter... vous permet d'exporter le contenu du fichier sous forme de texte.

Page Vérification

Cette page permet de vérifier l'intégrité des données et de la structure. La vérification peut porter sur les enregistrements et/ou les index ainsi que sur les objets du développement (méthodes, formulaires...). La page effectue uniquement une vérification des objets. Si des erreurs sont trouvées et des réparations requises, il vous sera nécessaire d'utiliser la [Page Réparation](#).

Actions

La page comporte quatre boutons d'action permettant un accès direct aux fonctions de vérification.

Lorsque la base est chiffrée, la cohérence des données chiffrées est vérifiée. Si aucune clé de données valide n'a été fournie, une boîte de dialogue s'affiche pour vous demander de saisir une phrase secrète ou la clé des données.

- Vérifier les enregistrements et les index : lance la procédure de vérification globale des données.
- Vérifier uniquement les enregistrements : lance la procédure de vérification des enregistrements uniquement (les index ne sont pas vérifiés).
- Vérifier uniquement les index : lance la procédure de vérification des index uniquement (les enregistrements ne sont pas vérifiés).

La vérification des enregistrements et des index peut également être effectuée en mode détaillé, table par table (cf. paragraphe "Détails" ci-dessous).

Voir le compte rendu

Quelle que soit la vérification demandée, 4D génère un fichier de compte-rendu dans le dossier `Logs` de l'application. Ce fichier liste l'ensemble des vérifications effectuées et signale chaque erreur rencontrée, le cas échéant ([OK] est affiché lorsque la vérification est correcte). Il est créé au format XML et est nommé `NomApplicationVerify_Logaaaa-mm-jj hh-mm-ss.xml`" où :

- `ApplicationName` est le nom du fichier de structure sans extension, par exemple "Factures",
- `aaaa-mm-jj hh-mm-ss` est l'horodatage du fichier, basé sur la date et l'heure système locales au moment du lancement de l'opération de vérification, par exemple "2019-02-11 15-20-45".

Lorsque vous cliquez sur le bouton `Voir le compte rendu`, 4D affiche le fichier de compte-rendu le plus récent dans le navigateur par défaut de l'ordinateur.

Détails

Le bouton `Liste des tables` provoque l'affichage d'une page détaillée permettant de visualiser et de sélectionner les enregistrements et les index à vérifier :

Tables	Action	Status
Employee	Verify indexes	
Records	(7324866 record(s))	
Indexed fields	(1 indexed field(s))	
Employee.ID		
Company	Verify indexes	
Records	(7322302 record(s))	
Indexed fields	(1 indexed field(s))	
Company.ID		
Cities		✓
Records	(2073735 record(s))	

Some records (2 073 735 out of 16 720 903) will be verified

All indexes (2) will be verified

Verify

La désignation des éléments à vérifier permet notamment de gagner du temps lors de la vérification.

La liste principale affiche toutes les tables de la base. Pour chaque table, vous pouvez limiter la vérification aux enregistrements et/ou à chaque index. Cliquez sur l'icône en forme de triangle pour déployer le contenu d'une table ou les index d'un champ et sélectionnez/ désélectionnez les cases à cocher en fonction de vos souhaits. Par défaut, tout est sélectionné. Vous pouvez également utiliser les boutons raccourcis Tout sélectionner, Tout désélectionner, Tous les enregistrements et Tous les index .

Pour chaque ligne de table, la colonne "Action" résume les opérations à effectuer. Lorsque la table est déployée, les lignes "Enregistrements" et "Champs indexés" indiquent le nombre d'éléments concernés.

La colonne "Etat" affiche le statut de la vérification de chaque élément à l'aide de symboles :

	Vérification effectuée, pas de problème
	Vérification effectuée, problèmes rencontrés
	Vérification partielle effectuée
	Vérification non effectuée

Cliquez sur le bouton Vérifier pour lancer la vérification ou sur le bouton Standard pour retourner à la page standard.

Le bouton Voir le compte rendu permet d'afficher le fichier de compte-rendu dans le navigateur par défaut de l'ordinateur (cf. paragraphe [Voir le compte rendu](#) ci-dessus).

La page standard ne tient pas compte des modifications effectuées dans la page détaillée : lorsque vous cliquez sur un bouton de vérification dans la page standard, tous les éléments sont vérifiés. En revanche, les paramétrages effectués dans la page détaillée sont conservés d'une session à l'autre.

Page compactage

Cette page permet d'accéder aux fonctions de compactage du fichier de données et de structure.

Pourquoi compacter ?

Le compactage d'un fichier répond à deux types de besoins :

- Réduction de taille et optimisation des fichiers : les fichiers peuvent comporter des emplacements inutilisés (des "trous"). En effet, lorsque vous supprimez des enregistrements, des formulaires, etc., l'emplacement qu'ils occupaient précédemment dans le fichier devient vacant. 4D réutilise ces emplacements vides lorsque c'est possible, mais la taille des données étant variable, les suppressions ou modifications successives génèrent inévitablement des espaces inutilisables pour le programme. Il en va de même lorsqu'une grande quantité de données vient d'être supprimée : les emplacements vides restent inaccessibles dans le fichier. Le rapport entre la taille du fichier de données et l'espace réellement utilisé pour les données est le taux d'occupation des données. Un taux trop faible peut entraîner, outre un gaspillage de place, une dégradation des performances de la base. La fonction de compactage permet de réorganiser et d'optimiser le stockage des données afin de faire disparaître les "trous". Les zones "Informations" résument les données relatives à la fragmentation des fichiers et suggèrent les opérations à effectuer. Les onglets de la page [Informations](#) du CSM indiquent la fragmentation courante des fichiers de la base.
- Réenregistrement intégral des données en appliquant le formatage courant défini dans le fichier de structure. Ce principe est utile lorsque les données d'une même table ont été stockées dans différents formats, par exemple après un changement dans la structure de la base.

Le compactage n'est disponible qu'en mode maintenance. Si vous tentez d'effectuer cette opération en mode standard, une boîte de dialogue d'alerte vous prévient que l'application va être fermée puis relancée en mode maintenance. Il est possible de compacter un fichier de données non ouvert par l'application (cf. paragraphe [Compacter les enregistrements et les index](#) ci-dessous).

Le compactage standard

Pour démarrer directement le compactage du fichier de données ou de structure, cliquez sur le bouton correspondant dans la fenêtre du CSM.



Le compactage entraînant la duplication du fichier d'origine, le bouton est désactivé si la place sur le disque contenant le fichier est insuffisante.

Cette opération défragmente le fichier principal ainsi que les éventuels fichiers d'index. 4D effectue une copie des fichiers d'origine et les place dans un dossier nommé Replaced Files (Compacting), créé à côté du fichier d'origine. Si vous effectuez plusieurs compactages, un nouveau dossier est créé à chaque fois. Il est nommé "Replaced Files (Compacting)_1", "Replaced Files (Compacting)_2", etc. Vous pouvez modifier le dossier dans lequel les fichiers d'origine sont sauvegardés via le mode avancé.

A l'issue de l'opération, les fichiers défragmentés remplacent automatiquement les fichiers d'origine. L'application est immédiatement opérationnelle sans aucune autre manipulation.

Lorsque la base est chiffrée, le compactage comprend le chiffrement et le déchiffrement et requiert ainsi la clé de chiffrement des données courante. Si aucune clé de chiffrement valide n'a encore été fournie, une boîte de dialogue s'affiche pour vous demander de saisir la phrase secrète ou la clé des données.

Attention : Chaque compactage entraîne la duplication du fichier d'origine et donc l'augmentation de la taille du dossier de l'application. Il est important de tenir compte de ce mécanisme (notamment sous macOS où les applications 4D apparaissent sous forme de packages) pour que l'application ne grossisse pas de façon excessive. Une intervention manuelle à l'intérieur du package peut être utile afin de supprimer les copies des fichiers d'origine.

Voir le compte rendu

Une fois le compactage terminé, 4D génère un fichier de compte-rendu dans le dossier Logs du projet. Ce fichier liste l'ensemble des opérations qui ont été menées. Il est créé au format xml et est nommé *ApplicationName_Compact_Log_yyyy-mm-dd hh-mm-ss.xml*" où :

- *ApplicationName* est le nom du fichier de structure sans extension, par exemple "Factures",
- *aaaa-mm-jj hh-mm-ss* est l'horodatage du fichier, basé sur la date et l'heure système locales au moment du lancement de l'opération de vérification, par exemple "2019-02-11 15-20-45".

Lorsque vous cliquez sur le bouton Voir le compte rendu, 4D affiche le fichier de compte-rendu le plus récent dans le navigateur par défaut de l'ordinateur.

Mode avancé

La page Compactage comporte un bouton Avancé, permettant d'accéder à une page d'options pour le compactage des fichiers de données et de structure.

Compackter les enregistrements et les index

La zone Compackter les enregistrements et les index affiche le chemin d'accès du fichier de données courant ainsi qu'un bouton [...] permettant de désigner un autre fichier de données. Lorsque vous cliquez sur ce bouton, une boîte de dialogue standard d'ouverture de documents s'affiche, vous permettant de désigner le fichier de données à compacter. Vous devez sélectionner un fichier de données compatible avec le fichier de structure ouvert. Une fois la boîte de dialogue validée, le chemin d'accès du fichier à compacter est indiqué dans la fenêtre.

Le second bouton [...] permet de désigner un autre emplacement pour les sauvegardes des fichiers originaux effectuées avant compactage. Cette option permet notamment de compacter des fichiers volumineux en utilisant différents disques.

Forcer la mise à jour des enregistrements

Lorsque cette option est cochée, 4D réécrit chaque enregistrement de chaque table lors de l'opération de compactage, en fonction de sa description en structure. Lorsque l'option n'est pas cochée, 4D réorganise uniquement le stockage des données sur le disque. Cette option est utile dans les cas suivants :

- Lorsque des modifications de types de champs ont été apportées à la structure d'une application après que des données ont été saisies. Par exemple, vous pouvez avoir changé le type d'un champ Entier long en Réel. 4D autorise même des modifications entre des types très différents (avec risques de pertes de données), par exemple un champ Réel peut être changé en Texte et inversement. Dans ce cas, 4D ne convertit pas rétroactivement les données déjà saisies, elles ne sont converties que si les enregistrements sont chargés puis sauvegardés. L'option permet de forcer la conversion de toutes les données.
- Lorsqu'une option de stockage externe des données de type Texte, Image ou BLOB a été modifiée après que des données aient été saisies. Ce cas peut notamment se produire après conversion d'une base en version 4D antérieure à la v13. Comme pour le cas du retypage ci-dessus, 4D ne modifie pas rétroactivement les données déjà saisies. Pour cela, vous pouvez forcer la mise à jour des enregistrements afin d'appliquer le nouveau mode de stockage aux enregistrements déjà saisis.
- Lorsque des champs ou des tables ont été supprimé(e)s. Dans ce cas, le compactage avec mise à jour des enregistrements permet de récupérer la place de ces données supprimées et donc de réduire la taille du fichier.

La sélection de cette option entraîne la mise à jour de tous les index.

Compackter la table d'adresses

(option accessible uniquement lorsque la précédente est cochée)

Cette option provoque la reconstruction complète de la table d'adresses des enregistrements au moment du compactage. Cette opération permet d'optimiser la taille de la table d'adresses. Elle est principalement utile dans les bases de données où de très nombreux enregistrements ont été créés puis supprimés. Dans les autres cas, l'optimisation ne sera pas déterminante.

A noter que cette option ralentit le compactage de façon conséquente et qu'elle rend invalides les ensembles sauvegardés via la commande `SAVE SET`. Il est d'ailleurs fortement recommandé dans ce cas de supprimer les ensembles sauvegardés car leur utilisation peut conduire à des sélections de données erronées.

- Le compactage tient compte des enregistrements des tables placées dans la corbeille. La présence d'un grand nombre d'enregistrements dans la corbeille peut constituer un facteur de ralentissement supplémentaire pour l'opération.
- L'utilisation de cette option rend la table d'adresses, et donc la base de données, incompatibles avec le fichier d'historique courant (s'il en existe un). Il sera automatiquement sauvegardé et un nouveau fichier d'historique devra être créé au prochain lancement de l'application.
- Vous pouvez déterminer si la table d'adresses a besoin d'être compactée en comparant sa taille avec le nombre total d'enregistrements dans la Page [Informations](#) du CSM.

Page Retour en arrière

La page Retour arrière du CSM permet d'accéder à la fonction de retour en arrière parmi les opérations effectuées dans le fichier de données. Elle s'apparente à une fonction d'annulation multi-niveaux. Elle est utile notamment lorsqu'un enregistrement a été supprimé par erreur dans la base de données.

Pour que cette fonction soit accessible, il est impératif que l'application travaille avec un fichier d'historique.

The list below shows all the performed operations recorded in the log file since the last backup.

Operation	Action	Table	Primary...	Process	Size	Date	Hour	System User	4D User	Values	Record
1006	Deletion	Parts	103		63	16/12/2019	18:10	aschmitt	Main user		102
1007	Deletion	Parts	102		63	16/12/2019	18:10	aschmitt	Main user		101
1008	Deletion	Parts	101		63	16/12/2019	18:10	aschmitt	Main user		100
1009	Deletion	Parts	100		63	16/12/2019	18:10	aschmitt	Main user		99
1010	Deletion	Parts	99		63	16/12/2019	18:10	aschmitt	Main user		98
1011	Deletion	Parts	98		63	16/12/2019	18:10	aschmitt	Main user		97
1012	Deletion	Parts	97		63	16/12/2019	18:10	aschmitt	Main user		96
1013	Deletion	Parts	146		63	16/12/2019	18:10	aschmitt	Main user		145
1014	Deletion	Parts	145		63	16/12/2019	18:10	aschmitt	Main user		144
1015	Deletion	Parts	144		63	16/12/2019	18:10	aschmitt	Main user		143
1016	Deletion	Parts	143		63	16/12/2019	18:10	aschmitt	Main user		142
1017	Deletion	Parts	142		63	16/12/2019	18:10	aschmitt	Main user		141
1018	Deletion	Parts	141		63	16/12/2019	18:10	aschmitt	Main user		140
1019	Deletion	Parts	8		63	16/12/2019	18:10	aschmitt	Main user		7
1020	Deletion	Parts	7		63	16/12/2019	18:10	aschmitt	Main user		6
1021	Deletion	Parts	6		63	16/12/2019	18:10	aschmitt	Main user		5

By selecting a specific log action in the list above and clicking the Rollback button, 4D Application will close the current data file, restore and open the selected backup of the database and perform all the above logged actions including the selected one.

The rollback operation cannot be undone, but the current data file will be renamed and stored on the disk.

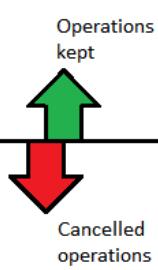
Current log file

Rollback

Si la base de données est chiffrée et si aucune clé de données valide correspondant au fichier d'historique ouvert n'a été fournie, les valeurs chiffrées ne s'affichent pas dans la colonne Valeurs et un dialogue s'affiche, demandant la saisie d'une phrase secrète ou de la clé de données, si vous cliquez sur le bouton Revenir en arrière.

Le contenu et le fonctionnement de la liste des opérations sont identiques à ceux de la fenêtre [d'analyse d'activités](#). Pour plus d'informations, reportez-vous à la section Page Analyse d'activités.

Pour effectuer un retour en arrière parmi les opérations, vous devez sélectionner la ligne située après laquelle toutes les opérations doivent être annulées. L'opération de la ligne sélectionnée sera la dernière conservée. Si par exemple vous souhaitez annuler une suppression, sélectionnez l'opération située juste avant la suppression. L'opération de suppression et les opérations suivantes seront annulées.



A screenshot of a database table showing a history of operations. The table has columns: Operation, Action, Table, Primary..., Process, Size, Date, Hour, System User, 4D User, Values, and Recorc. A horizontal line separates the table from a legend. The legend contains two items: 'Operations kept' with a green arrow pointing upwards, and 'Cancelled operations' with a red arrow pointing downwards. A vertical line labeled 'Selected operation' points to the row with ID 1014.

Operation	Action	Table	Primary...	Process	Size	Date	Hour	System User	4D User	Values	Recorc
1006	Deletion	Parts	103		63	16/12/2019	18:10	aschmitt	Main user		102
1007	Deletion	Parts	102		63	16/12/2019	18:10	aschmitt	Main user		101
1008	Deletion	Parts	101		63	16/12/2019	18:10	aschmitt	Main user		100
1009	Deletion	Parts	100		63	16/12/2019	18:10	aschmitt	Main user		99
1010	Deletion	Parts	99		63	16/12/2019	18:10	aschmitt	Main user		98
1011	Deletion	Parts	98		63	16/12/2019	18:10	aschmitt	Main user		97
1012	Deletion	Parts	97		63	16/12/2019	18:10	aschmitt	Main user		96
1013	Deletion	Parts	146		63	16/12/2019	18:10	aschmitt	Main user		145
1014	Deletion	Parts	145		63	16/12/2019	18:10	aschmitt	Main user	144	
1015	Deletion	Parts	144		63	16/12/2019	18:10	aschmitt	Main user		143
1016	Deletion	Parts	143		63	16/12/2019	18:10	aschmitt	Main user		142
1017	Deletion	Parts	142		63	16/12/2019	18:10	aschmitt	Main user		141
1018	Deletion	Parts	141		63	16/12/2019	18:10	aschmitt	Main user		140
1019	Deletion	Parts	8		63	16/12/2019	18:10	aschmitt	Main user		7
1020	Deletion	Parts	7		63	16/12/2019	18:10	aschmitt	Main user		6
1021	Relation	Parts	6		63	16/12/2019	18:10	aschmitt	Main user		5

Cliquez ensuite sur le bouton Revenir en arrière. 4D vous demande de confirmer l'opération. Si vous cliquez sur OK, les données sont alors restituées dans l'état exact où elles se trouvaient au moment de l'action sélectionnée.

Le menu situé en bas de la fenêtre vous permet de sélectionner le fichier d'historique à utiliser lorsque vous appliquez la fonction de retour en arrière à une base restituée. Dans ce cas, vous devez désigner le fichier d'historique correspondant à l'archive.

Le principe mis en oeuvre pour la fonction de retour arrière est le suivant : lorsque l'utilisateur clique sur le bouton Revenir en arrière, 4D referme la base de données courante et restitue la dernière sauvegarde des données de la base. La base restituée est ensuite ouverte et 4D intègre les opérations de l'historique jusqu'à l'opération sélectionnée. Si la base n'avait pas encore été sauvegardée, 4D repart d'un fichier de données vierge.

Page Réparation

Cette page permet de réparer le fichier de données ou le fichier de structure lorsqu'il a été endommagé. Generally, you will only use these functions under the supervision of 4D technical teams, when anomalies have been detected while opening the application or following a [verification](#).

Attention : Chaque réparation entraîne la duplication du fichier d'origine et donc l'augmentation de la taille du dossier de l'application. Il est important de prendre cela en considération (notamment sous macOS, où les applications 4D apparaissent sous forme de paquet) afin de ne pas augmenter excessivement la taille de l'application. Une intervention manuelle à l'intérieur du package peut être utile afin de supprimer les copies des fichiers d'origine.

La réparation n'est disponible qu'en mode maintenance. Si vous tentez d'effectuer cette opération en mode standard, une boîte de dialogue d'alerte vous prévient que l'application va être fermée puis relancée en mode maintenance. Lorsque la base est chiffrée, la réparation des données comprend le déchiffrement et le chiffrement et nécessite ainsi la clé de chiffrement de données courante. Si aucune clé de chiffrement valide n'a déjà été fournie, une boîte de dialogue s'affiche pour demander pour demander le mot de passe ou la clé de chiffrement (voir [Page Chiffrement](#)).

Fichiers

Fichier de données à réparer

Chemin d'accès du fichier de données courant. Le bouton [...] permet de désigner un autre fichier de données. Lorsque vous cliquez sur ce bouton, une boîte de dialogue standard d'ouverture de documents s'affiche, vous permettant de désigner le fichier de données à réparer. Une fois cette boîte de dialogue validée, le chemin d'accès du fichier à réparer est indiqué dans la fenêtre. Si vous effectuez une réparation par [réparation par en-têtes d'enregistrements](#), vous pouvez sélectionner tout fichier de données. Une fois cette boîte de dialogue validée, le chemin d'accès du fichier à réparer est indiqué dans la fenêtre.

Dossier de sauvegarde

Par défaut, le fichier de données original sera dupliqué avant réparation. Il sera placé dans un sous-dossier libellé "Replaced files (repairing)" dans le dossier de l'application. Le second bouton [...] permet de désigner un autre emplacement pour les sauvegardes des fichiers originaux effectuées avant réparation. Cette option permet notamment de réparer des fichiers volumineux en utilisant différents disques.

Fichiers réparés

4D crée un nouveau fichier de données vide à l'emplacement du fichier d'origine. Le fichier d'origine est déplacé dans le dossier nommé "%Replaced Files (Repairing) date heure" dont l'emplacement a été défini dans la zone de "Dossier de sauvegarde" (dossier de l'application par défaut). Le fichier vide est rempli avec les données récupérées.

Réparation standard

La réparation standard permet de réparer des données dans lesquelles seuls quelques enregistrements ou index sont endommagés (les tables d'adresses sont intactes). Les données sont compactées et réparées. A noter que ce type de réparation ne peut être effectué que si le fichier de données et le fichier de structure correspondent.

A l'issue de la procédure, la page "Réparation" du CSM est affichée. Un message indique si la réparation a été effectuée avec succès. Dans ce cas, vous pouvez immédiatement ouvrir l'application.



Réparation par en-têtes d'enregistrements

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

Les enregistrements de 4D sont de taille variable : il est donc nécessaire, pour les retrouver, de conserver dans une table spéciale l'endroit où ils sont stockés sur votre disque. Le programme accède donc à l'adresse de l'enregistrement par l'intermédiaire d'un index et d'une table d'adresses. Si seuls des enregistrements ou des index sont endommagés, l'option de réparation standard suffira généralement pour résoudre le problème. C'est lorsque la table d'adresses est touchée qu'il faudra en venir à une récupération plus sophistiquée, puisqu'il faut la reconstituer. Pour réaliser cette opération, le CSM utilise le marqueur qui se trouve en en-tête de chaque enregistrement. Les marqueurs peuvent être comparés à des résumés des enregistrements, comportant l'essentiel de leurs informations, et à partir desquels une reconstitution de la table d'adresses est possible.

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

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

Lorsque vous cliquez sur le bouton Réparer, 4D effectue une analyse complète du fichier de données. A l'issue de cette analyse, le résultat est affiché dans la fenêtre suivante :

The screenshot shows the 'Centre de Sécurité et de Maintenance' window with the 'Réparation' tab selected. On the left, a sidebar lists various maintenance tasks: Informations, Analyse d'activités, Vérification, Sauvegarde, Compactage, Retour arrière, Restitution, Réparation (which is selected), and Chiffrement. The main area has two tables. The left table, 'Enregistrements trouvés dans le fichier de données', lists records found in the data file with columns: Ordre (Order), Nombre (Number), Table de destination (Destination Table), and Récupérer (Retrieve). The right table, 'Tables du fichier de structure', lists tables from the structure file with columns: Ordre (Order) and Tables. Below the tables are buttons: 'Afficher les numéros de table' (Show table numbers), 'Ignorer enregistrements' (Ignore records), 'Identifier table' (Identify table), 'Voir le compte rendu' (View report), 'Retour' (Back), and 'Récupérer' (Repair). A warning message at the bottom states: '4D n'a pas réussi à déterminer à quelle table appartiennent certains enregistrements. Pour relier ces enregistrements à une table de la structure, cliquez sur le bouton "Identifier table".'

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

La zone "Enregistrements trouvés dans le fichier de données" comporte deux tableaux synthétisant les informations issues de l'analyse du fichier de données.

- Le premier tableau liste les informations issues de l'analyse du fichier de données. Chaque ligne représente un groupe d'enregistrements récupérables dans le fichier de données :
 - La colonne Ordre indique l'ordre de récupération des groupes d'enregistrements.
 - La colonne Nombre indique le nombre d'enregistrements contenus dans la table.
 - La colonne Table de destination indique le nom des tables ayant pu être automatiquement associées aux groupes d'enregistrements identifiés. Les noms des tables attribuées automatiquement sont affichés en caractères verts. Les groupes qui n'ont pas encore été attribués, c'est-à-dire, les tables qui n'ont pas pu être associées à des enregistrements sont affichées en caractères rouges.
 - La colonne Récupérer permet vous permet d'indiquer pour chaque groupe si vous souhaitez récupérer les enregistrements. Par défaut, l'option est cochée pour tous les groupes avec les enregistrements qui peuvent être associés à une table.
- Le deuxième tableau liste les tables du fichier de structure.

Attribution manuelle

Si, du fait de l'endommagement de la table d'adresses, un ou plusieurs groupes d'enregistrements n'ont pas pu être attribués à des tables, vous pouvez les attribuer manuellement. Pour attribuer une table à un groupe non identifié, sélectionnez le groupe dans le premier tableau. Lorsque vous sélectionnez des enregistrements non identifiés, la zone "Contenu des enregistrements" affiche une prévisualisation du contenu des premiers enregistrements du groupe afin de vous permettre de les attribuer plus facilement :

The screenshot shows the 'Repair' section of the Maintenance and Security Center. On the left, a sidebar lists various maintenance tasks: Information, Activity analysis, Verify, Backup, Compact, Rollback, Restore, Repair (which is selected and highlighted in blue), and Encrypt. The main area has two tables. The left table, 'Records found in the data file', lists groups of records with columns for Order, Count, Destination table, and Recover (checkbox). The second row, 'Unidentified destination.', has its checkbox unchecked. The right table, 'Tables of the structure file', lists available tables with columns for Order and Tables. Below these tables are buttons for 'Display table numbers', 'Ignore records', and 'Identify table'. A warning message states: '4D was not able to determine which table certain records belong to. In order to associate these records with a table in the structure, click on the "Identify table" button.' At the bottom, there's a table preview titled 'Content of the records' showing sample data from fields #1 to #4. Finally, at the bottom right are 'Open log file', 'Back', and 'Recover' buttons.

Sélectionnez ensuite la table à attribuer dans le tableau des "Tables non attribuées" puis cliquez sur le bouton Identifier table. Vous pouvez également attribuer une table par glisser-déposer. Le groupe d'enregistrements est alors associé à la table, il sera récupéré dans cette table. Les noms des tables attribuées manuellement sont affichés en caractères noirs. Le bouton Ignorer enregistrements permet de supprimer l'association effectuée manuellement entre une table et un groupe d'enregistrements.

Voir le compte rendu

Une fois la réparation terminée, 4D génère un fichier de compte-rendu dans le dossier Logs du projet. Ce fichier liste l'ensemble des opérations qui ont été menées. Il est créé au format xml et est nommé : *ApplicationName_Repair_Log_yyyy-mm-dd hh-mm-ss.xml*" où :

- *ApplicationName* est le nom du fichier de structure sans extension, par exemple "Factures",
- *aaaa-mm-jj hh-mm-ss* est l'horodatage du fichier, basé sur la date et l'heure système locales au moment du lancement de l'opération de vérification, par exemple "2019-02-11 15-20-45".

Lorsque vous cliquez sur le bouton Voir le compte rendu, 4D affiche le fichier de compte-rendu le plus récent dans le navigateur par défaut de l'ordinateur.

Page chiffrement

Vous pouvez vous aider de cette page pour chiffrer ou *déchiffrer* (i.e. enlever le chiffrement) le fichier de données, en fonction du statut de l'attribut Chiffrable défini pour chaque table de la base.

For detailed information about data encryption in 4D, please refer to the "Encrypting data" section in the *Design Reference* manual. Vous pouvez également lire l'article de blog intitulé [A deeper look into 4D data encryption](#).

Un nouveau dossier est créé à chaque opération de chiffrement/déchiffrement. Il est nommé "Replaced Files (Encrypting) yyyy-mm-dd hh-mm-ss" ou "Replaced Files (Decrypting) yyyy-mm-dd hh-mm-ss".

Le chiffrement est disponible uniquement en [mode maintenance](#). Si vous tentez d'effectuer cette opération en mode standard, une boîte de dialogue d'alerte vous prévient que l'application va être fermée puis relancée en mode maintenance

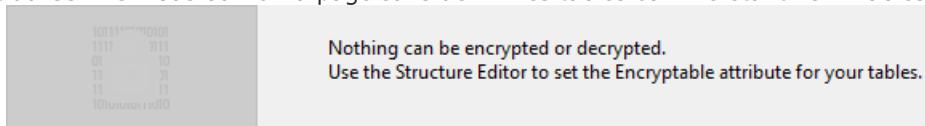
Attention :

- Le chiffrement d'un fichier de données est une opération de longue durée. Un indicateur de progression de l'opération s'affiche (et peut être interrompu par l'utilisateur). À noter également que le chiffrement d'une application comprend toujours une étape de compactage.
- Chaque opération de chiffrement génère une copie du fichier de données, ce qui augmente la taille du dossier de l'application. Il est important de prendre cela en considération (notamment sous macOS, où les applications 4D apparaissent sous forme de paquet) afin de ne pas augmenter excessivement la taille de l'application. Le déplacement ou la suppression manuels des copies du fichier original dans le paquet peut aider à réduire la taille du paquet.

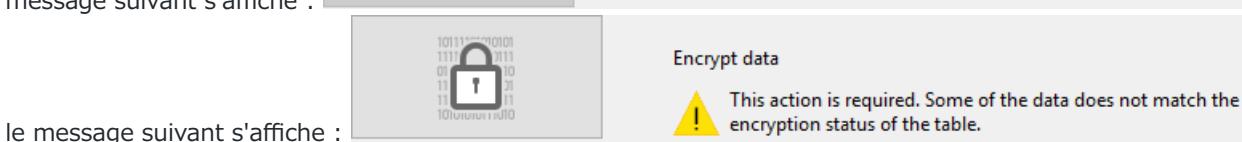
Chiffrer des données pour la première fois

Trois étapes sont nécessaires pour effectuer le tout premier chiffrement de vos données à l'aide du CSM :

1. Dans l'éditeur de structure, cochez l'attribut Chiffrable pour chaque table dont vous souhaitez chiffrer les données. Consultez la section "Propriétés des tables".
2. Ouvrez la page Chiffrement du CSM. Si vous ouvrez la page sans définir les tables comme étant Chiffrables, le message suivant s'affiche :



Sinon,



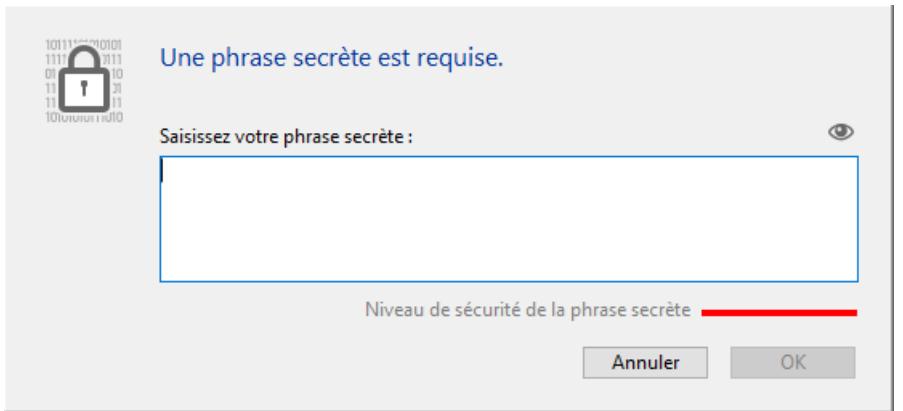
le message suivant s'affiche :

Cela signifie que le statut Chiffrable d'au moins une table a été modifié et que le fichier de données n'a pas encore été chiffré. **Note : **Le même message s'affiche lorsque le statut Chiffrable a été modifié dans un fichier de données déjà chiffré ou après le déchiffrement d'un fichier de données (voir ci-dessous).

3. Cliquez sur le bouton de Chiffrement.



Vous serez ensuite invité à saisir une phrase secrète pour votre fichier de données :



La phrase secrète est utilisée pour générer la clé de chiffrement des données. Une phrase secrète est une version plus sécurisée d'un mot de passe et peut contenir un grand nombre de caractères. For example, you could enter a passphrases such as "We all came out to Montreux" or "My 1st Great Passphrase!!" The security level indicator can help you evaluate the strength of your passphrase:

[Passphrase security level:](#)



(deep green is the highest level)

4. Tapez sur Entrée pour confirmer votre phrase secrète sécurisée.

Le processus de chiffrement est alors lancé. Si le CSM est ouvert en mode standard, l'application est rouverte en mode maintenance.

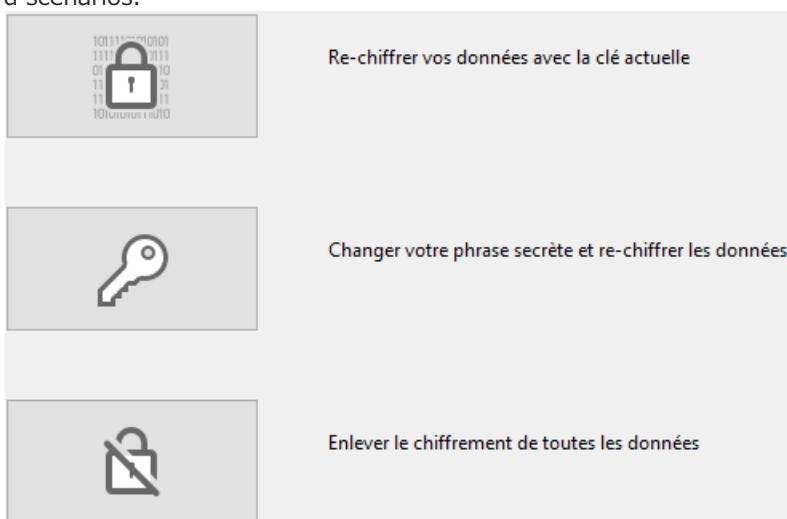
4D propose de sauvegarder la clé de chiffrement (voir le paragraphe [Sauvegarder la clé de chiffrement](#) ci-dessous). Vous pouvez la sauvegarder à ce moment précis ou bien ultérieurement. Vous pouvez également ouvrir le fichier d'historique du chiffrement.

Si le processus de chiffrement est réussi, la page Chiffrement affiche les boutons Opérations de maintenance liées au chiffrement.

Attention : Durant l'opération de chiffrement, 4D crée un nouveau fichier de données vide et y insère des données à partir du fichier de données original. Les enregistrements correspondant aux tables "chiffrées" sont chiffrés puis copiés ; les autres enregistrements sont uniquement copiés (une opération de compactage est également exécutée). Si l'opération est réussie, le fichier de données original est déplacé vers un dossier "Replaced Files (Encrypting)". Si vous souhaitez transmettre un fichier de données chiffré, assurez-vous d'avoir préalablement déplacé/retiré tout fichier de données non chiffrées du dossier de l'application.

Opérations de maintenance liées au chiffrement

When an application is encrypted (see above), the Encrypt page provides several encryption maintenance operations, corresponding to standard scenarios.



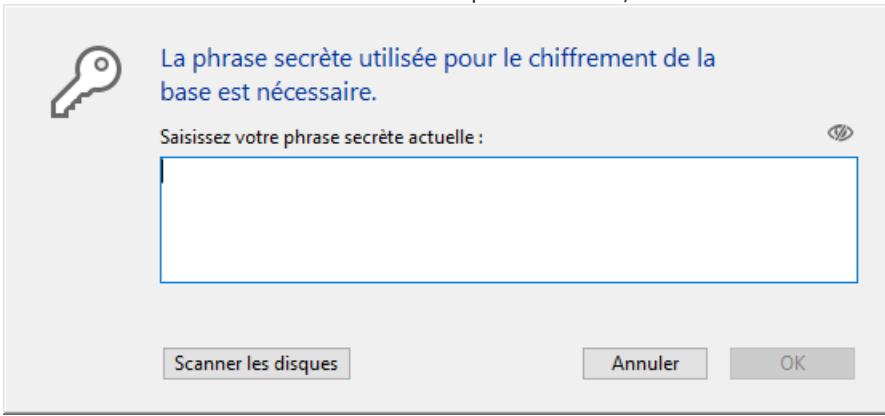
Fournir la clé de chiffrement des données actuelle

Pour des raisons de sécurité, toutes les opérations de maintenance liées au chiffrement nécessitent la clé de chiffrement des données actuelle.

- Si la clé de chiffrement des données est déjà chargée dans le trousseau 4D(1), elle est automatiquement réutilisée

par 4D.

- Si la clé de chiffrement des données n'est pas identifiée, vous devez la fournir. La boîte de dialogue suivante s'affiche



À ce stade, deux options s'offrent à vous :

- entrer la phrase secrète actuelle(2) et cliquer sur OK. OR
- connecter un appareil tel qu'une clé USB et cliquer sur le bouton Scanner les disques.

(1) Le trousseau de 4D stocke toutes les clés de chiffrement de données valides saisies durant la session d'application session.

(2) Le mot de passe actuel correspond au mot de passe utilisé pour générer la clé de chiffrement actuelle.

Dans tous les cas, si des informations valides sont fournies, 4D redémarre en mode maintenance (si ce n'est pas déjà le cas) et exécute l'opération.

Re-chiffrer les données à l'aide de la clé de chiffrement actuelle

Cette opération est utile lorsque l'attribut Chiffrable a été modifié pour une ou plusieurs tables contenant des données. Dans ce cas, afin d'éviter toute incohérence dans le fichier de données, 4D désactive l'accès en écriture aux enregistrements des tables dans l'application. Il est alors nécessaire de re-chiffrer les données pour restituer un statut de chiffrement valide.

1. Cliquez sur Re-chiffrer les données à l'aide de la clé actuelle .
2. Saisissez la clé de chiffrement des données actuelle.

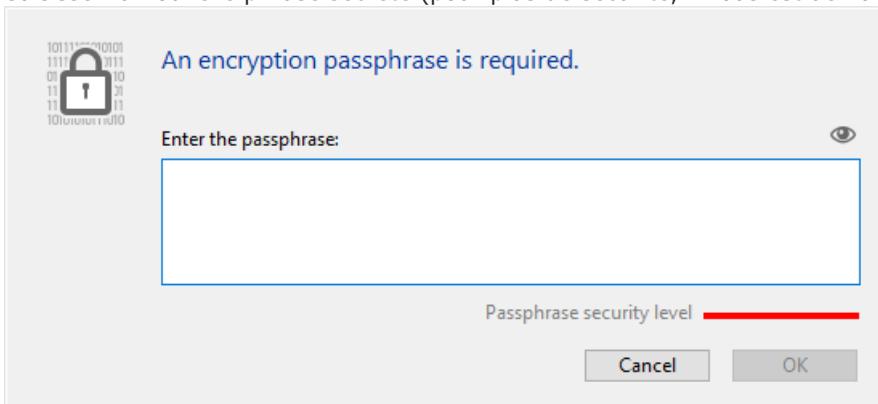
Le fichier de données est correctement re-chiffré à l'aide de la clé actuelle et un message de confirmation s'affiche :



Changer votre phrase secrète et re-chiffrer les données

Cette opération est utile en cas de modification de la clé de chiffrement des données actuelle. Par exemple, il se peut que vous la modifiez pour vous conformer aux règles de sécurité (telles que la nécessité de modifier la phrase secrète tous les trois mois).

1. Cliquez sur Changer votre phrase secrète et re-chiffrer les données .
2. Saisissez la clé de chiffrement des données actuelle.
3. Saisissez la nouvelle phrase secrète (pour plus de sécurité, il vous est demandé de la saisir deux fois) :



Le fichier de données est entièrement

déchiffré et un message de confirmation s'affiche :



Vos données ont été chiffrées avec succès.

Enlever le chiffrement de toutes les données

Cette opération supprime tout le chiffrement du fichier de données. Si vous ne souhaitez plus que vos données soient chiffrées :

1. Cliquez sur Enlever le chiffrement de toutes les données.
2. Saisissez la clé de chiffrement de données actuelle (voir Fournir la clé de chiffrement des données actuelle).

Le fichier de données est entièrement déchiffré et un message de confirmation s'affiche :



Your data is no longer encrypted.

Une fois que le fichier de données est déchiffré, le statut de chiffrement des tables ne correspond plus à leur attribut Chiffrable. Pour restituer un statut de mise en correspondance, vous devez décocher tous les attributs Chiffrable au niveau de la structure de la base.

Sauvegarder la clé de chiffrement

4D vous permet de sauvegarder la clé de chiffrement des données dans un fichier créé à cet effet. La sauvegarde de ce fichier sur un appareil externe tel qu'une clé USB facilitera l'utilisation d'une application chiffrée, étant donné que l'utilisateur connecterait cet appareil uniquement pour fournir la clé avant d'ouvrir l'application et accéder aux données chiffrées.

Vous pouvez sauvegarder la clé de chiffrement chaque fois qu'une nouvelle phrase secrète est fournie :

- lorsque l'application est chiffrée pour la première fois,
- lorsque l'application est re-chiffrée avec une nouvelle phrase secrète.

Les clés de chiffrement successives peuvent être sauvegardées sur le même appareil.

Fichier d'historique

Une fois qu'une opération de chiffrement est terminée, 4D génère un fichier dans le dossier Logs de l'application. Il est créé au format XML et nommé "*ApplicationName_Encrypt_Log_yyyy-mm-dd hh-mm-ss.xml*" ou "*ApplicationName_Decrypt_Log_yyyy-mm-dd hh-mm-ss.xml*".

Chaque fois qu'un nouveau fichier d'historique est généré, un bouton Voir le compte rendu s'affiche dans la page CSM.

Le fichier d'historique liste toutes les opérations internes liées au processus de chiffrement/déchiffrement qui ont été exécutées, ainsi que les erreurs (le cas échéant).

Sauvegarde

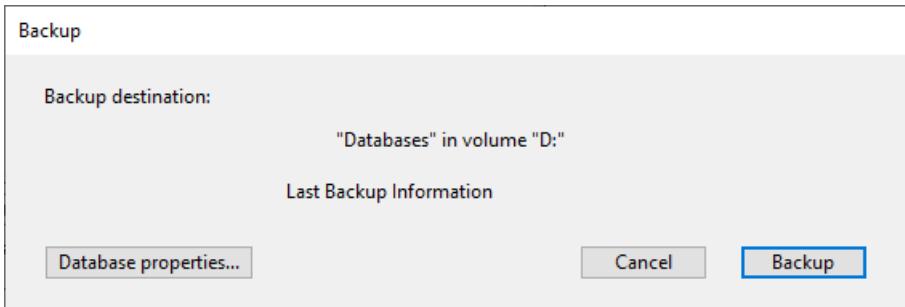
Une sauvegarde peut être déclenchée de trois manières :

- Manuellement, via la commande Sauvegarde... du menu Fichier de 4D ou le bouton Sauvegarde du [Centre de sécurité et de maintenance \(CSM\)](#).
- Automatiquement, via le programmeur paramétrable dans les Propriétés,
- Par programmation, à l'aide de la commande `BACKUP`.

4D Server : Il est possible de déclencher "manuellement" une sauvegarde depuis un poste distant, via une méthode appelant la commande `BACKUP`. Dans tous les cas, la commande sera exécutée sur le serveur.

Sauvegarde manuelle

1. Choisissez la commande Sauvegarde... dans le menu Fichier de 4D.



La fenêtre de sauvegarde s'affiche :

Vous pouvez visualiser l'emplacement du dossier de sauvegarde à l'aide du pop up menu associé à la zone "Destination de la sauvegarde". Cet emplacement est défini dans la [Page Sauvegarde/Configuration des Propriétés de la base](#).

- Vous pouvez également sélectionner [Centre de sécurité et de maintenance](#) de 4D et afficher la [Page Sauvegarde](#).

Le bouton Propriétés de la base... provoque l'affichage de la boîte de dialogue des Propriétés de structure.

2. Cliquez sur le bouton Sauvegarde pour déclencher la sauvegarde avec les paramètres courants.

Sauvegarde automatique périodique

Les sauvegardes périodiques sont déclenchées automatiquement. Elles sont configurées dans la [Page Sauvegarde/Périodicité des Propriétés](#).

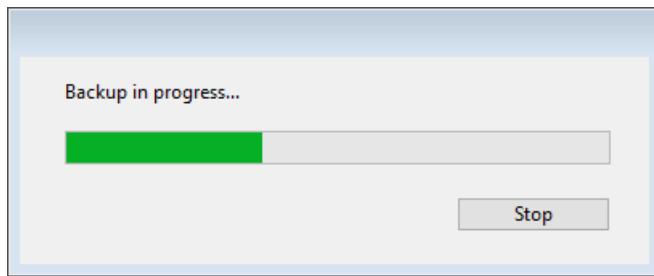
Les sauvegardes s'effectuent automatiquement au moment défini, sans intervention de l'utilisateur. Pour plus d'informations sur le fonctionnement de cette boîte de dialogue, reportez-vous à la section [Définir des sauvegardes périodiques](#).

Commande BACKUP

Lorsque la commande `BACKUP` est exécutée depuis une méthode, la sauvegarde est alors déclenchée avec les paramètres courants définis dans les propriétés. Vous pouvez utiliser la Méthode base `On Backup Startup` et `On Backup Shutdown` afin de contrôler le déroulement de la sauvegarde (cf. le manuel [Langage de 4D](#)).

Déroulement de la sauvegarde

Une fois qu'une sauvegarde a été déclenchée, 4D affiche une boîte de dialogue comportant un thermomètre indiquant la progression de l'opération :



Le thermomètre est également affiché en bas de la page "[Sauvegarde](#)" du CSM si vous avez utilisé cette boîte de dialogue.

Le bouton Arrêter permet à l'utilisateur d'interrompre la sauvegarde à tout instant (cf. paragraphe [En cas de problème au cours de la sauvegarde](#)).

Le compte-rendu de la dernière sauvegarde (succès ou échec) est stocké dans la zone "Informations sur la dernière sauvegarde" dans la [Page Sauvegarde du CSM](#) ou dans la Page de maintenance de 4D Server. Il est également enregistré dans le Backup journal.txt.

Accès à l'application durant la sauvegarde

4D verrouille les process liés aux types de fichiers inclus dans la sauvegarde : si seul le fichier de structure est sauvegardé, l'accès à la structure est impossible mais l'accès aux données est autorisé. Pendant la sauvegarde, les accès à l'application sont restreints par 4D en fonction du contexte.

A l'inverse, si seul le fichier de données est sauvegardé, l'accès à la structure reste autorisé. Dans ce cas, les possibilités d'accès à l'application sont les suivantes :

- Avec 4D version monoposte, l'application est verrouillée en lecture et en écriture, tous les process sont gelés. Toute action est alors impossible.
- Avec 4D Server, l'application est verrouillée uniquement en écriture, les postes clients peuvent consulter les données. Si un poste client envoie une requête d'ajout, de suppression ou de modification au serveur, il obtient une fenêtre l'invitant à attendre la fin de la sauvegarde. Une fois l'application sauvegardée, la fenêtre disparaît d'elle-même et l'action est exécutée. Pour annuler la requête en cours et ne pas avoir à attendre la fin de la sauvegarde, il suffit de cliquer sur le bouton Annuler l'opération. Cependant, si l'action en attente provient d'une méthode lancée avant la sauvegarde, il est déconseillé de l'annuler car seules les opérations restant à effectuer seront annulées. Or, une méthode "à moitié" exécutée peut conduire à des incohérences logiques dans les données. Lorsque l'action en attente provient d'une méthode et que l'utilisateur clique sur le bouton Annuler l'opération, 4D Server renvoie l'erreur -9976 (Cette commande ne peut être exécutée car la base est en cours de sauvegarde).

En cas de problème au cours de la sauvegarde

Il est possible qu'une sauvegarde ne s'effectue pas correctement. Les causes de l'échec peuvent être diverses : interruption par l'utilisateur, fichier joint introuvable, disque de destination défaillant, transaction non terminée, etc. 4D traite l'incident selon la cause de l'échec.

Dans tous les cas, le statut de la dernière sauvegarde (succès ou échec) est stocké dans la zone "Informations sur la dernière sauvegarde" dans la [Page Sauvegarde du CSM](#) ou dans la Page de maintenance de 4D Server, ainsi que dans le Backup journal.txt.

- Interruption par l'utilisateur : le bouton Arrêter de la boîte de dialogue de progression de la sauvegarde permet aux utilisateurs d'interrompre la sauvegarde à tout instant. Dans ce cas, la copie des éléments est stoppée et l'erreur 1406 est générée. Vous pouvez intercepter cette erreur dans la Méthode base `On Backup Shutdown`.
- Fichier joint introuvable : lorsqu'un fichier joint est introuvable, 4D effectue une sauvegarde partielle (sauvegarde des fichiers de l'application et des fichiers joints accessibles) et retourne une erreur.
- Sauvegarde impossible (disque plein ou protégé en écriture, disque manquant, panne du disque, transaction non terminée, projet non lancé au moment d'une sauvegarde automatique périodique, etc.) : S'il s'agit du premier échec, 4D effectuera ultérieurement une seconde tentative. Le délai d'attente entre les deux tentatives est défini dans la Page Sauvegarde/Sauvegarde& et Restitution des Propriétés. Si la seconde tentative échoue également, une boîte de dialogue d'alerte système est affichée et une erreur est générée. Vous pouvez intercepter cette erreur dans la Méthode base `On Backup Shutdown`.

Journal de sauvegarde (Backup Journal)

Pour faciliter le suivi et la vérification des sauvegardes, le module de sauvegarde résume chaque opération effectuée dans un fichier spécial, similaire à un journal d'activité. Comme un manuel intégré, toutes les opérations de la base de données (sauvegardes, restaurations, intégrations de fichiers d'historique) sont consignées dans ce fichier, qu'elles aient été planifiées ou exécutées manuellement. La date et l'heure auxquelles ces opérations ont eu lieu sont également notées dans le journal.

Le journal de sauvegarde s'appelle "Backup Journal[001].txt" et se trouve dans le dossier "Logs" du projet. Le journal de sauvegarde peut être ouvert avec n'importe quel éditeur de texte.

Gestion de la taille du journal de sauvegarde

Dans certaines stratégies de sauvegarde (par exemple, dans le cas où de nombreuses pièces jointes sont sauvegardées), le journal de sauvegarde peut rapidement atteindre une taille importante. Deux mécanismes peuvent être utilisés pour gérer cette taille :

- Sauvegarde automatique : Avant chaque sauvegarde, l'application examine la taille du fichier backup journal courant. Si elle est supérieure à 10 Mo, le fichier courant est archivé et un nouveau fichier est créé avec le numéro [xxx] incrémenté, par exemple "Backup Journal[002] .txt". Une fois le numéro de fichier 999 atteint, la numérotation reprend à 1 et les fichiers existants seront remplacés.
- Possibilité de réduire la quantité d'informations enregistrées : Pour ce faire, il suffit de modifier la valeur de la clé `VerboseMode` dans le fichier `Backup.4DSettings` du projet. Par défaut, cette clé est définie sur True. Si vous définissez la valeur de cette clé sur False, seules les informations principales sont stockées dans le journal de sauvegarde : la date et l'heure du début de l'opération et les éventuelles erreurs générées. Les clés XML concernant la configuration de la sauvegarde sont décrites dans le manuel *Sauvegarde des clés XML 4D*.

backupHistory.json

Toutes les informations concernant les dernières opérations de sauvegarde et de restitution sont stockées dans le fichier `backupHistory.json` de l'application. Ce dernier enregistre le chemin de chaque fichier sauvegardé (y compris les pièces jointes) ainsi que le numéro, la date, l'heure, la durée et le statut de chaque opération. Afin de limiter la taille du fichier, le nombre d'opérations enregistrées et le nombre de sauvegardes disponibles ("Keep only the last X backup files") définies dans les propriétés de sauvegarde est identique.

Le fichier `backupHistory.json` se situe dans le dossier de destination de sauvegarde courant. Vous pouvez obtenir le chemin de ce fichier à l'aide de la déclaration suivante :

```
$backupHistory:=Get 4D file(Backup history file)
```

ATTENTION

La suppression ou le déplacement du fichier `backupHistory.json` entraînera la réinitialisation du prochain numéro de sauvegarde. Le fichier `backupHistory.json` est formaté afin d'être utilisé par l'application 4D. Si vous recherchez un état lisible sur les opérations de sauvegarde, le journal de sauvegarde sera plus précis.

Paramètres de sauvegarde

Les paramètres de sauvegarde sont définis sur trois pages dans la [boîte de dialogue des Paramètres](#). Vous pouvez définir :

- la périodicité des sauvegardes automatiques
- les fichiers à inclure dans chaque sauvegarde
- les fonctionnalités avancées permettant d'exécuter des tâches automatiques

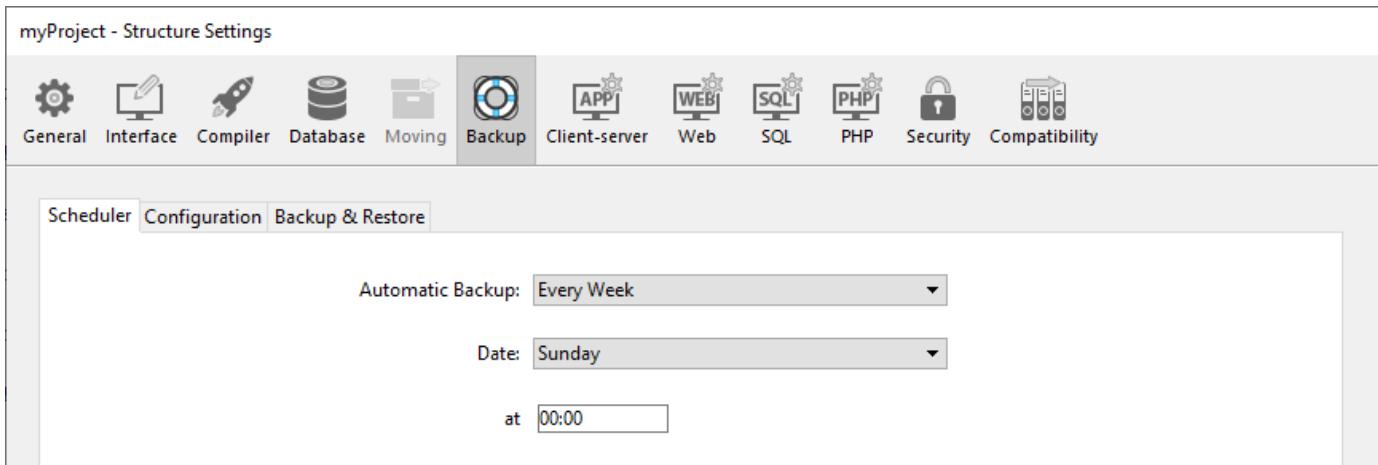
Les propriétés définies dans cette boîte de dialogue sont écrits dans le fichier *Backup.4DSettings*, stocké dans le dossier *Settings*.

Sauvegardes périodiques

Vous pouvez automatiser les sauvegardes de vos applications ouvertes avec 4D ou 4D Server (même lorsqu'aucun poste distant n'est connecté). Le principe consiste à définir une fréquence de sauvegarde (en heures, jours, semaines ou mois) ; à chaque échéance, 4D déclenche automatiquement une sauvegarde en tenant compte des paramètres de sauvegarde courants.

Si l'application n'était pas lancée au moment théorique de la sauvegarde, 4D considère au lancement suivant que la sauvegarde a échoué et applique les paramétrages adéquats, définis dans les Propriétés (cf. paragraphe [En cas de problème au cours de la sauvegarde](#)).

Les paramètres des sauvegardes périodiques sont définis dans la Page [Sauvegarde/Périodicité des Propriétés](#) :



Les options regroupées dans cet onglet permettent de définir et de paramétrer des sauvegardes périodiques automatiques de l'application. Vous pouvez choisir un paramétrage standard rapide ou personnaliser entièrement la périodicité. Diverses options apparaissent en fonction de la valeur définie dans le menu Sauvegarde automatique :

- Jamais : la fonction de sauvegarde périodique est inactivée.
- Toutes les heures : programme une sauvegarde automatique par heure, à partir de la prochaine heure.
- Tous les jours : programme une sauvegarde automatique par jour. Une zone de saisie vous permet d'indiquer l'heure à laquelle la sauvegarde doit être déclenchée.
- Toutes les semaines : programme une sauvegarde automatique par semaine. Deux zones de saisie supplémentaires vous permettent d'indiquer le jour et l'heure de la sauvegarde.
- Tous les mois : programme une sauvegarde automatique par mois. Deux zones de saisie supplémentaires vous permettent d'indiquer le jour du mois et l'heure de la sauvegarde.
- Personnalisée : permet de configurer des sauvegardes périodiques "sur-mesure". Lorsque vous sélectionnez cette, plusieurs zones de saisie supplémentaires apparaissent :
 - Toutes les N heure(s) : permet de programmer des sauvegardes sur une base horaire. Vous pouvez saisir une valeur comprise entre 1 et 24.
 - Tous les N jour(s) à N : permet de programmer des sauvegardes sur une base journalière. Saisissez par exemple 1 si vous souhaitez une sauvegarde quotidienne. Lorsque vous cochez cette option, vous devez

indiquer l'heure à laquelle la sauvegarde doit être déclenchée.

- Toutes les N semaine(s), jour à N : permet de programmer des sauvegardes sur une base hebdomadaire. Saisissez 1 si vous souhaitez une sauvegarde hebdomadaire. Lorsque vous cochez cette option, vous devez indiquer le ou les jour(s) de la semaine et l'heure à laquelle chaque sauvegarde doit être déclenchée. Vous pouvez cocher un ou plusieurs jour(s) de la semaine. Par exemple, vous pouvez utiliser cette option pour définir deux sauvegardes hebdomadaires : une le mercredi et une le vendredi.
- Tous les N mois, Ne jour à N : permet de programmer des sauvegardes sur une base mensuelle. Saisissez 1 si vous souhaitez une sauvegarde mensuelle. Lorsque vous cochez cette option, vous devez indiquer le jour de chaque mois auquel la sauvegarde doit être déclenchée, ainsi que l'heure de déclenchement.

Les basculements entre l'heure standard et l'heure d'été peuvent affecter temporairement le planificateur automatique et déclencher la sauvegarde suivante avec un décalage d'une heure. Cela ne se produit qu'une seule fois et les sauvegardes ultérieures sont lancées à l'heure prévue.

Configuration

La Page Sauvegarde/Configuration des Propriétés permet de désigner les fichiers à sauvegarder, l'emplacement des fichiers de sauvegarde et le fichier d'historique. Ces paramètres sont spécifiques à chaque application ouverte par 4D ou 4D Server.

myProject - Structure Settings

General Interface Compiler Database Moving Backup Client-server Web SQL PHP Security Compatibility

Scheduler Configuration Backup & Restore

Content

Data
 Structure
 User Structure (only for binary database)

Attachments:

./Data/data.4DIndx
./Data/data.4DSyncData
./Data/data.4DSyncHeader

Delete Add folder... Add file...

Backup File Destination Folder

"myProject" in volume "D:"

Used Space: 130,19 Go Free Space: 346,26 Go

Log Management

Use Log:

"data.journal" in volume "D:"

4D Server : Ces paramètres peuvent être définis depuis le poste 4D Server uniquement.

Contenu

Cette zone permet de désigner les fichiers et/ou dossiers à copier lors de la prochaine sauvegarde.

- Data : fichier de données de l'application. Lorsque cette option est cochée, les éléments suivants sont

- automatiquement sauvegardés en même temps que les données :
- le fichier journal courant de l'application (le cas échéant),
 - le dossier **Settings** complet situé [à côté du fichier de données](#) (le cas échéant), c'est-à-dire *les paramètres utilisateur pour les données*.
- Structure : fichiers et dossiers du projet d'application. Dans le cas de projets compilés, cette option permet de sauvegarder le fichier .4dz. Lorsque cette option est cochée, le dossier complet **Settings** situé [au même niveau que le dossier Project](#), c'est-à-dire *les paramètres utilisateur*, est automatiquement sauvegardé.
 - Fichier de structure utilisateur (uniquement pour les bases binaires) : [fonctionnalité obsolète](#)
 - Attachments : cette zone permet de désigner un ensemble de fichiers et/ou de dossiers à sauvegarder en même temps que l'application. Ces fichiers peuvent être de tout type (documents ou modèles de plug-ins, étiquettes, états, images, etc.). Vous pouvez désigner soit des fichiers individuels, soit des dossiers dont le contenu sera intégralement sauvegardé. Chaque élément joint est listé avec son chemin d'accès complet dans la zone "Fichiers joints".
 - Supprimer : retire de la liste des fichiers joints l'élément sélectionné.
 - Ajouter dossier... : affiche une boîte de dialogue permettant de sélectionner un dossier à joindre à la sauvegarde. En cas de restitution, le dossier sera récupéré avec sa structure interne. Vous pouvez désigner tout dossier ou volume connecté au poste, à l'exception du dossier contenant les fichiers de l'application.
 - Ajouter fichier... : affiche une boîte de dialogue permettant de sélectionner un fichier à joindre à la sauvegarde.

Emplacement des fichiers de sauvegarde

Cette zone permet de visualiser et de modifier l'emplacement auquel seront stockés les fichiers de sauvegarde ainsi que les fichiers de sauvegarde du fichier d'historique (le cas échéant).

Pour visualiser l'emplacement des fichiers, cliquez dans la zone afin d'afficher leur chemin d'accès sous forme de pop up menu.

Pour modifier l'emplacement auquel ces fichiers devront être enregistrés, cliquez sur le bouton [...]. Une boîte de dialogue de sélection de dossier apparaît, vous permettant de désigner un dossier ou un volume devant accueillir les sauvegardes. Les zones "Espace utilisé" et "Espace libre" sont automatiquement mises à jour et indiquent l'espace disque disponible sur le volume du dossier sélectionné.

Gestion du fichier d'historique

L'option Utiliser le fichier d'historique indique, lorsqu'elle est cochée, que l'application exploite un fichier d'historique. Son chemin d'accès est précisé au-dessous de l'option. Lorsque cette option est cochée, il n'est pas possible d'ouvrir l'application sans fichier d'historique.

Par défaut, tout projet créé avec 4D utilise un fichier d'historique (option cochée dans la [Page Général des Préférences de 4D](#)). Le fichier d'historique est nommé par défaut *data.journal* et est placé dans le dossier Data.

L'activation d'un nouveau fichier d'historique nécessite que les données de l'application soient préalablement sauvegardées. Lorsque vous cochez cette option, un message vous informe qu'une sauvegarde est nécessaire. La création du fichier d'historique est différée et ne sera effective qu'après la prochaine sauvegarde de l'application.

Sauvegarde et restitution

La modification des options de sauvegarde et de restauration est facultative. Leurs valeurs par défaut correspondent à une utilisation standard de la fonction.

myProject - Structure Settings

Paramètres généraux

- Conserver uniquement les N derniers fichiers de sauvegarde : ce paramètre permet d'activer et de configurer le mécanisme de suppression des fichiers de sauvegarde les plus anciens, afin d'éviter tout risque de saturation du volume. Le principe de fonctionnement est le suivant : après avoir terminé la sauvegarde courante, 4D efface l'archive la plus ancienne si celle-ci est localisée au même endroit que l'archive à sauvegarder et porte le même nom (vous pouvez, pour des raisons d'économie de place, demander que l'archive la plus ancienne soit effacée avant la sauvegarde). Si, par exemple, le nombre de jeux est fixé à 3, les trois premières sauvegardes créent successivement les archives MaBase-0001, MaBase-0002 et MaBase-0003. Lors de la quatrième sauvegarde, l'archive MaBase-0004 est créée alors que l'archive MaBase-0001 est supprimée. Par défaut, le mécanisme de suppression des jeux est activé et 4D conserve 3 jeux de sauvegarde. Pour ne pas activer le mécanisme, désélectionnez l'option.

Ce paramètre concerne à la fois les sauvegardes de l'application et les sauvegardes de l'historique.

- Sauvegarder uniquement si le fichier de données a été modifié : lorsque cette option est cochée, 4D déclenche les sauvegardes périodiques uniquement si des données ont été ajoutées, modifiées ou supprimées depuis la dernière sauvegarde. Dans le cas contraire, la sauvegarde prévue est annulée et reportée à l'échéance suivante. Aucune erreur n'est générée ; le report est toutefois indiqué dans le Journal des sauvegardes. Cette option permet notamment d'économiser du temps machine sur la sauvegarde d'applications principalement utilisées en consultation. A noter qu'elle ne prend pas en compte les éventuelles modifications apportées au fichier de structure ou aux fichiers joints.

Ce paramètre concerne à la fois les sauvegardes de l'application et les sauvegardes de l'historique.

- Effacer la sauvegarde la plus ancienne avant sauvegarde / après sauvegarde : cette option n'est utilisée que si l'option "Conserver uniquement les N derniers fichiers de sauvegarde" est cochée. Elle vous permet de spécifier si 4D doit commencer par effacer l'archive la plus ancienne avant d'effectuer une sauvegarde (option avant) ou si

l'effacement doit être effectué après la sauvegarde (option après). Pour que ce mécanisme fonctionne, l'archive la plus ancienne ne doit pas avoir été renommée ou déplacée.

- Si la sauvegarde échoue : cette option permet de définir le mécanisme de prise en charge des échecs des sauvegardes (sauvegarde impossible). Lorsqu'une sauvegarde est impossible, 4D permet d'effectuer une nouvelle tentative.
 - Réessayer à la nouvelle date et heure programmée : cette option n'a de sens que dans le cadre de sauvegardes automatiques périodiques. Elle revient à annuler la sauvegarde ayant échoué. Une erreur est générée.
 - Réessayer dans N seconde(s), minute(s) ou heure(s) : lorsque cette option est cochée, une nouvelle tentative de sauvegarde est effectuée à l'issue du délai défini. Ce mécanisme permet d'anticiper certaines circonstances bloquant la sauvegarde. Vous pouvez fixer un délai en secondes, minutes ou heures à l'aide du menu correspondant. Si la nouvelle tentative échoue également, une erreur est générée et l'échec est inscrit dans les zones de statut de la dernière sauvegarde et dans le Journal des sauvegardes.
 - Annuler l'opération au bout de N tentatives : ce paramètre permet de définir le nombre de fois que le module de sauvegarde réessaiera de lancer la sauvegarde en cas d'échec. Si, à l'issue du nombre d'essais défini, la sauvegarde n'a pas pu être effectuée, elle est annulée et l'erreur 1401 est générée ("Le nombre maximal de tentatives de sauvegarde est atteint, la sauvegarde automatique est temporairement désactivée"). Dans ce cas, aucune nouvelle sauvegarde automatique ne sera lancée tant que l'application n'aura pas été redémarrée ou qu'une sauvegarde manuelle n'aura été effectuée avec succès. Ce paramètre est utile notamment pour éviter qu'en cas d'impossibilité prolongée de la sauvegarde (nécessitant une intervention humaine), l'application n'effectue inutilement de nombreuses tentatives au détriment de ses performances. Par défaut, ce paramètre n'est pas coché.

4D considère qu'une sauvegarde a échoué si l'application n'était pas lancée au moment théorique de la sauvegarde automatique périodique.

Archive

Ces options s'appliquent aux fichiers de sauvegarde principaux et aux fichiers de sauvegarde de l'historique.

- Taille du segment (Mo) 4D vous permet de segmenter les archives, c'est-à-dire de les découper en morceaux de taille fixe. Ce fonctionnement permet par exemple de stocker une sauvegarde sur plusieurs volumes (DVDs, usb, etc.). Au moment de la restitution, 4D fusionnera automatiquement les segments. Chaque segment est appelé MonApplication[xxxx-yyyy].4BK, où xxxx représente le numéro de la sauvegarde et yyyy celui du segment. Par exemple, les trois segments de la sixième sauvegarde de la base MonApplication seront appellés MonApplication[0006-0001].4BK, MonApplication[0006-0002].4BK et MonApplication[0006-0003].4BK. Le menu Taille du segment est une combo box permettant de définir la taille en Mo de chaque segment de sauvegarde. Vous pouvez choisir une des tailles prédéfinies ou saisir une taille spécifique entre 0 et 2048. Si vous passez 0, aucune segmentation n'est effectuée (équivaut à la valeur Aucune).
- Taux de compression Par défaut, les sauvegardes sont compressées par 4D. En contrepartie, la phase de compression des fichiers peut ralentir sensiblement les sauvegardes dans le cas de la manipulation de gros volumes de données. L'option Taux de compression vous permet d'ajuster la compression :
 - Aucun : aucune compression n'est appliquée. La sauvegarde peut être sensiblement plus rapide mais les fichiers d'archives sont plus volumineux sur le disque.
 - Normal (par défaut) : cette option constitue un compromis vitesse de sauvegarde/taille des archives.
 - Elevé : le taux de compression maximal est appliqué aux archives. Les fichiers d'archives prennent le moins de place possible sur le disque mais la sauvegarde peut être sensiblement ralenti.
- Taux d'entrelacement et Taux de redondance 4D peut générer des archives à l'aide d'algorithmes spécifiques, basés sur des mécanismes d'optimisation (entrelacement) et de sécurisation (redondance). Vous pouvez paramétrier ces mécanismes en fonction de vos besoins. Les menus associés à ces options vous permettent de choisir un taux Faible, Moyen, Elevé ou Aucun (défaut).
 - Taux d'entrelacement : l'entrelacement consiste à stocker les données dans des secteurs non contigus afin de limiter les risques en cas d'endommagement des secteurs. Plus le taux est élevé, plus la sécurité est élevée ; en contrepartie, le traitement des données consomme davantage de mémoire.
 - Taux de redondance : la redondance permet de sécuriser les données présentes dans un fichier en répétant plusieurs fois les mêmes informations. Plus le taux est élevé, plus le fichier est sécurisé, mais plus le stockage est lent et la taille du fichier importante.

Restitution automatique

- Restituer la dernière sauvegarde si la base est endommagée : lorsque cette option est cochée, le programme déclenche automatiquement la restitution du fichier de données de la dernière sauvegarde valide de l'application s'il détecte une anomalie (fichier corrompu par exemple) lors du lancement de l'application. Aucune intervention de l'utilisateur n'est requise ; l'opération est cependant consignée dans le Journal des sauvegardes.
- Intégrer le dernier historique si la base est incomplète : lorsque cette option est cochée, le programme intègre automatiquement l'historique lors de l'ouverture ou de la restitution de l'application.
 - Lors de l'ouverture d'une application, l'historique courant est automatiquement intégré si 4D détecte que des opérations stockées dans l'historique ne sont pas présentes dans les données. Cette situation se produit par exemple lorsqu'une panne de courant a lieu alors que des opérations non encore écrites sur le disque se trouvaient dans le cache de données.
 - Lors de la restitution de l'application, si le fichier d'historique courant ou un fichier de sauvegarde d'historique ayant le même numéro que le fichier de sauvegarde est stocké dans le même dossier, 4D examine son contenu. S'il contient des opérations non présentes dans le fichier de données, le programme l'intègre automatiquement.

Aucune boîte de dialogue n'est présentée à l'utilisateur, l'opération est entièrement automatique. Le but est de faciliter au maximum la remise en route de l'exploitation. L'opération est consignée dans le Journal des sauvegardes.

Dans le cas d'une restauration automatique, seuls les éléments suivants sont restaurés: - fichier .4DD - fichier .4DIndx - fichier .4DSyncData - fichier .4DSyncHeader - dossier External Data

Si vous souhaitez obtenir les fichiers joints ou les fichiers de projet, vous devez effectuer une [restauration manuelle](#).

Fichier d'historique (.journal)

Une application exploitée de manière continue enregistre en permanence des modifications, des ajouts ou des suppressions d'enregistrements. Réaliser des sauvegardes régulières des données est important mais ne permet pas, en cas d'incident, de récupérer les données saisies depuis la dernière sauvegarde. Pour répondre à ce besoin, 4D dispose d'un outil particulier : le fichier d'historique. Ce fichier permet d'assurer la sécurité permanente des données.

En outre, 4D travaille en permanence avec un cache de données situé en mémoire. Ce principe permet d'accélérer le fonctionnement des applications ; en effet, les accès mémoire sont bien plus rapides que les accès disque. Ce principe permet d'accélérer le fonctionnement des applications ; en effet, les accès mémoire sont bien plus rapides que les accès disque. Si un incident survient sur l'application avant que les données stockées dans le cache aient pu être écrites sur le disque, vous devrez intégrer le fichier d'historique courant afin de récupérer entièrement l'application.

Enfin, 4D dispose d'une fonction d'analyse du contenu du fichier d'historique, permettant également de faire remonter en arrière les opérations exécutées sur les données de l'application. Ces fonctions sont accessibles via le CSM : reportez-vous aux sections [Page Analyse d'activités](#) et [Page Retour arrière](#).

Fonctionnement du fichier d'historique

L'historique généré par 4D se présente sous la forme d'un fichier dans lequel toutes les opérations effectuées sur les données des tables journalisées de l'application viennent s'inscrire séquentiellement. Par défaut, toutes les tables sont journalisées, c'est-à-dire incluses dans l'historique, mais vous pouvez en désélectionner certaines via la propriété **Inclure** dans le fichier d'historique.

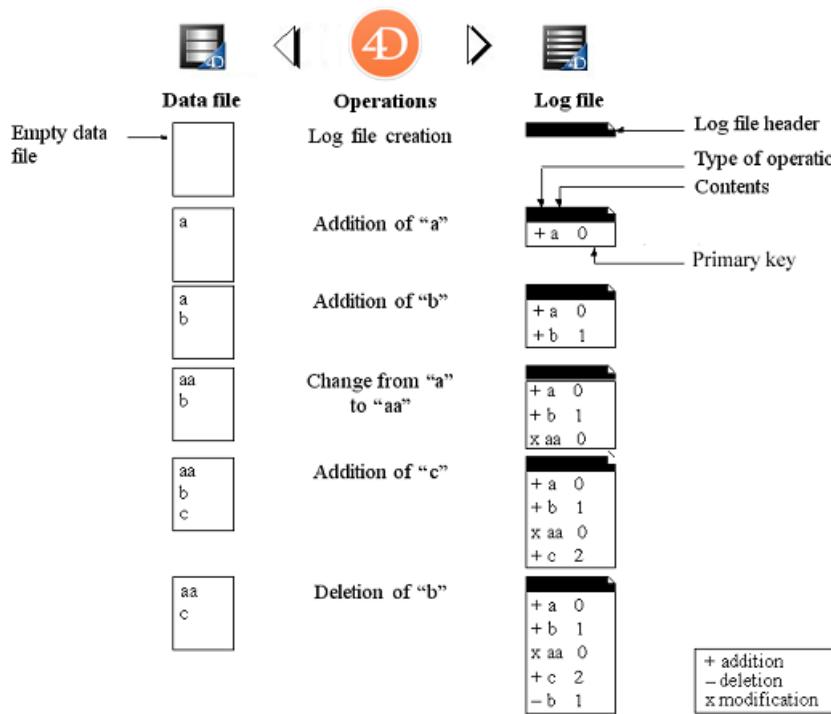
Ainsi, chaque opération effectuée par un utilisateur provoque deux actions simultanées : une première sur le fichier de données (l'instruction est exécutée normalement) et une seconde dans le fichier d'historique (la description de l'opération y est enregistrée). Le fichier d'historique se construit de manière indépendante, sans gêner ni ralentir le travail de l'utilisateur. Une application ne peut travailler qu'avec un seul fichier d'historique à la fois. Le fichier d'historique enregistre les types d'opérations suivants :

- Ouvertures et fermetures du fichier de données,
- Ouvertures et fermetures de process (contextes),
- Ajouts d'enregistrements et de BLOBs,
- Modifications d'enregistrements,
- Suppressions d'enregistrements,
- Créations et fermetures de transactions.

Pour plus d'informations sur ces actions, reportez-vous à la section [Page Analyse d'activités](#) du CSM.

Le fichier d'historique est géré par 4D. Il prend en compte indifféremment toutes les opérations affectant le fichier de données, que ce soient des manipulations effectuées par un utilisateur, des méthodes 4D, le moteur SQL, des plug-ins, un navigateur Web ou une application mobile.

Ce schéma résume le principe général de fonctionnement du fichier d'historique :



Le fichier d'historique courant est automatiquement sauvegardé avec le fichier de données courant. Ce mécanisme procure deux avantages principaux :

- Eviter la saturation du disque accueillant le fichier d'historique. En effet, sans sauvegarde, l'historique grossirait indéfiniment au fur et à mesure de l'exploitation de la base et finirait par saturer votre disque. A chaque sauvegarde du fichier de données, 4D ou 4D Server ferme le fichier d'historique courant et débute immédiatement un nouveau fichier vide, évitant ainsi le risque de saturation. L'ancien fichier d'historique est alors archivé puis éventuellement détruit, conformément au mécanisme des jeux de sauvegarde.
- Conserver les fichiers d'historique correspondant aux sauvegardes, afin de pouvoir analyser ou réparer a posteriori une application. En effet, l'intégration du fichier d'historique ne peut se faire que dans l'application qui lui correspond. Il est donc important, pour pouvoir intégrer correctement un fichier d'historique dans une sauvegarde, de disposer de sauvegardes et d'historiques archivés simultanément.

Créer le fichier d'historique

Par défaut, toute application créée avec 4D utilise un fichier d'historique (option définie dans la Page Général des Préférences de 4D). Le fichier d'historique est nommé par défaut *data.journal* et est placé dans le dossier Data.

Vous pouvez à tout moment savoir si votre application utilise un fichier d'historique : l'option Utiliser le fichier d'historique est cochée dans la Page Sauvegarde/Configuration des Propriétés. Si vous avez désélectionné cette option ou si vous utilisez une application sans fichier d'historique et souhaitez mettre en place une stratégie de sauvegarde avec fichier d'historique, il vous appartient d'en créer un.

Pour créer un fichier d'historique :

1. Dans la Page Sauvegarde/Configuration des Propriétés de structure, cochez l'option Utiliser le fichier d'historique. Le programme affiche une boîte de dialogue standard de création de fichier. Par défaut, le fichier d'historique est baptisé *data.journal*.
2. Conservez le nom du fichier par défaut ou choisissez-en un autre et déterminez l'emplacement du fichier. Si vous disposez d'au moins deux disques durs, il est recommandé de placer le fichier d'historique sur un autre disque que celui contenant le projet d'application. Si le disque dur de l'application est perdu, vous pouvez toujours rappeler votre fichier d'historique.
3. Cliquez sur le bouton Enregistrer. Le disque et le nom du fichier d'historique ouvert s'affichent alors dans la zone "Utiliser le fichier d'historique" de la boîte de dialogue. Vous pouvez cliquer dans cette zone afin d'afficher un pop up menu contenant l'enchaînement des dossiers à partir du disque.
4. Validez la boîte de dialogue des Propriétés.

Pour que vous puissiez directement créer un fichier d'historique, les données doivent se trouver dans une des situations

suivantes :

- Le fichier de données est vierge,
- Vous venez d'effectuer une sauvegarde et aucune modification sur les données n'a encore été effectuée.

Si vous cliquez sur OK, la sauvegarde démarre immédiatement puis l'historique est activé. Si vous cliquez sur Annuler, la sauvegarde démarre immédiatement puis l'historique est activé. Si vous cliquez sur Annuler, la requête est enregistrée mais la création du fichier d'historique est différée et sera créée uniquement après la prochaine sauvegarde de l'application. Cette précaution est indispensable car il vous faudra, pour restituer une application après un éventuel incident, disposer d'une copie de l'application dans laquelle pourront s'intégrer les opérations enregistrées dans le fichier d'historique.

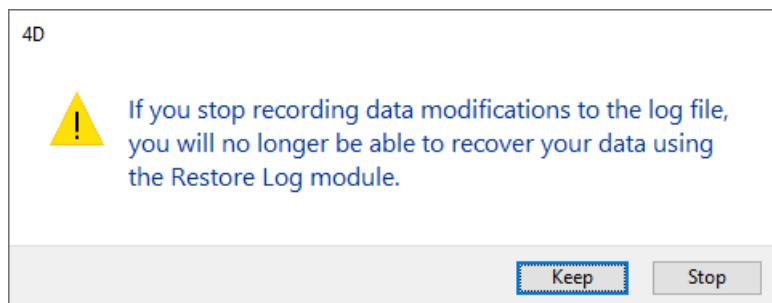
Sans autre manipulation de votre part, toutes les opérations effectuées sur les données s'inscriront dans ce fichier, et il sera utilisé également lors des ouvertures ultérieures de l'application.

Vous devrez créer un autre fichier d'historique si vous créez un nouveau fichier de données. Vous devrez désigner ou créer un autre fichier d'historique si vous ouvrez un autre fichier de données non associé à un fichier d'historique (ou si le fichier d'historique est manquant).

Fermer l'historique

Si vous souhaitez interrompre l'enregistrement du fichier d'historique courant, désélectionnez l'option Utiliser le fichier d'historique dans la Page Sauvegarde/Configuration des Propriétés.

4D affiche alors un message d'alerte afin d'attirer votre attention sur le fait que cette action vous prive de la sécurité apportée par le fichier d'historique :



Si vous cliquez sur le bouton Fermer, le fichier d'historique courant est immédiatement refermé (il n'est pas nécessaire que la boîte de dialogue des Propriétés soit ensuite validée).

Si vous souhaitez fermer votre fichier d'historique courant parce qu'il devient trop important, il est préférable d'effectuer une sauvegarde du fichier de données, ce qui entraînera la sauvegarde de l'historique.

4D Server : La commande `New log file` permet de refermer automatiquement l'historique courant et d'en démarrer un nouveau. Si le fichier d'historique devient inaccessible au cours de la session de travail, l'erreur 1274 est générée et 4D Server ne permet plus aux utilisateurs d'écrire ou de modifier des données. Lorsque le fichier d'historique est de nouveau accessible, il est nécessaire d'effectuer une sauvegarde.

Restitution

4D vous permet de récupérer l'intégralité des données d'une application en cas d'incident, quelles que soient ses causes. Deux catégories principales d'incidents peuvent se produire :

- L'arrêt inopiné de l'application pendant son exploitation. Cet incident peut se produire à cause d'une coupure de courant, la panne d'un élément du système, etc. Dans ce cas, en fonction de l'état courant du cache de données au moment de l'incident, la récupération de l'application peut nécessiter différentes opérations :
 - Si le cache était vide, l'application s'ouvre normalement. Toutes les modifications apportées à l'application ont été enregistrées. Ce cas ne nécessite aucune opération particulière.
 - Si le cache contenait des opérations, le fichier de données est intact mais il est nécessaire d'intégrer le fichier d'historique courant.
 - Si le cache était en cours d'écriture, le fichier de données est probablement endommagé. Il est nécessaire de restituer la dernière sauvegarde et d'intégrer le fichier d'historique courant.
- La perte de fichier(s) de l'application. Cet incident peut être causé par des secteurs défectueux sur le disque contenant l'application, un virus, une erreur de manipulation, etc. Il est nécessaire de restituer la dernière sauvegarde puis d'intégrer éventuellement l'historique courant. Pour savoir si une application a été endommagée à la suite d'un incident, il suffit de la relancer avec 4D. Le programme effectue un auto-diagnostic et précise les opérations de réparation à effectuer. En mode automatique, ces opérations sont effectuées directement, sans intervention de l'utilisateur. Si une stratégie de sauvegarde régulière a été mise en place, les outils de récupération de 4D vous permettront (dans la plupart des cas) de retrouver l'application dans l'état exact où elle se trouvait avant l'incident.

4D peut lancer automatiquement des procédures de récupération des applications après incident. Ces mécanismes sont gérés à l'aide de deux options accessibles dans la Page Sauvegarde/Sauvegarde & et Restitution de la fenêtre des Propriétés. Pour plus d'informations, reportez-vous au paragraphe [Restitution automatique](#)

. Si l'incident résulte d'une opération inappropriée sur effectuée sur les données (suppression d'un enregistrement par exemple), vous pouvez tenter de réparer le fichier de données en utilisant la fonction de "retour arrière" dans le fichier d'historique. Cette fonction est accessible dans la Page [Retour arrière](#) du CSM.

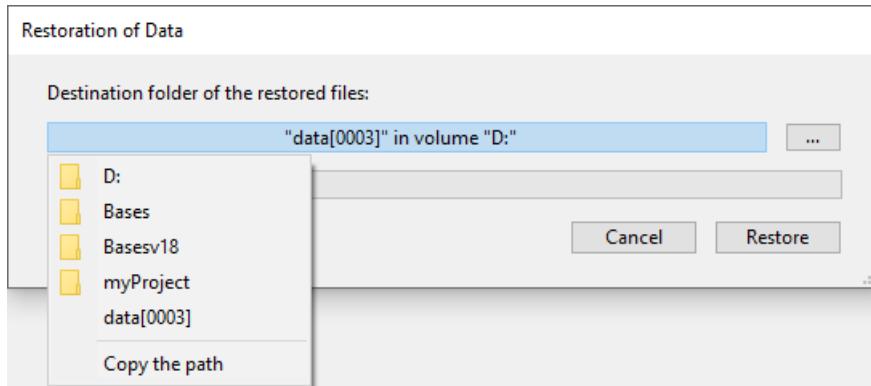
Restitution manuelle d'une sauvegarde (dialogue standard)

Vous pouvez restituer manuellement le contenu d'une archive générée par le module de sauvegarde. Une restitution manuelle peut être nécessaire par exemple pour restituer la totalité du contenu d'une archive (fichiers de structure et/ou fichiers joints inclus) ou à des fins de recherche sur des archives. La restitution manuelle peut éventuellement s'accompagner de l'intégration de l'historique courant.

La restitution manuelle des sauvegardes peut être réalisée soit via une boîte de dialogue standard d'ouverture de document, soit via la page "[Restitution](#)" du Centre de sécurité et de maintenance (CSM). La restitution via une boîte de dialogue standard permet de restituer n'importe quelle archive. En revanche, seules les archives associées à l'application ouverte peuvent être restituées.

Pour restituer manuellement une application via une boîte de dialogue standard :

1. Lancez l'application 4D et choisissez la commande Restituer... dans le menu Fichier. Il n'est pas obligatoire qu'un projet d'application soit ouvert. OU BIEN Exécutez la commande RESTORE depuis une méthode de 4D. Une boîte de dialogue standard d'ouverture de fichiers apparaît.
2. Désignez le fichier de sauvegarde (.4bk) ou le fichier de sauvegarde de l'historique (.4bl) à restituer et cliquez sur Ouvrir. Un boîte de dialogue apparaît, vous permettant de désigner l'emplacement auquel vous souhaitez que les fichiers soient restitués . Par défaut, 4D restitue les fichiers dans un dossier nommé "Nomarchive" (sans extension) placé à côté de l'archive. Vous pouvez afficher le chemin :



Vous pouvez également cliquer sur le bouton [...] et indiquer un autre emplacement.

3. Cliquez sur le bouton Restituer. 4D extrait tous les fichiers de la sauvegarde à l'emplacement défini. Si le fichier d'historique courant ou un fichier de sauvegarde d'historique ayant le même numéro que le fichier de sauvegarde est stocké dans le même dossier, 4D examine son contenu. S'il contient des opérations non présentes dans le fichier de données, le programme propose de l'intégrer. L'intégration est effectuée automatiquement si l'option d'intégration automatique de l'historique est cochée (cf. paragraphe [Restitution automatique](#)).

(Facultatif) Cliquez sur OK pour intégrer le fichier d'historique dans l'application restituée. Si la restitution et l'intégration se sont déroulées correctement, 4D affiche une boîte de dialogue indiquant que l'opération a réussi.

5. Cliquez sur OK.

Le dossier d'arrivée est affiché. Lors de la restitution, 4D place tous les fichiers sauvegardés dans ce dossier, quelle que soit la position des fichiers originaux sur le disque au moment de la sauvegarde. De cette façon, vous retrouverez plus facilement vos fichiers.

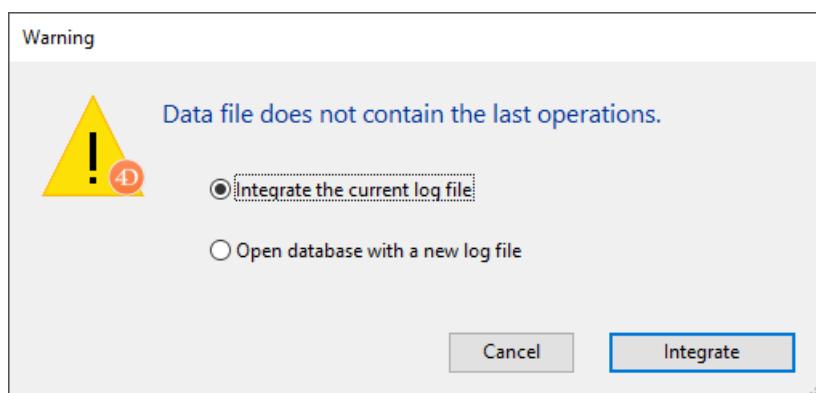
Tout contenu lié au fichier de données (dossier files et [Settings](#)) est automatiquement restauré dans un sous-dossier [Data](#) du dossier de destination.

Restitution manuelle d'une sauvegarde (CSM)

La [page Restitution](#) du Centre de sécurité et de maintenance (CSM) vous permet de restituer manuellement une archive de l'application courante.

Intégration manuelle de l'historique

Si vous n'avez pas coché l'option d'intégration automatique du fichier d'historique dans la page Restitution du CSM (cf. [Intégration successive de plusieurs fichiers d'historiques](#)), une boîte de dialogue d'alerte apparaît à l'ouverture de l'application lorsque 4D constate que le fichier d'historique contient plus d'opérations qu'il n'en a été effectué dans le fichier de données.



Pour que ce mécanisme fonctionne, 4D doit être en mesure d'accéder au fichier d'historique à son emplacement courant.

Vous pouvez choisir d'intégrer ou non les données de l'historique courant. Ne pas intégrer l'historique courant permet notamment d'éviter de reproduire des erreurs effectuées sur les données.

Aperçu

4D en mode local et remote ainsi que 4D Server disposent d'un moteur de serveur web intégré (aussi appelé serveur HTTP) qui vous permet de créer et publier des applications web afin de tirer le maximum de vos bases de données 4D.

Administration simplifiée

Vous pouvez démarrer ou arrêter la publication de l'application web à tout moment. Pour ce faire, il suffit de sélectionner une commande dans un menu ou d'exécuter une ligne de code.

Vous pouvez aisément suivre l'activité du serveur web 4D dans la fenêtre d'administration de 4D, ou via des [URLs spéciales](#).

Prêt à l'emploi

Le serveur web 4D crée automatiquement un dossier racine et une page d'accueil par défaut, disponibles immédiatement.

Sécurité

La sécurité des données est présente à tous les stades d'implémentation du serveur web 4D. Les niveaux de sécurité sont évolutifs, et les options les plus sécurisées sont généralement sélectionnées par défaut. La sécurité du serveur web 4D est basée sur les éléments suivants :

- Sandboxing via la définition d'un [dossier HTML racine](#) par défaut
- Contrôle de l'usage des ressources du serveur (par exemple, via l'option qui détermine [le nombre maximum de process web simultanés](#)).
- Contrôle du contenu exposé : Seul le contenu que vous exposez explicitement est disponible via des requêtes web directes ou des requêtes REST. Vous devez déclarer :
 - [Les méthodes projet](#) exposées via requêtes HTTP
 - [Les fonctions ORDA](#) exposées via requêtes REST
 - [Les tables et champs](#) que vous ne voulez pas rendre disponibles via requêtes REST
- Sandboxing via la définition d'un [dossier HTML racine](#) par défaut
- Contrôle de l'usage des ressources du serveur (par exemple, via l'option qui détermine [le nombre maximum de process web simultanés](#)).

Consultez le document [4D Security guide](#) pour une vue d'ensemble des fonctions de sécurité de 4D.

Sessions Utilisateur

protocole TLS (HTTPS),

Point d'accès pour requêtes REST

Le serveur web 4D permet d'accéder aux données stockées dans vos applications 4D via des requêtes REST. Les requêtes REST offrent un accès direct à toutes les opérations de bases de données telles que l'ajout, la lecture, la modification, l'organisation ou la recherche.

Les requêtes REST sont détaillées dans la section [serveur REST](#).

Extension des paramètres

Les requêtes REST sont détaillées dans la section [serveur REST](#).

Templates et URLs

Le serveur web 4D offre un accès aux données stockées dans vos applications 4D à travers des pages de template et des URLs spécifiques.

- Authentification : [fonctionnalités d'authentification](#) flexibles et personnalisables, basées sur des paramètres intégrés, ainsi que des Méthodes base de secours ([Sur authentification Web](#) pour le serveur web et [Sur authentification REST](#) pour le serveur REST),
- [Les URLs spécifiques](#) permettent à 4D d'être appelé pour exécuter tout type d'action. Ces URLs peuvent également être utilisées comme des actions de formulaire pour déclencher des traitements de données quand l'utilisateur poste des formulaires HTML.

Méthodes base dédiées

Le serveur web 4D offre un accès aux données stockées dans vos applications 4D à travers des pages de template et des URLs spécifiques.

Configuration

Les paramètres du serveur web 4D comprennent les paramètres de sécurité, les ports d'écoute, les chemins par défaut et diverses options couvrant toutes les fonctionnalités du serveur. 4D fournit des valeurs par défaut pour tous les paramètres.

Configurer les paramètres

Vous pouvez configurer les paramètres du serveur web 4D, en fonction de la portée et du serveur que vous souhaitez configurer :

Setting location	Portée	Serveur web concerné
objet webServer	Temporaire (session courante)	N'importe quel serveur web, y compris les serveurs Web de composants
WEB SET OPTION ou commande WEB XXX	Temporaire (session courante)	Serveur principal
Boîte de dialogue des Propriétés (pages Web)	Permanent (toutes les sessions, stocké sur le disque)	Serveur principal

Certains paramètres ne sont pas disponibles depuis tous les emplacements.

Cache

Peut être configuré via	Nom	Commentaires
Fenêtre de configuration	Page de configuration/Utilisation du cache Web 4D	
Fenêtre de configuration	Page de configuration/Taille du cache des pages	

Active et configure le cache des pages web.

Le serveur web 4D dispose d'un cache qui permet de charger les pages statiques, les images GIF, les images JPEG (<512 kb) et les feuilles de style (fichiers .css) en mémoire, au fur et à mesure qu'elles sont appelées. L'utilisation du cache permet d'augmenter considérablement les performances du serveur web lors de l'envoi de pages statiques. Le cache est partagé entre tous les processus web. Lorsque le cache est activé, le serveur Web 4D recherche d'abord dans le cache toute page statique demandée par le navigateur. S'il trouve la page, il l'envoie immédiatement. Sinon, 4D charge la page à partir du disque et la place dans le cache.

Vous pouvez modifier la taille du cache dans la zone Taille du cache des pages . La valeur à définir dépend du nombre et de la taille des pages statiques de votre site Web, ainsi que des ressources dont dispose la machine hôte.

Lorsque vous utilisez votre base de données web, vous pouvez vérifier les performances du cache en utilisant la commande WEB LIRE STATISTIQUES . Si, par exemple, vous remarquez que le taux d'utilisation du cache est proche de 100 %, vous pouvez envisager d'augmenter la taille qui lui a été allouée. Les URL [/4DSTATS] et [/4DHMLSTATS] vous permettent également d'obtenir des informations sur l'état du cache.

Dossier de certificat

Peut être configuré via	Nom	Commentaires
objet webServer	<code>certificateFolder</code>	La propriété Text, mais il peut s'agir d'un objet <code>4D.Folder</code> lorsqu'il est utilisé avec le paramètre <code>settings</code> de la fonction <code>start()</code> .

Dossier qui contient les fichiers de certificat TLS pour le serveur web.

Par défaut, avec 4D ou 4D Server, ces fichiers doivent être placés à côté du [dossier du projet](#).

Avec 4D à distance, ces fichiers doivent être placés dans le dossier des ressources locales de la base de données sur la machine distante (voir le paragraphe [Dossier base 4D Client](#) de la commande [Dossier 4D](#)). Vous devez copier ces fichiers manuellement sur la machine distante.

Les fichiers de certificat TLS sont `key.pem` (document contenant la clé de chiffrement privée) et `cert.pem` (document contenant le certificat).

Jeu de caractères

Peut être configuré via	Nom	Commentaires
objet webServer	<code>characterSet</code>	Entier long (MIBEnum) ou chaîne de caractères (nom)
<code>WEB SET OPTION</code>	<code>Web character set</code>	Entier long (MIBEnum) ou chaîne de caractères (nom)
Fenêtre de configuration	Page Options (II)/Jeu standard	Menu popup

Définit le jeu de caractères à utiliser par le serveur web 4D. La valeur par défaut dépend de la langue du système d'exploitation.

Ce paramètre est également utilisé pour générer des États Rapides au format HTML.

Suite cryptographique

Peut être configuré via	Nom	Commentaires
objet webServer	<code>cipherSuite</code>	Text

Suite cryptographique utilisée pour le protocole sécurisé. Fixe la priorité des algorithmes de chiffrement implémentés par le serveur web. Peut être une séquence de chaînes séparées par des deux-points (par exemple "ECDHE-RSA-AES128 -..."). Voir la [page des chiffrements](#) sur le site OpenSSL.

La liste de chiffrement par défaut utilisée par 4D peut être modifiée pour la session à l'aide de la commande `FIXER PARAMETRE BASE`, auquel cas la modification s'applique à l'ensemble de l'application 4D, y compris le serveur web, le serveur SQL, les connexions client/serveur, ainsi que le client HTTP et toutes les commandes 4D qui font appel au protocole sécurisé.

Paramètres CORS

Peut être configuré via	Nom	Commentaires
objet webServer	CORSSettings	Collection d'objets (Liste des hôtes et méthodes autorisées pour le service CORS)
WEB SET OPTION	Web CORS settings	Collection d'objets (Liste des hôtes et méthodes autorisées pour le service CORS)
Fenêtre de configuration	Propriétés > Web > Options (II) > Noms de domaine et Méthodes HTTP autorisées	Cliquez sur le bouton [+] pour ajouter un nom de domaine autorisé et sa ou ses méthodes

Liste des hôtes et méthodes autorisées pour le service CORS.

Noms de domaine

Nom de domaine ou adresse IP à partir desquels les pages externes sont autorisées à envoyer des demandes de données au serveur via CORS. Plusieurs attributs de domaine peuvent être ajoutés pour créer une liste blanche. Plusieurs syntaxes sont supportées :

- 192.168.5.17:8081
- 192.168.5.17
- 192.168.*
- 192.168.*:8081
- <http://192.168.5.17:8081>
- http://*.myDomain.com
- <http://myProject.myDomain.com>
- *.myDomain.com
- myProject.myDomain.com
- *

Méthodes HTTP autorisées (propriété méthodes)

Méthodes HTTP acceptées pour l'hôte CORS correspondant. Les méthodes HTTP suivantes sont prises en charge :

- GET
- HEAD
- POST
- PUT
- DELETE
- OPTIONS
- TRACE
- PATCH

Séparez chaque méthode par un ";" (ex : "post;get"). Si Méthodes est vide, null ou non défini, toutes les méthodes sont activées.

Voir aussi

[Activer CORS](#)

Debug log

Peut être configuré via	Nom	Commentaires
objet webServer	debugLog	number
WEB SET OPTION	Web debug log	number

Status of the HTTP request log file of the web server ([HTTPDebugLog_nn.txt](#), stored in the "Logs" folder of the application -- nn is the file number). Il est utile pour déboguer les problèmes liés au serveur Web. Il enregistre chaque

demande et chaque réponse en mode brut (raw). Les requêtes sont enregistrées dans leur totalité (en-têtes compris). Les parts du body peuvent optionnellement être enregistrées.

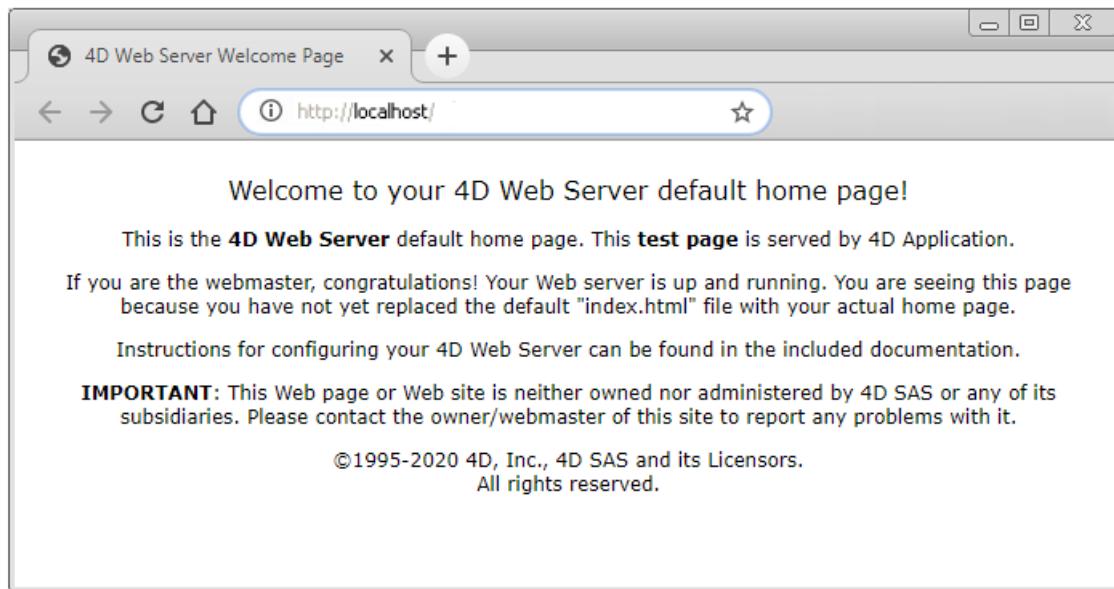
Valeur	Constante	Description
0	wdl disable	Les debug logs Web HTTP sont désactivés
1	wdl enable without body	Web HTTP debug log is enabled without body parts (body size is provided in this case)
3	wdl enable with response body	Web HTTP debug log is enabled with body part in response only
5	wdl enable with request body	Web HTTP debug log is enabled with body part in request only
7	wdl enable with all body parts	Web HTTP debug log is enabled with body parts in response and request

Page d'accueil par défaut

Peut être configuré via	Nom	Commentaires
objet webServer	defaultHomepage	Text
WEB SET HOME PAGE		Peut être différente pour chaque web process
Fenêtre de configuration	Configuration > Page d'accueil par défaut	

Désigne une page comme page d'accueil par défaut pour le serveur web. Cette page peut être statique ou [semi-dynamic].

Par défaut, quand le serveur web est lancé pour la première fois, 4D crée une page d'accueil appelée "index.html" et la place dans le dossier HTML racine. Si vous ne modifiez pas cette configuration, n'importe quel browser se connectant au serveur web obtiendra cette page :



Vous pouvez désigner une autre page comme page d'accueil par défaut en entrant son chemin d'accès :

- Le chemin est relatif au [dossier HTML racine](#),
- Le chemin est exprimé avec la syntaxe POSIX (les dossiers sont séparés par un slash (/)),
- Le chemin ne doit pas commencer ou finir par un slash.

Par exemple, pour que la page d'accueil par défaut soit "MyHome.htm", si elle se trouve dans le dossier "Web" (lui-même situé dans le dossier racine HTML par défaut), utilisez "Web/MyHome.htm".

Si vous ne spécifiez aucune page d'accueil par défaut, la méthode base `On Web Connection` est appelée. Il vous appartient de traiter la demande de manière procédurale.

Activer CORS

Peut être configuré via	Nom	Commentaires
objet webServer	<code>CORSEnabled</code>	True pour activer CORS (False par défaut).
<code>WEB SET OPTION</code>	<code>Web CORS enabled</code>	0 (désactivé, par défaut) ou 1 (activé)
Fenêtre de configuration	Page Options (II) > Activer CORS	Décoché par défaut

Le serveur Web 4D implémente le cross-origin resource sharing (CORS) pour permettre à des pages Web spécifiques servies à partir d'un autre domaine d'accéder aux ressources de l'application Web actuelle via des appels XHR, par exemple via REST. Pour des raisons de sécurité, les requêtes "cross-domain" sont interdites par défaut au niveau du navigateur. Pour des raisons de sécurité, les requêtes "cross-domain" sont interdites par défaut au niveau du navigateur. Lorsqu'elle l'option est activée, les appels XHR (par exemple, les requêtes REST) provenant de pages Web situées en dehors du domaine peuvent être autorisés dans votre application (vous devez définir la liste des adresses autorisées dans la liste de domaines CORS, voir Paramètres CORS ci-dessous).

Lorsqu'elle est désactivée (par défaut), toutes les demandes intersites envoyées avec CORS sont ignorées.

Pour plus d'informations sur CORS, veuillez consulter la [page de partage de ressources cross-origin](#) sur Wikipedia.

Voir aussi

[Paramètres CORS](#)

Activer HTTP

Peut être configuré via	Nom	Commentaires
objet webServer	<code>HTTPEnabled</code>	boolean
<code>WEB SET OPTION</code>	<code>Web HTTP enabled</code>	
Fenêtre de configuration	Configuration > Activer HTTP	

Indique si le web server accepte des connexions non sécurisées.

Activer HTTPS

Peut être configuré via	Nom	Commentaires
objet webServer	<code>HTTPSEnabled</code>	boolean
<code>WEB SET OPTION</code>	<code>Web HTTPS enabled</code>	
Fenêtre de configuration	Configuration > Activer HTTPS	

Statut de la communication via HTTPS. Cette option est décrite dans [cette section](#).

Activer HSTS

Peut être configuré via	Nom	Commentaires
objet webServer	<code>HSTSEnabled</code>	Booléen, True pour activer le HSTS (False par défaut)
<code>WEB SET OPTION</code>	<code>Web HSTS enabled</code>	0 (désactivé, par défaut) ou 1 (activé)

État de HTTP Strict Transport Security (HSTS).

Lorsque [HTTPS est activé](#), n'oubliez pas que si [HTTP est également activé](#), le navigateur peut toujours basculer entre HTTPS et HTTP (par exemple, dans la zone URL du navigateur, l'utilisateur peut remplacer "https" par "http"). Pour interdire les redirections http, vous pouvez [désactiver le HTTP](#), cependant dans ce cas un message d'erreur est affiché lors des requêtes HTTP du client.

HSTS permet au serveur web 4D de déclarer que les navigateurs ne doivent interagir avec lui que par des connexions HTTPS sécurisées. Une fois activé, le serveur Web 4D ajoutera automatiquement des informations relatives au HSTS à tous les en-têtes des réponses. Les navigateurs enregistreront les informations HSTS la première fois qu'ils recevront une réponse du serveur web 4D, puis toutes les futures demandes HTTP seront automatiquement transformées en demandes HTTPS. La durée de stockage de ces informations par le navigateur est spécifiée avec le paramètre Web HSTS max age.

Activer le HSTS requiert que HTTPS soit [activé](#) sur le serveur. [Le HTTP](#) doit également être activé pour permettre les connexions initiales du client.

Vous pouvez vérifier le mode de connexion utilisé en utilisant la commande `WEB Is secured connection`.

HSTS Max Age

Peut être configuré via	Nom	Commentaires
objet webServer	HSTSMaxAge	nombre en secondes
WEB SET OPTION	<code>Web HSTS max age</code>	nombre en secondes

Spécifie la durée maximale (en secondes) d'activation de HSTS pour chaque nouvelle connexion client. Ces informations sont stockées côté client pendant la durée spécifiée. Valeur par défaut : 63072000 (2 ans)

Avertissement: Une fois le HSTS activé, les connexions des clients continueront à utiliser ce mécanisme pendant la durée spécifiée. Lorsque vous testez vos applications, il est recommandé de définir une courte durée pour pouvoir passer du mode de connexion sécurisé au mode non sécurisé si nécessaire.

Niveau de compression

Peut être configuré via	Nom	Commentaires
objet webServer	HTTPCompressionLevel	
WEB SET OPTION	<code>Web HTTP compression level</code>	S'applique au Web et au service Web

Niveau de compression pour tous les échanges HTTP compressés pour le serveur web 4D (requêtes clients ou réponses serveur). Cette option vous permet d'optimiser les échanges en privilégiant soit la vitesse d'exécution (moins de compression), soit la quantité de compression (moins de vitesse). Le choix d'une valeur dépend de la taille et du type de données échangées.

Passez une valeur de 1 à 9 où 1 est la compression la plus rapide et 9 la plus élevée. Vous pouvez également passer -1 pour un compromis entre vitesse et taux de compression. Par défaut, le niveau de compression est de 1 (compression plus rapide).

Seuil de compression HTTP

Peut être configuré via	Nom	Commentaires
objet webServer	HTTPCompressionThreshold	
WEB SET OPTION	Web HTTP compression threshold	

Dans le cadre des échanges HTTP optimisés, seuil de taille des requêtes en dessous duquel les échanges ne doivent pas être compressés. Ce paramètre est utile pour éviter de perdre du temps machine en compressant les petits échanges.

Comme valeur, passez la taille exprimée en octets. Par défaut, le seuil de compression est fixé à 1024 octets.

Port HTTP

Peut être configuré via	Nom	Commentaires
objet webServer	HTTPPort	number
WEB SET OPTION	Web port ID	
Fenêtre de configuration	Configuration > Port HTTP	

Numéro de port IP (TCP) d'écoute pour HTTP. Par défaut, 4D publie une application Web sur le port HTTP normal (port TCP), qui est le port 80. Si ce port est déjà utilisé par un autre service Web, vous devez modifier le port HTTP utilisé par 4D pour ce projet.

Sous macOS, la modification du port HTTP permet de lancer le serveur web 4D sans être l'utilisateur root de la machine (voir [macOS HelperTool](#)).

À partir d'un navigateur Web, vous devez inclure le numéro de port HTTP par défaut dans l'adresse que vous saisissez pour vous connecter à l'application Web. L'adresse doit comporter un suffixe composé de deux points suivis du numéro de port. Par exemple, si vous utilisez le port HTTP numéro 8080, indiquez "123.4.567.89:8080".

Avertissement : Si vous utilisez des numéros de port TCP autres que ceux par défaut (80 pour HTTP standard et 443 pour HTTPS), veillez à ne pas utiliser des numéros de port utilisés par défaut pour d'autres services que vous pourriez vouloir utiliser simultanément. Par exemple, si vous prévoyez également d'utiliser le protocole FTP sur votre machine de serveur Web, n'utilisez pas les ports TCP 20 et 21, qui sont les ports par défaut pour ce protocole. Les numéros de ports inférieurs à 256 sont réservés à des services connus, et les numéros de ports de 256 à 1024 sont réservés à des services spécifiques issus des plateformes UNIX. Pour une sécurité maximale, spécifiez un numéro de port au-delà de ces intervalles (par exemple, dans les 2000 ou 3000).

Si vous spécifiez 0, 4D utilisera le numéro de port HTTP 80 par défaut.

HTTP Trace

Peut être configuré via	Nom	Commentaires
objet webServer	HTTPTrace	Booléen, false par défaut
WEB SET OPTION	Web HTTP TRACE	Entier long, 0 par défaut (désactivé)

Activation de la méthode HTTP TRACE dans le serveur web 4D. Pour des raisons de sécurité, le serveur web 4D rejette par défaut les demandes HTTP TRACE avec une erreur 405. Si nécessaire, vous pouvez activer la méthode HTTP TRACE, auquel cas le serveur Web 4D répond aux demandes HTTP TRACE avec la request line, l'en-tête et le body.

Port HTTPS

Peut être configuré via	Nom	Commentaires
objet webServer	HTTPSPort	number

| WEB SET OPTION | Web HTTPS port ID ||

|Settings dialog box|[Configuration page/HTTPS Port](#)||

Numéro de port IP d'écoute pour les connections HTTP via TLS. La valeur par défaut est 443 (valeur standard). Voir aussi [HTTP Port](#) pour plus d'informations sur les numéros de port.

Conservation des process inactifs

Peut être configuré via	Nom	Commentaires
objet webServer	inactiveProcessTimeout	
WEB SET OPTION	Web inactive process timeout	
Fenêtre de configuration	Options (I) > Conservation des Process inactifs	Curseur

Durée de vie (en minutes) des process inactifs associés aux sessions. À la fin du délai d'attente (tiemout), le process est tué sur le serveur, la méthode base `On Web Close Process` est appelée, puis le contexte de session est détruit.

Valeur par défaut : 480 minutes (passez 0 pour restaurer la valeur par défaut)

Durée de vie des sessions inactives

Peut être configuré via	Nom	Commentaires
objet webServer	inactiveSessionTimeout	
WEB SET OPTION	Web inactive session timeout	

Durée de vie (en minutes) des sessions inactives (durée définie dans le cookie). À la fin de cette période, le cookie de session expire et n'est plus envoyé par le client HTTP.

Valeur par défaut : 480 minutes (passez 0 pour restaurer la valeur par défaut)

Adresse IP d'écoute

Peut être configuré via	Nom	Commentaires
objet webServer	IPAddressToListen	
WEB SET OPTION	Web IP address to listen	
Fenêtre de configuration	Configuration > Adresse IP	Menu popup

Adresses IP (chaînes) sur lesquelles le serveur web 4D recevra les requêtes HTTP (4D local et 4D Server).

Par défaut, aucune adresse spécifique n'est définie (Valeur Any dans la boîte de dialogue Paramètres), cela signifie que le serveur répond à toutes les adresses IP. Lorsque vous désignez une adresse spécifique, le serveur ne répond qu'aux demandes envoyées à cette adresse. Cette fonction est conçue pour être utilisée avec les serveurs Web 4D situés sur des machines ayant plusieurs adresses TCP/IP. Par exemple, c'est régulièrement le cas dans des contextes d'hébergement.

Valeurs possibles: Chaîne de caractères représentant l'adresse IP. Les formats IPv6 (e.g.

"2001:0db8:0000:0000:ff00:0042:8329") et IPv4 (e.g. "123.45.67.89") sont tous les deux supportés.

À propos du support de l'IPv6

- Aucun avertissement lorsque le port TCP est occupé.

Lorsque le serveur est configuré pour répondre sur les adresses IP "Any", si le port TCP est utilisé par une autre application, cela n'est pas indiqué au démarrage du serveur. En fait, le serveur 4D ne détecte pas d'erreur dans ce cas car le port reste libre sur l'adresse IPv6. Cependant, il n'est pas possible d'y accéder en utilisant l'adresse IPv4 de la machine, ni au moyen de l'adresse locale : 127.0.0.1.

Si votre serveur 4D ne semble pas répondre sur le port défini, vous pouvez tester l'adresse [::1] sur la machine serveur (équivalent à 127.0.0.1 pour IPv6, ajoutez [:portNum] pour tester un autre numéro de port). Si 4D répond, il est probable qu'une autre application utilise le port en IPv4.

- Adresses IPv4 mappées en IPv6

. Pour normaliser le traitement, 4D fournit une représentation hybride standard des adresses IPv4 en IPv6. Ces adresses sont écrites avec un préfixe de 96 bits au format IPv6, suivi de 32 bits écrits dans la notation décimale à point d'IPv4. Par exemple, ::ffff:192.168.2.34 représente l'adresse IPv4 192.168.2.34.

- Indication des numéros de port

. Comme la notation IPv6 utilise les deux-points (:), l'ajout de numéros de port peut entraîner une certaine confusion, par exemple :

```
2001:0DB8::85a3:0:ac1f:8001 // adresse IPv6  
2001:0DB8::85a3:0:ac1f:8001:8081 // adresse IPv6 avec port 8081
```

Pour éviter cette confusion, nous recommandons d'utiliser la notation [] lorsque vous combinez une adresse IPv6 avec un numéro de port. Par exemple:

```
[2001:0DB8::85a3:0:ac1f:8001]:8081 //IPv6 address with port 8081
```

Keep Session

Peut être configuré via	Nom	Commentaires
objet webServer	keepSession	
WEB SET OPTION	Web keep session	
Fenêtre de configuration	Options (I) page/Legacy sessions (single process sessions)	uniquement dans les projets convertis

Statut de la gestion de l'ancienne session pour le serveur Web 4D (obsolète).

Lorsque cette option est cochée, l'option "réutilisation des contextes temporaires" est automatiquement cochée (et verrouillée).

Enregistrement des logs

Peut être configuré via	Nom	Commentaires
objet webServer	logRecording	
WEB SET OPTION	Web log recording	
Fenêtre de configuration	Page log (type)	Menu popup

Démarre ou arrête l'enregistrement des requêtes reçues par le serveur Web 4D dans le fichier *logweb.txt* et définit son format. Par défaut, les requêtes ne sont pas enregistrées (0/Pas de journal). Lorsqu'il est activé, le fichier *logweb.txt* est automatiquement placé dans le dossier Logs.

Ce paramètre vous permet de sélectionner le format de ce fichier. Valeurs possibles :

Valeur	Nom du format	Description
0	Pas de journal	Par défaut
1	CLF	Format de journal commun - Chaque ligne du fichier représente une requête, telle que : host rfc931 user [DD/MMM/YYYY:HH:MM:SS] "request" state length - Chaque champ est séparé par un espace et chaque ligne se termine par la séquence CR/LF.
2	DLF	Combined Log Format - Similaire au format NSI mais ajoute deux champs HTTP supplémentaires à la fin de chaque requête : Referer et User-agent.
3	ELF	Extended Log Format - À personnaliser dans la boîte de dialogue des Propriétés
4	WLF	Webstar Log Format- À personnaliser dans la boîte de dialogue des Propriétés

Les formats 3 et 4 sont des formats personnalisés dont le contenu doit être défini au préalable dans la boîte de dialogue des [Paramètres](#). Si vous utilisez l'un de ces formats sans qu'aucun de ses champs n'ait été sélectionné sur cette page, le fichier d'enregistrement des logs ne sera pas généré.

Process Web simultanés maxi

Peut être configuré via	Nom	Commentaires
objet webServer	<code>maxConcurrentProcesses</code>	
WEB SET OPTION	<code>Web max concurrent processes</code>	
Fenêtre de configuration	Options (I) > Process Web simultanés maxi	

Strictly high limit of concurrent web processes that can be simultaneously open on the server when no sessions or legacy sessions are used (scalable sessions support an [unlimited number](#) of preemptive processes). Ce paramètre permet d'éviter une saturation du serveur lorsqu'il reçoit un nombre important de requêtes. Ce paramètre permet d'éviter une saturation du serveur lorsqu'il reçoit un nombre important de requêtes. Lorsque le nombre maximal de processus Web simultanés (moins un) est atteint, 4D ne crée plus de nouveaux process et envoie le statut HTTP [503 – Service indisponible](#) à toutes les nouvelles requêtes.

La valeur par défaut est 100. Vous pouvez la fixer entre 10 et 32000.

Taille de requête maximale

Peut être configuré via	Nom	Commentaires
objet webServer	<code>maxRequestSize</code>	
WEB SET OPTION	<code>Web maximum requests size</code>	

Taille maximale (en octets) des requêtes HTTP entrantes (POST) que le serveur Web est autorisé à traiter. Par défaut, la valeur est de 2 000 000, c'est-à-dire un peu moins de 2 Mo. Le dépassement de la valeur maximale (2 147 483 648) indique, en pratique, qu'aucune limite n'est fixée.

Cette limite est utilisée pour éviter la saturation du serveur Web en raison de requêtes entrantes trop volumineuses. Cette limite est utilisée pour éviter la saturation du serveur Web en raison de requêtes entrantes trop volumineuses.

Valeurs possibles: 500 000 - 2147483648.

Nombre maximal de sessions

Peut être configuré via	Nom	Commentaires
objet webServer	<code>maxSessions</code>	
WEB SET OPTION	Web max sessions	

Nombre maximum de sessions simultanées. Lorsque vous atteignez la limite, la session la plus ancienne est fermée (et la méthode base `On Web Close Process` est appelée) si le serveur Web doit en créer une nouvelle. Le nombre de sessions simultanées ne peut pas dépasser le [nombre maximal de process Web](#) (100 par défaut).

Valeur par défaut : 100 (passez 0 pour restaurer la valeur par défaut).

Version TLS minimale

Peut être configuré via	Nom	Commentaires
objet webServer	<code>minTLSVersion</code>	number

Version TLS minimale acceptée pour les connexions. Les tentatives de connexion de clients prenant en charge uniquement les versions inférieures au minimum seront rejetées.

Valeurs possibles :

- 1 = TLSv1_0
- 2 = TLSv1_1
- 3 = TLSv1_2 (par défaut)
- 4 = TLSv1_3

En cas de modification, le serveur doit être redémarré pour utiliser la nouvelle valeur.

La version TLS minimale utilisée par 4D peut être modifiée pour la session à l'aide de la commande `SET DATABASE PARAMETER`, auquel cas la modification s'applique à l'ensemble de l'application 4D, y compris le serveur Web, le serveur SQL et les connexions client/serveur.

Nom

Peut être configuré via	Nom	Commentaires
objet webServer	<code>name</code>	

Nom de l'application de serveur Web. Pratique lors du démarrage des serveurs Web.

Version OpenSSL

Peut être configuré via	Nom	Commentaires
objet webServer	<code>openSSLVersion</code>	Lecture seule

Version de la bibliothèque OpenSSL utilisée.

Perfect Forward Secrecy

Peut être configuré via	Nom	Commentaires
objet webServer	<code>perfectForwardSecrecy</code>	Booléen, lecture seule

Vrai si le PFS est disponible sur le serveur web (voir la section [TLS](#)).

Réutiliser les contextes temporaires (en mode distant)

Peut être configuré via	Nom	Commentaires
Fenêtre de configuration	Options (I) > Process Web simultanés maxi	

This option is only available when No sessions option is checked.

Permet d'optimiser le fonctionnement du 4D Web Server en mode distant en réutilisant les process web créés pour le traitement des demandes web précédentes. En fait, le serveur web de 4D nécessite un process web spécifique pour le traitement de chaque requête web; en mode distant, lorsque nécessaire, ce processus se connecte à la machine du 4D Server afin d'accéder au moteur de données et de base de données. Il génère ainsi un contexte temporaire en utilisant ses propres variables, sélections, etc. Une fois la demande traitée, ce process est arrêté.

Lorsque l'option Réutiliser les contextes temporaires est cochée, en mode distant, 4D maintient les process web spécifiques et les réutilise pour les demandes suivantes. Supprimer l'étape de création du process améliore les performances du serveur web.

En contrepartie, vous devez veiller à initialiser systématiquement les variables utilisées dans les méthodes 4D afin d'éviter l'obtention des résultats incorrects. De même, il est nécessaire d'effacer toutes les sélections en cours ou les enregistrements définis lors de la demande précédente.

Cette option n'a d'effet qu'avec un serveur web 4D en mode distant. Avec un 4D en mode local, tous les process Web (autres que les process de session) sont arrêtés après leur utilisation.

Robots.txt

Certains robots (moteurs de recherche, crawlers...) parcourent les serveurs web et les pages statiques. Si vous ne voulez pas que les robots puissent accéder à l'ensemble de votre site, vous pouvez définir les URL auxquelles ils ne sont pas autorisés à accéder.

Pour ce faire, placez le fichier ROBOTS.TXT à la racine du serveur. Ce fichier doit être structuré comme suit :

```
User-Agent: <name>
Disallow: <URL> or <beginning of the URL>
```

Par exemple :

```
User-Agent: *
Disallow: /4D
Disallow: /%23%23
Disallow: /GIFS/
```

- "User-Agent: *" - tous les robots sont concernés.
- "Disallow: /4D" - Les robots ne sont pas autorisés à accéder aux URLs commençant par /4D.
- "Disallow : /%23%23" - les robots ne sont pas autorisés à accéder aux URL commençant par /%23%23.
- "Disallow : /GIFS/" - les robots ne sont pas autorisés à accéder au dossier /GIFS/ ou à ses sous-dossiers.

Autre exemple :

```
User-Agent: *
Disallow: /
```

Dans ce cas, les robots n'auront accès à aucune partie du site.

Dossier racine

Peut être configuré via	Nom	Commentaires
objet webServer	<code>rootFolder</code>	La propriété Text, mais il peut s'agir d'un objet 4D.Folder lorsqu'il est utilisé avec le paramètre <i>settings</i> de la fonction <code>start()</code>
<code>WEB SET ROOT FOLDER</code>		
Fenêtre de configuration	Configuration > Racine HTML par défaut	

Chemin du dossier racine du serveur web, i.e le dossier dans lequel 4D va chercher les pages HTML statiques et semi-dynamiques, les images, etc. à envoyer aux navigateurs. Le chemin d'accès est au format POSIX (chemin entier). Le serveur web doit être redémarré pour que le nouveau dossier racine soit pris en compte.

De plus, le dossier racine HTML définit, sur le disque dur du serveur web, le niveau hiérarchique au-dessus duquel les fichiers ne seront pas accessibles. Si une URL demandée ou une commande 4D tente d'accéder à un fichier situé au-dessus du dossier racine HTML, une erreur est renvoyée indiquant que le fichier n'a pas été trouvé.

Par défaut, 4D définit un dossier racine HTML nommé `WebFolder`. S'il n'existe pas encore, le dossier racine HTML est physiquement créé sur le disque au moment où le serveur Web est lancé pour la première fois. Le dossier racine est créé :

- avec 4D en mode local ou 4D Server, à côté du [dossier du projet](#).
- avec 4D en mode remote, dans le fichier des ressources locales.

Vous pouvez désigner un autre dossier HTML racine comme page d'accueil par défaut en entrant son chemin d'accès.

- Le chemin est relatif au [dossier du projet](#) (4D local et 4D Server) ou au dossier contenant l'application 4D ou le package logiciel (4D en mode distant).
- Le chemin est exprimé avec la syntaxe POSIX (les dossiers sont séparés par un slash (/)),
- Pour "remonter" d'un niveau dans la hiérarchie des dossiers, saisissez ".." (deux points) avant le nom de dossier
- Le chemin ne doit pas commencer par un slash (sauf si vous souhaitez que le dossier racine HTML soit le dossier distant du projet ou de 4D, mais pour interdire l'accès aux dossiers ci-dessus, auquel cas vous pouvez passer "/" comme dossier racine).

Par exemple, si vous voulez que le dossier racine HTML soit le sous-dossier "Web" du dossier "MyWebApp", entrez "MyWebApp/Web".

Lorsque le dossier racine HTML est modifié, le cache est effacé afin que les fichiers dont l'accès est restreint ne soient pas stockés.

Scalable Sessions

Peut être configuré via	Nom	Commentaires
objet webServer	<code>scalableSession</code>	
<code>WEB SET OPTION</code>	<code>Web scalable session</code>	
Fenêtre de configuration	Options (I) page/Scalable sessions (multi-process sessions)	

Scalable session management enabling status for the 4D web server. Web server sessions are detailed in the [User sessions](#) page.

Session Cookie Domain

Peut être configuré via	Nom	Commentaires
objet webServer	<code>sessionCookieDomain</code>	
WEB SET OPTION	Web session cookie domain	

Champ "path" du cookie de session. Utilisé pour contrôler la portée des cookies de session. Par exemple, si vous définissez la valeur "/4DACTION" pour ce sélecteur, le client enverra un cookie uniquement pour les requêtes dynamiques commençant par 4DACTION, et non pour les images, les pages statiques, etc.

Nom du cookie de session

Peut être configuré via	Nom	Commentaires
objet webServer	<code>sessionCookieName</code>	
WEB SET OPTION	Web session cookie name	

Nom du cookie utilisé pour stocker l'ID de session. Par défaut = "4DSID".

Chemin du cookie de session

Peut être configuré via	Nom	Commentaires
objet webServer	<code>sessionCookiePath</code>	
WEB SET OPTION	Web session cookie path	

Valeur du champ "domaine" du cookie de session. Utilisé pour contrôler la portée des cookies de session. Par exemple, si vous définissez la valeur "/4DACTION" pour ce sélecteur, le client enverra un cookie uniquement pour les requêtes dynamiques commençant par 4DACTION, et non pour les images, les pages statiques, etc.

Session Cookie SameSite

Peut être configuré via	Nom	Commentaires
objet webServer	<code>sessionCookieSameSite</code>	

Valeur de l'attribut `SameSite` du cookie de session. Cet attribut vous permet de déclarer si votre cookie doit être limité à un contexte de première partie ou de même site, comme une protection contre certaines attaques CSRF ([cross-site request forgery](#)).

Pour une description détaillée de l'attribut `SameSite`, veuillez vous reporter à la [documentation de Mozilla](#) ou à [cette page de développement web](#).

Trois valeurs sont disponibles :

- "Strict" (valeur par défaut de l'attribut `SameSite` pour les cookies de session 4D) : les cookies ne seront envoyés que dans le contexte de première partie, c'est-à-dire le contexte correspondant au domaine du site, et jamais à des sites Web tiers.
- "Lax": Cookies are not sent on cross-site subrequests (for example to load images or frames into a third-party site), but are sent when a user is navigating to the origin site (i.e. they follow a link).
- "None": Cookies are sent in all contexts, i.e in responses to both first-party and cross-origin requests. When "None" value is used, the cookie `Secure` attribute must also be set (or the cookie will be blocked).

La valeur de l'attribut `Secure` du cookie de session est automatiquement définie sur "True" si la connexion est HTTPS (quelle que soit la valeur de l'attribut `SameSite`).

Il n'est pas recommandé de définir `SameSite=None` sur un serveur HTTP car l'attribut `Secure` sera absent

(utilisé uniquement en HTTPS) et les cookies seront bloqués.

Utiliser des process préemptifs

Peut être configuré via	Nom	Commentaires
Fenêtre de configuration	Options (I) > Process Web simultanés maxi	

This option enables the preemptive mode for your application's web server code when No sessions option is selected (the preemptive mode is always enabled with scalable sessions). When this option is checked in this context, the 4D compiler will automatically evaluate the thread-safety property of each piece of [web-related code](#) and return errors in case of incompatibility.

Propriétés obsolètes

Les paramètres suivants sont toujours pris en charge mais reposent sur des fonctionnalités ou des technologies obsolètes. Il est généralement recommandé de conserver les valeurs par défaut.

Autoriser l'accès aux bases de données par le biais des URL 4DSYNC

Cette option contrôle le support des requêtes de synchronisation HTTP contenant des URLs dépréciées `/4DSYNC`.

Validation de l'adresse IP de la session

Cette option n'est pas disponible en [mode sessions évolutives](#) (il n'y a pas de validation).

Statut de validation d'adresse IP pour les cookies de session. Pour des raisons de sécurité, le serveur Web vérifie par défaut l'adresse IP de chaque requête contenant un cookie de session et la rejette si cette adresse ne correspond pas à l'adresse IP utilisée pour créer le cookie. Dans certaines applications spécifiques, vous souhaiterez peut-être désactiver cette validation et accepter les cookies de session, même lorsque leurs adresses IP ne correspondent pas. Par exemple, lorsque les appareils mobiles basculent entre les réseaux Wifi et 4G/5G, leur adresse IP change. Dans ce cas, vous devez passer 0 à cette option pour permettre aux clients de continuer à utiliser leurs sessions Web même lorsque les adresses IP changent. Note : ce paramètre réduit le niveau de sécurité de votre application. Une fois modifiée, cette option prend effet immédiatement (il n'est pas nécessaire de redémarrer le serveur HTTP).

Envoyer directement les caractères étendus

Lorsque cette option est cochée, le serveur web envoie les caractères étendus tels quels dans les pages semi-dynamiques, sans les convertir en entités HTML. Cette option a augmenté la vitesse d'exécution sur la plupart des systèmes d'exploitation étrangers (en particulier le système japonais).

Utiliser les connexions persistantes

Le serveur Web 4D peut utiliser des connexions persistantes. Cette option permet de maintenir une seule connexion TCP ouverte pour l'ensemble des échanges entre le navigateur web et le serveur afin d'économiser les ressources du système et d'optimiser les transferts.

L'option Utiliser les connexions persistantes active ou désactive les connexions TCP persistantes pour le serveur web. Cette option est activée par défaut. Dans la plupart des cas, il est conseillé de garder cette option cochée car elle accélère les échanges. Si le navigateur Web ne prend pas en charge le maintien de la connexion, le serveur Web 4D bascule automatiquement à l'HTTP/1.0.

La fonction de connexions persistantes du serveur Web 4D s'applique à toutes les connexions TCP/IP (HTTP, HTTPS). Notez toutefois que les connexions persistantes ne sont pas toujours utilisées pour tous les process web 4D.

Dans certains cas, d'autres fonctions internes optimisées peuvent être appelées. Les connexions persistantes sont principalement utiles pour les pages statiques.

Deux options permettent de définir le mode de fonctionnement des connexions persistantes :

- Nombre de demandes par connexion : Permet de définir le nombre maximal de requêtes et de réponses capables d'être transmises sur une connexion persistante. Limiter le nombre de demandes par connexion permet d'éviter le server flooding, provoqué par un trop grand nombre de requêtes entrantes (technique utilisée par les pirates informatiques).

La valeur par défaut (100) peut être augmentée ou diminuée en fonction des ressources de la machine hébergeant le serveur 4D Web.

- Délai avant déconnexion : Cette valeur définit l'attente maximale (en secondes) pour le maintien d'une connexion TCP sans réception d'une requête de la part du navigateur web. Une fois ce délai écoulé, le serveur ferme la connexion.

Si le navigateur Web envoie une requête après la fermeture de la connexion, une nouvelle connexion TCP est automatiquement créée. Cette opération est invisible pour l'utilisateur.

Administration

4D offre plusieurs outils intégrés permettant de démarrer, arrêter ou administrer le serveur web intégré.

Démarrer le Serveur Web 4D

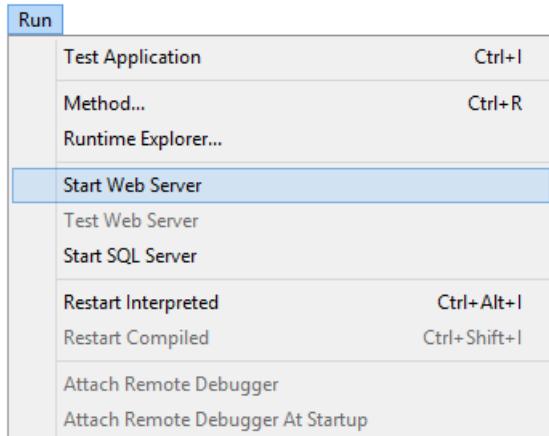
Une licence 4D Web Application est requise pour pouvoir lancer le serveur web de 4D ou 4D Server. Pour plus d'informations, consultez le [site web de 4D](#).

Un projet 4D peut démarrer et surveiller un serveur Web pour l'application principale (hôte) ainsi que chaque composant hébergé.

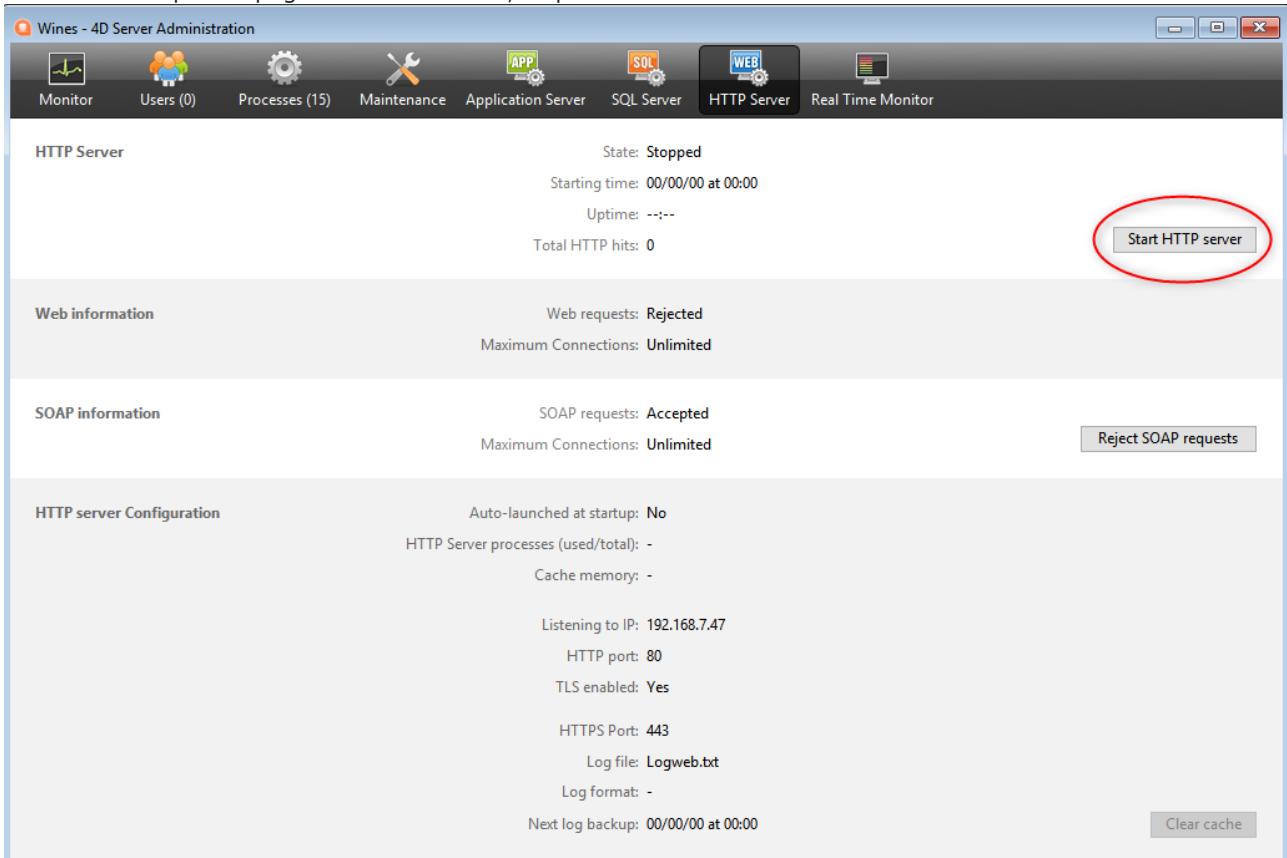
Il y a plusieurs manières de démarrer le serveur Web principal :

- via un bouton/une commande de menu :

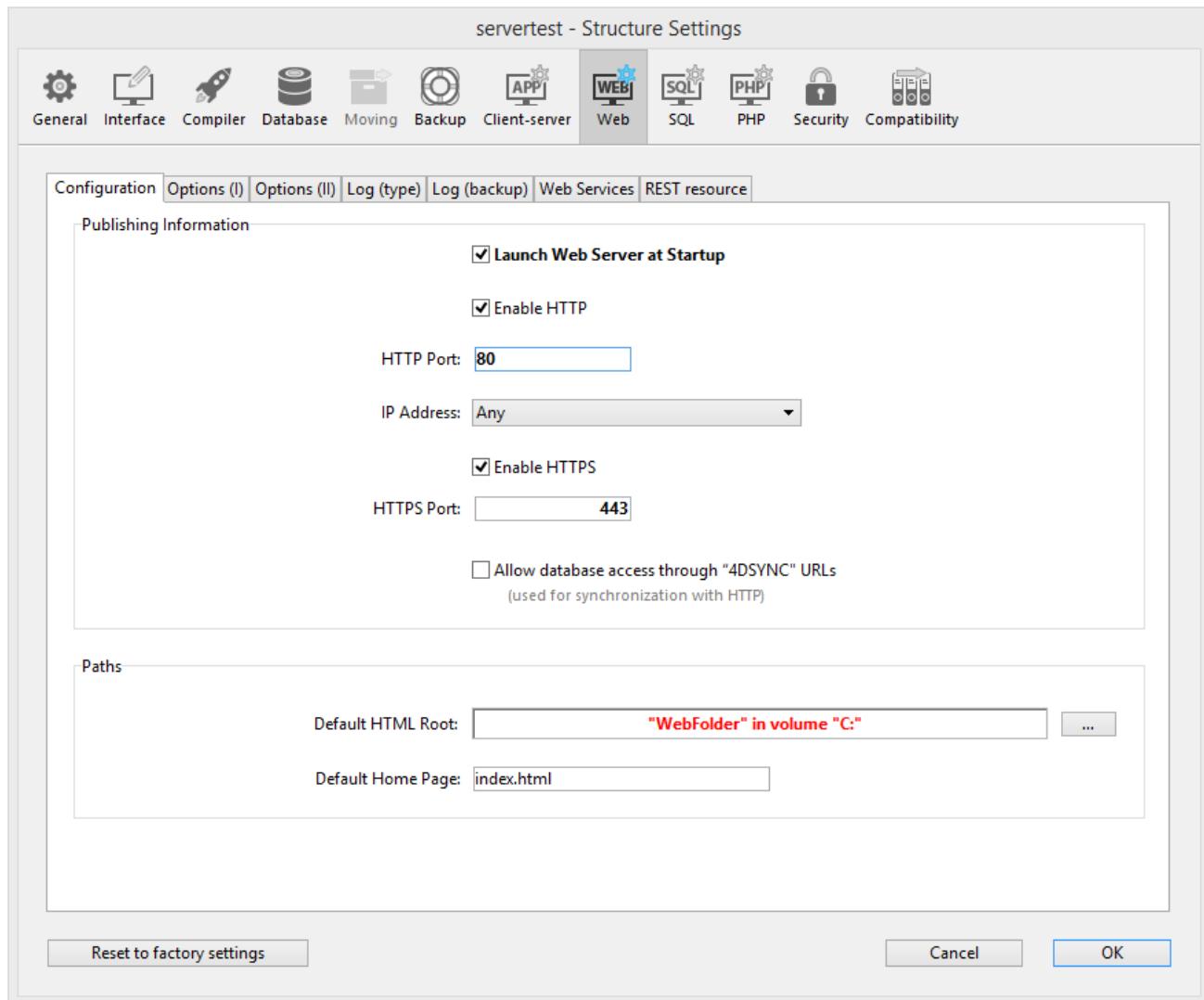
- 4D: Exécution > démarrer le serveur Web



- 4D Server : depuis la page du serveur HTTP, cliquez sur Démarrer le serveur HTTP



- automatiquement à l'ouverture de l'application 4D, en cochant l'option Lancer le serveur Web au démarrage dans Propriétés > Web > Configuration



- Par programmation, en appelant la fonction `webServer.start()` ou la commande `WEB START SERVER`.

Le serveur Web de n'importe quel composant peut être lancé en appelant la fonction `webServer.start()` sur l'objet serveur Web du composant.

Il n'est pas nécessaire de relancer l'application 4D pour démarrer ou arrêter le serveur Web.

Arrêter le Serveur Web 4D

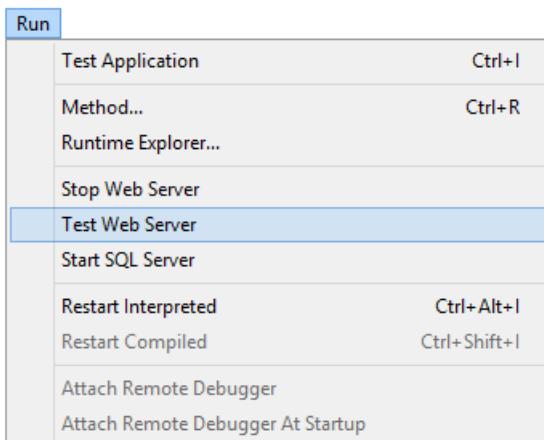
Il y a plusieurs manières d'arrêter le serveur Web principal :

- Via le menu 4D, Exécution > Arrêter le serveur Web, ou via le bouton Arrêter le serveur HTTP de 4D Server (les deux items affichent Démarrer... quand le serveur n'est pas encore démarré).
- Par programmation, en appelant la fonction `webServer.stop()` ou la commande `WEB STOP SERVER`.

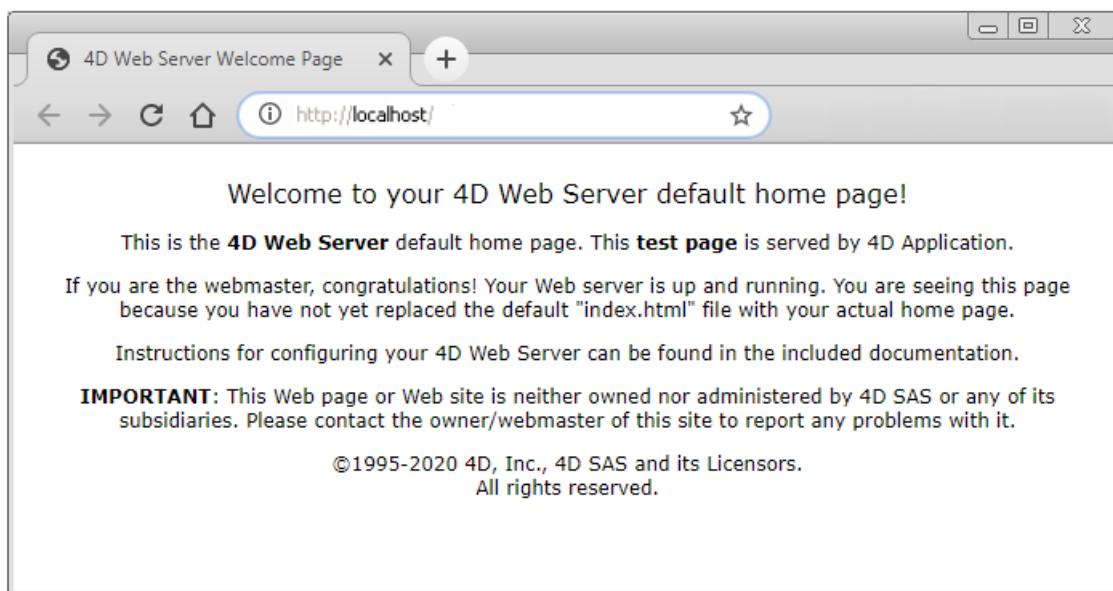
Le serveur Web de n'importe quel composant peut être arrêté en appelant la fonction `webServer.stop()` sur l'objet serveur Web du composant.

Tester le Serveur Web 4D

La commande Tester le serveur Web peut être utilisé pour s'assurer que le serveur web intégré fonctionne correctement (4D uniquement). Cette commande est accessible depuis le menu Run lorsque le serveur web est lancé :



Lorsque vous sélectionnez cette commande, la page d'accueil du site Web publié par l'application 4D s'affiche dans une fenêtre de votre navigateur par défaut :



Cette commande vous permet de vérifier que le serveur web, l'affichage de la page d'accueil, etc. fonctionnent correctement. La page est appelée en utilisant l'URL *localhost*, qui est le raccourci standard désignant l'adresse IP de la machine sur laquelle le navigateur web est exécuté. Cette commande prend en compte le numéro du [port de publication TCP](#) spécifié dans les paramètres.

Vider le cache

À tout moment, vous pouvez vider le cache des pages et des images qu'il contient (si, par exemple, vous avez modifié une page statique et que vous souhaitez la recharger dans le cache).

Pour ce faire, vous pouvez :

- 4D : cliquez sur le bouton Vider le cache dans la page Web/Options (I) de la boîte de dialogue des Paramètres.
- 4D Server : cliquez sur le bouton Vider le cache dans la page HTTP de la fenêtre [Administration de 4D Server](#).

Le cache est alors immédiatement effacé.

Vous pouvez aussi utiliser l'url </4DCACHECLEAR>.

Explorateur d'exécution

La page Watch (rubrique Web) de l'Explorateur d'exécution affiche les informations du serveur Web, notamment :

- Occupation du cache Web : indique le nombre de pages présentes dans le cache web ainsi que le pourcentage d'utilisation. Cette information n'est disponible que si le serveur web est actif et que la taille du cache est supérieure

à 0.

- Temps d'activité du serveur Web : indique la durée d'utilisation (au format heures:minutes:secondes) du serveur Web. Ces informations ne sont disponibles que si le serveur web est actif.
- Nombre de requêtes http : indique le nombre total de requêtes HTTP reçues depuis le démarrage du serveur web, ainsi qu'un nombre instantané de requêtes par seconde (mesure prise entre deux mises à jour de l'Explorateur d'exécution). Ces informations ne sont disponibles que si le serveur web est actif.

URLs d'administration

Les URL d'administration Web vous permettent de contrôler le site web publié sur votre serveur. 4D Web Server accepte quatre URLs spécifiques : `/4DSTATS`, `/4DHTMLSTATS`, `/4DCACHECLEAR` et `/4WEBTEST`.

`/4DSTATS`, `/4DHTMLSTATS` et `/4DCACHECLEAR` ne sont disponibles que pour le concepteur et l'administrateur de la base de données. Si le système de mot de passe 4D n'a pas été activé, ces URL sont disponibles pour tous les utilisateurs. `/4WEBTEST` est toujours disponible.

`/4DSTATS`

L'URL `/4DSTATS` renvoie plusieurs éléments d'information dans un tableau HTML (affichable dans un navigateur) :

Élément	Description
Cache Current Size	Taille actuelle du cache du serveur web (en octets)
Cache Max Size	Taille maximale du cache (en octets)
Cached Object Max Size	Taille maximale de chaque objet dans le cache (en octets)
Cache Use	Pourcentage de cache utilisé
Cached Objects	Nombre d'objects trouvés dans le cache, images incluses

Ces informations peuvent vous permettre de vérifier le fonctionnement de votre serveur et éventuellement d'adapter les paramètres correspondants.

La commande `WEB LIRE STATISTIQUES` permet également d'obtenir des informations sur la façon dont le cache est utilisé pour les pages statiques.

`/4DHTMLSTATS`

L'URL `/4DHTMLSTATS` retourne les mêmes informations que l'URL `/4DSTATS`, également sous forme de tableau. La différence est que le champ Cached objects ne compte que les pages HTML (les fichiers d'images ne sont pas pris en compte). De plus, cette URL retourne le champ Filtered Objects.

Élément	Description
Cache Current Size	Taille actuelle du cache du serveur web (en octets)
Cache Max Size	Taille maximale du cache (en octets)
Cached Object Max Size	Taille maximale de chaque objet dans le cache (en octets)
Cache Use	Pourcentage de cache utilisé
Cached Objects	Nombre d'objects trouvés dans le cache, sans les images
Filtered Objects	Nombre d'objets dans le cache non pris en compte par l'URL, notamment les images

`/4DCACHECLEAR`

L'URL `/4DCACHECLEAR` retire immédiatement les pages statiques et les images du cache. Il vous permet donc de

"forcer" la mise à jour des pages qui ont été modifiées.

/4DWEBTEST

L'URL `/4DWEBTEST` permet de vérifier l'état du serveur web. Lorsque cette URL est appelée, 4D renvoie un fichier texte avec les champs HTTP suivants remplis :

Champ HTTP	Description	Exemple
Date	date courante au format RFC 822 format	Mon, 7 Dec 2020 13:12:50 GMT
Server	4D/numéro de version	4D/18.5.0 (Build 18R5.257368)
User-Agent	nom et version @ adresse IP du client	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.183 Safari/537.36 @ 127.0.0.1

Logs

4D vous permet de générer deux historiques de requêtes Web :

- un historique de débogage, utile dans la phase de développement du serveur web (`HTTPDebugLog.txt`),
- un journal standardisé des requêtes web, plutôt utilisé à des fins statistiques (`logweb.txt`).

Les deux fichiers journaux sont automatiquement créés dans le dossier Logs du projet d'application.

HTTPDebugLog.txt

Le [fichier de débogage](#) peut être activé via l'objet `web server` ou la commande `WEB FIXER OPTION`

Ce fichier d'historique enregistre chaque requête HTTP et chaque réponse en mode brut (raw). Les requêtes sont enregistrées dans leur totalité (en-têtes compris). Les parts du body peuvent optionnellement être enregistrées.

Les champs suivants sont enregistrés pour chaque requête et réponse :

Nom des champs	Description
SocketID	ID du socket utilisé pour la communication
PeerIP	Adresse IPv4 de l'hôte (client)
PeerPort	Port utilisé par l'hôte (client)
TimeStamp	Horodatage en millisecondes (depuis le démarrage du système)
ConnectionID	Connexion UUID (UUID du VTCPSocket utilisé pour la communication)
SequenceNumber	Numéro d'opération séquentiel et unique dans la session d'historique

logweb.txt

Le [fichier d'enregistrements d'historique Web](#) peut être activé via l'objet `web server`, la commande `WEB FIXER OPTION` ou la page Web > Log des Propriétés. Vous devez sélectionner un format d'historique.

CLF/DLF

Chaque ligne du fichier représente une requête, comme : `host rfc931 user [DD/MMM/YYYY:HH:MM:SS] "request" state length` Chaque champ est séparé par un espace et chaque ligne se termine par la séquence CR/LF (caractère 13, caractère 10).

Le format DLF (Distilled Log Format) est similaire au format CLF (Common Log format) et utilise exactement la même structure. Il ajoute simplement deux champs HTTP supplémentaires à la fin de chaque requête : Referer et User-agent. Voici la description des formats CLF/DLF (non personnalisables) :

Nom des champs	Description
host	Adresse IP du client (ex. 192.100.100.10)
rfc931	information non générée par 4D, c'est toujours - (le signe moins)
user	le nom d'utilisateur tel qu'il est authentifié, ou bien - (le signe moins). Si le nom de l'utilisateur contient des espaces, ils sont remplacés par des _ (tiret bas).
DD/MMM/YYYY:HH:MM:SS	DD : jour, MMM : abréviation de 3 lettres pour le nom du mois (Jan, Feb,...), YYYY : année, HH : heure, MM : minutes, SS : secondes. La date et heure sont locales au serveur.
request	requête envoyée par le client (ex. GET /index.htm HTTP/1.0)
state	réponse donnée par le serveur
length	taille des données retournées (HTTP header exclu) ou 0
Referer	DLF uniquement. Contient l'URL de la page qui pointe vers le document de la requête.
User-agent	DLF uniquement - Contient le nom et la version du navigateur ou du logiciel du client à l'origine de la demande

ELW/WLF

Le format ELF (Extended Log Format) est très répandu dans le monde des navigateurs HTTP. Il peut être utilisé pour construire des historiques sophistiqués qui répondent à des besoins spécifiques. Pour cette raison, le format ELF peut être personnalisé : il est possible de choisir les champs à enregistrer ainsi que leur ordre d'insertion dans le fichier.

Le format WLF (WebStar Log format) a été développé spécifiquement pour le serveur 4D WebSTAR.

Configurer les champs

Lorsque vous choisissez le format ELF ou WLF, la zone "Web Log Token Selection" affiche les champs disponibles pour le format choisi. Vous devrez sélectionner chaque champ à inclure dans le journal. Pour ce faire, cochez les champs souhaités.

Le même champ ne peut pas être sélectionné deux fois.

Le tableau suivant répertorie les champs disponibles pour chaque format (par ordre alphabétique) et décrit leur contenu :

Champ	ELF	WLF	Valeur
BYTES_RECEIVED		X	Nombre d'octets reçus par le serveur
BYTES_SENT	X	X	Nombre d'octets envoyés par le serveur au client
C_DNS	X	X	Adresse IP du DNS (ELF : champ identique au champ C_IP)
C_IP	X	X	Adresse IP du client (par exemple 192.100.100.10)
CONNECTION_ID		X	Numéro unique de la connexion
CS(COOKIE)	X	X	Informations sur les cookies contenus dans la requête HTTP
CS(HOST)	X	X	Champ Host de la requête HTTP
CS(REFERER)	X	X	URL de la page pointant vers le document demandé
CS(USER_AGENT)	X	X	Informations sur le logiciel et le système d'exploitation du client
CS_SIP	X	X	Adresse IP du serveur
CS_URI	X	X	URI sur lequel la requête est effectuée
CS_URI_QUERY	X	X	Paramètres d'interrogation de la requête
CS_URI_STEM	X	X	Partie de la requête sans les paramètres d'interrogation
DATE	X	X	DD: jour, MMM: abréviation de 3 lettres pour le mois (Jan, Feb,...), YYYY: année
METHOD	X	X	Méthode HTTP utilisée pour la requête adressée au serveur
PATH_ARGS		X	CGI parameters: string located after the "\$" character
STATUS	X	X	Réponse fournie par le serveur
TIME	X	X	HH: heure, MM: minutes, SS: secondes
TRANSFER_TIME	X	X	Délai ayant été nécessaire au serveur pour générer la réponse
USER	X	X	Nom d'utilisateur s'il s'est authentifié, sinon - (signe moins). Si le nom d'utilisateur contient des espaces, ils sont remplacés par des _ (traits de soulignement)
Variable URL		X	URL demandé par le client

Les dates et heures sont données au format GMT

Fréquence de backup

Comme la taille d'un *logweb.txt* fichier évoluer considérablement, il est possible de mettre en place un mécanisme d'archivage automatique. Le déclenchement d'une backup peut être basé sur une certaine période de temps (exprimée en heures, jours, semaine ou mois), ou sur la taille du fichier ; lorsque le délai fixé (ou la taille du fichier) est atteinte, 4D ferme et archive automatiquement le fichier d'historique en cours et en crée un nouveau.

Lorsque la sauvegarde du fichier d'historique web est déclenchée, le fichier d'historique est archivé dans un dossier nommé "Archives Logweb", qui est créé au même niveau que le fichier *logweb.txt*.

Le fichier archivé est renommé sur le modèle suivant : "YYYY_MM_DD_Thh_mm_ss.txt". Par exemple, pour un dossier archivé le 4 septembre 2020 à 15 h 50 et 7 secondes : "D2020_09_04_T15_50_07.txt." et 7 secondes : "D2020_09_04_T15_50_07.txt."

Paramètres de backup

Les paramètres de sauvegarde automatique du *logweb.txt* sont définis sur la page Web > Journal (périodicité) des Paramètres :

webServer - Structure Settings

Backup Frequency for Web Log File

- No Backup
- Every hour(s) starting at
- Every day(s) at
- Every week(s)
 - Monday at
 - Tuesday at
 - Wednesday at
 - Thursday at
 - Friday at
 - Saturday at
 - Sunday at
- Every month(s) Day at
- Every

[Reset to factory settings](#) [Cancel](#) [OK](#)

Vous devez d'abord choisir la fréquence (jours, semaines, etc.) ou le critère de la taille limite du fichier en cliquant sur le bouton radio correspondant. Vous devez ensuite spécifier le moment précis du backup si nécessaire.

- Pas de sauvegarde du journal : La fonction de sauvegarde programmée est désactivée.
- Toutes les X heure(s) : Cette option est utilisée pour programmer des sauvegardes sur une base horaire. Vous pouvez entrer une valeur entre 1 et 24.
 - à partir de: Permet de définir l'heure du déclenchement du premier backup.
- Tous les N jour(s) à N : permet de programmer des backups sur une base journalière. Saisissez 1 si vous souhaitez une sauvegarde hebdomadaire. Lorsque vous cochez cette option, vous devez indiquer l'heure à laquelle la sauvegarde doit être déclenchée.
- Tous les N jour(s) à N : permet de programmer des backups sur une base hebdomadaire. Saisissez 1 si vous souhaitez une sauvegarde hebdomadaire. Lorsque vous cochez cette option, vous devez indiquer le ou les jours de la semaine et l'heure à laquelle chaque sauvegarde doit être déclenchée. Vous pouvez cocher un ou plusieurs jour(s) de la semaine. Par exemple, vous pouvez utiliser cette option pour définir deux sauvegardes hebdomadaires : une le mercredi et une le vendredi.
- Tous les N mois, Ne jour à N : permet de programmer des sauvegardes sur une base mensuelle. Saisissez 1 si vous souhaitez une sauvegarde mensuelle. Lorsque vous cochez cette option, vous devez indiquer le jour de chaque mois auquel la sauvegarde doit être déclenchée, ainsi que l'heure de déclenchement.
- Tous les N Mo : Cette option est utilisée pour programmer les sauvegardes en fonction de la taille du fichier journal courant. Un backup se déclenche automatiquement quand le fichier atteint la taille spécifiée. La taille limite du fichier peut être fixée à 1, 10, 100 ou 1000 Mo.

Objet Serveur Web

Un projet 4D peut démarrer et surveiller un serveur Web pour l'application principale (hôte) ainsi que chaque composant hébergé.

Par exemple, si vous avez installé deux composants dans votre application principale, vous pouvez démarrer et contrôler jusqu'à trois serveurs Web indépendants à partir de votre application :

- un serveur web pour l'application hôte,
- un serveur web pour le composant n°1,
- un serveur web pour le composant n°2.

En dehors de la mémoire, il n'y a pas de limite au nombre de composants et donc, de serveurs Web, pouvant être rattachés à un seul projet d'application 4D.

Chaque serveur web 4D, y compris le serveur web de l'application principale, est exposé comme un objet spécifique de la classe `4D.WebServer`. Une fois instancié, un objet serveur Web peut être géré depuis l'application courante ou depuis n'importe quel composant à l'aide d'un [grand nombre de propriétés et de fonctions](#).

Les [commandes WEB](#) héritées du langage 4D sont prises en charge mais ne peuvent pas sélectionner le serveur Web auquel elles s'appliquent (voir ci-dessous).

Chaque serveur web (application hôte ou composant) peut être utilisé dans son propre contexte, notamment :

- les appels vers la méthode base `On Web Authentication` et `On Web Connection`,
- le traitement des balises 4D et les appels de méthodes,
- sessions web et gestion du protocole TLS.

Cela vous permet de développer des composants indépendants et des fonctionnalités qui accompagnent leurs propres interfaces Web.

Instancier un objet serveur web

L'objet serveur Web de l'application hôte (serveur Web par défaut) est automatiquement chargé par 4D au démarrage. Ainsi, si vous écrivez dans un projet nouvellement créé :

```
$nbSrv:=WEB Server list.length  
//la valeur de $nbSrv est 1
```

Pour instancier un objet serveur web, appelez la commande `WEB Server` :

```
//créer une variable objet de la classe 4D.WebServer  
var webServer : 4D.WebServer  
    //appeler le serveur Web depuis le contexte courant  
webServer:=WEB Server  
  
    //équivalent à  
webServer:=WEB Server(Web server database)
```

Si l'application utilise des composants et que vous souhaitez appeler :

- le serveur Web de l'application hôte à partir d'un composant ou
- le serveur qui a reçu la requête (quel que soit le serveur)

vous pouvez également utiliser :

```

var webServer : 4D.WebServer
    //appeler le serveur web hôte depuis un composant
webServer:=WEB Server(Web server host database)
    //appeler le serveur web cible
webServer:=WEB Server(Web server receiving request)

```

Fonctions du serveur web

Un [objet de classe Web server](#) contient les fonctions suivantes :

Fonctions	Paramètres	Valeur retournée	Description
start()	settings (objet)	status (object)	Démarre le serveur web
stop()	-	-	Stoppe le serveur web

Pour démarrer et arrêter un serveur Web, il suffit d'appeler les fonctions [start\(\)](#) et [stop\(\)](#) de l'objet serveur Web :

```

var $status : Object
    //pour démarrer un serveur web avec les paramètres par défaut
$status:=webServer.start()
    //pour démarrer un serveur web avec des paramètres personnalisés
    //objet $settings contenant des propriétés du serveur web
webServer.start($settings)

    //pour stopper le serveur web
$status:=webServer.stop()

```

Propriétés du serveur web

Un objet serveur Web contient [diverses propriétés](#) qui configurent le serveur Web.

Ces propriétés sont définies :

- à l'aide du paramètre `<0>settings</0>` de la fonction [.start\(\)](#) (sauf pour les propriétés en lecture seule, voir ci-dessous),
 - si elles ne sont pas utilisées, à l'aide de la commande [WEB SET OPTION](#) (applications hôtes uniquement),
 - si elles ne sont pas utilisées, dans les paramètres de l'application hôte ou du composant.
- Si le serveur Web n'est pas démarré, les propriétés contiennent les valeurs qui seront utilisées au prochain démarrage du serveur Web.
 - Si le serveur Web est démarré, les propriétés contiennent les valeurs réelles utilisées par le serveur Web (les paramètres par défaut peuvent avoir été remplacés par le paramètre `settings` de la fonction [.start\(\)](#)).

isRunning, name, openSSLVersion et perfectForwardSecrecy sont des propriétés en lecture seule qui ne peuvent pas être prédéfinies dans le paramètre objet `settings` pour la fonction [start\(\)](#).

Portée des commandes 4D Web

Le langage 4D contient [plusieurs commandes](#) permettant de contrôler le serveur Web. Cependant, ces commandes sont destinées à fonctionner avec un seul serveur Web (par défaut). Lorsque vous utilisez ces commandes dans le contexte d'objets serveur Web, assurez-vous que leur portée est appropriée.

Commande	Portée
SET DATABASE PARAMETER	Application hôte du serveur web
WEB CLOSE SESSION	Serveur Web ayant reçu la requête
WEB GET BODY PART	Serveur Web ayant reçu la requête
WEB Get body part count	Serveur Web ayant reçu la requête
WEB Get Current Session ID	Serveur Web ayant reçu la requête
WEB GET HTTP BODY	Serveur Web ayant reçu la requête
WEB GET HTTP HEADER	Serveur Web ayant reçu la requête
WEB GET OPTION	Application hôte du serveur web
WEB Get server info	Application hôte du serveur web
WEB GET SESSION EXPIRATION	Serveur Web ayant reçu la requête
WEB Get session process count	Serveur Web ayant reçu la requête
WEB GET STATISTICS	Application hôte du serveur web
WEB GET VARIABLES	Serveur Web ayant reçu la requête
WEB Is secured connection	Serveur Web ayant reçu la requête
WEB Is server running	Application hôte du serveur web
WEB SEND BLOB	Serveur Web ayant reçu la requête
WEB SEND FILE	Serveur Web ayant reçu la requête
WEB SEND HTTP REDIRECT	Serveur Web ayant reçu la requête
WEB SEND RAW DATA	Serveur Web ayant reçu la requête
WEB SEND TEXT	Serveur Web ayant reçu la requête
WEB SET HOME PAGE	Application hôte du serveur web
WEB SET HTTP HEADER	Serveur Web ayant reçu la requête
WEB SET OPTION	Application hôte du serveur web
WEB SET ROOT FOLDER	Application hôte du serveur web
WEB START SERVER	Application hôte du serveur web
WEB STOP SERVER	Application hôte du serveur web
WEB Validate digest	Serveur Web ayant reçu la requête

Template pages

Le serveur Web de 4D vous permet d'utiliser des pages de modèles HTML contenant des balises, c'est-à-dire un mélange de code HTML statique et de références 4D ajoutées via des [balises de transformation](#) telles que 4DTEXT, 4DIF ou 4DINCLUDE. These tags are usually inserted as HTML type comments (`<!--#4DTagName TagValue-->`) into the HTML source code.

When these pages are sent by the HTTP server, they are parsed and the tags they contain are executed and replaced with the resulting data. The pages received by the browsers are thus a combination of static elements and values coming from 4D processing.

For example, if you write in an HTML page:

```
<P>Welcome to <!--#4DTEXT vtSiteName-->!</P>
```

The value of the 4D variable `vtSiteName` will be inserted in the HTML page.

Tags for templates

The following 4D tags are available:

- 4DTEXT, to insert 4D variables and expressions as text,
- 4DHTML, to insert HTML code,
- 4DEVAL, to evaluate any 4D expression,
- 4DSCRIPT, to execute a 4D method,
- 4DINCLUDE, to include a page within another one,
- 4DBASE, to modify the default folder used by the 4DINCLUDE tag,
- 4DCODE, to insert 4D code,
- 4DIF, 4ELSE, 4ELSEIF and 4ENDIF, to insert conditions in the HTML code,
- 4DLOOP et 4DENDLOOP, pour faire des boucles dans le code HTML,
- 4DEACH et 4DENDEACH, pour boucler des collections, des entity selections ou des propriétés d'objets.

Ces balises sont décrites dans la page [Balises de transformation](#).

It is possible to mix tags. For example, the following HTML code is allowed:

```

<HTML>
...
<BODY>
<!--#4DSCRIPT/PRE_PROCESS--> (Method call)
<!--#4DIF (myvar=1)--> (If condition)
    <!--#4DINCLUDE banner1.html--> (Subpage insertion)
<!--#4DENDIF--> (End if)
<!--#4DIF (myvar=2)-->

    <!--#4DINCLUDE banner2.html-->
<!--#4DENDIF-->

<!--#4DLOOP [TABLE]--> (loop on the current selection)
<!--#4DIF ([TABLE]ValNum>10)--> (If [TABLE]ValNum>10)
    <!--#4DINCLUDE subpage.html--> (subpage insertion)
<!--#4DELSE--> (Else)
    <B>Value: <!--#4DTEXT [TABLE]ValNum--></B><BR>
        (Field display)
<!--#4DENDIF-->
<!--#4DENDLOOP--> (End for)
</BODY>
</HTML>

```

Tag parsing

For optimization reasons, the parsing of the HTML source code is not carried out by the 4D Web server when HTML pages are called using simple URLs that are suffixed with ".HTML" or ".HTM".

Parsing of the contents of template pages sent by 4D web server takes place when `WEB SEND FILE` (.htm, .html, .shtm, .shtml), `WEB SEND BLOB` (text/html type BLOB) or `WEB SEND TEXT` commands are called, as well as when sending pages called using URLs. Dans ce dernier cas, à des fins d'optimisation, les pages suffixées par ".htm" et ".html" ne sont PAS parsées. In order to "force" the parsing of HTML pages in this case, you must add the suffix ".shtm" or ".shtml" (for example, <http://www.server.com/dir/page.shtm>). An example of the use of this type of page is given in the description of the `WEB GET STATISTICS` command. XML pages (.xml, .xsl) are also supported and always parsed by 4D.

You can also carry out parsing outside of the Web context when you use the `PROCESS 4D TAGS` command.

Internally, the parser works with UTF-16 strings, but the data to parse may have been encoded differently. When tags contain text (for example `4DHTML`), 4D converts the data when necessary depending on its origin and the information available (charset). Below are the cases where 4D parses the tags contained in the HTML pages, as well as any conversions carried out:

Action / Command	Content analysis of the sent pages	Support of \$ syntax(*)	Character set used for parsing tags
Pages called via URLs	X, except for pages with ".htm" or ".html" extensions	X, except for pages with ".htm" or ".html" extensions	Use of charset passed as parameter of the "Content-Type" header of the page. If there is none, search for a META-HTTP EQUIV tag with a charset. Otherwise, use of default character set for the HTTP server
WEB SEND FILE	X	-	Use of charset passed as parameter of the "Content-Type" header of the page. If there is none, search for a META-HTTP EQUIV tag with a charset. Otherwise, use of default character set for the HTTP server
WEB SEND TEXT	X	-	No conversion necessary
WEB SEND BLOB	X, if BLOB is of the "text/html" type	-	Use of charset set in the "Content-Type" header of the response. Otherwise, use of default character set for the HTTP server
Inclusion by the <!-- #4DINCLUDE--> tag	X	X	Use of charset passed as parameter of the "Content-Type" header of the page. If there is none, search for a META-HTTP EQUIV tag with a charset. Otherwise, use of default character set for the HTTP server
PROCESS 4D TAGS	X	X	Text data: no conversion. BLOB data: automatic conversion from the Mac-Roman character set for compatibility

(*) The alternative \$-based syntax is available for 4DHTML, 4DTEXT and 4DEVAL tags.

Accès aux méthodes 4D via le Web

L'exécution d'une méthode 4D avec 4DEACH, 4DELSEIF, 4DEVAL, 4DHTML, 4DIF, 4DL0OP, 4DSCRIPT, ou 4DTEXT à partir d'une requête web est soumise à la valeur de l'attribut [disponible via des balises 4D et des URL \(4DACTION...\)](#) définie dans les propriétés de la méthode. Si cet attribut n'est pas vérifié pour la méthode, celle-ci ne peut pas être appelée à partir d'une requête Web.

Prevention of malicious code insertion

4D tags accept different types of data as parameters: text, variables, methods, command names, etc. When this data is provided by your own code, there is no risk of malicious code insertion since you control the input. However, your database code often works with data that was, at one time or another, introduced through an external source (user input, import, etc.).

In this case, it is advisable to not use tags such as 4DEVAL or 4DSCRIPT, which evaluate parameters, directly with this sort of data.

De plus, selon le [principe de la récursivité](#), le code malveillant peut lui-même inclure des balises de transformation. In this case, it is imperative to use the 4DTEXT tag. Imagine, for example, a Web form field named "Name", where users must enter their name. This name is then displayed using a <!--#4DHTML vName--> tag in the page. If text of the "<!-- #4DEVAL QUIT 4D-->" type is inserted instead of the name, interpreting this tag will cause the application to be exited. To avoid this risk, you can just use the 4DTEXT tag systematically in this case. Since this tag escapes the special HTML characters, any malicious recursive code that may have been inserted will not be reinterpreted. Pour se référer à l'exemple précédent, le champ "Name" contiendra, dans ce cas, " <!--#4DEVAL QUIT 4D--> " qui ne sera pas transformé.

Traitements des requêtes HTTP

The 4D web server provides several features to handle HTTP requests:

- the `On Web Connection` database method, a router for your web application,
- the `/4DACTION` URL to call server-side code
- `WEB GET VARIABLES` to get values from HTML objects sent to the server
- other commands such as `WEB GET HTTP BODY`, `WEB GET HTTP HEADER`, or `WEB GET BODY PART` allow to customize the request processing, including cookies.
- the `COMPILER_WEB` project method, to declare your variables.

On Web Connection

The `On Web Connection` database method can be used as the entry point for the 4D Web server.

Database method calls

The `On Web Connection` database method is automatically called when the server receives any URL that is not a path to an existing page on the server. The database method is called with the URL.

For example, the URL "`a/b/c`" will call the database method, but "`a/b/c.html`" will not call the database method if the page "`c.html`" exists in the "`a/b`" subfolder of the [WebFolder](#).

The request should have previously been accepted by the `On Web Authentication` database method (if it exists) and the web server must be launched.

Syntaxe

`On Web Connection($1 : Text ; $2 : Text ; $3 : Text ; $4 : Text ; $5 : Text ; $6 : Text)`

Paramètres	Type		Description
\$1	Text	<-	Variable URL
\$2	Text	<-	HTTP headers + HTTP body (up to 32 kb limit)
\$3	Text	<-	IP address of the web client (browser)
\$4	Text	<-	Adresse IP du serveur
\$5	Text	<-	Nom d'utilisateur
\$6	Text	<-	Password

You must declare these parameters as shown below:

```
//On Web Connection database method  
  
C_TEXT($1;$2;$3;$4;$5;$6)  
  
//Code for the method
```

Alternatively, you can use the [named parameters](#) syntax:

```
// On Web Connection database method
#DECLARE ($url : Text; $header : Text; \
$BrowserIP : Text; $ServerIP : Text; \
$user : Text; $password : Text)
```

Calling a 4D command that displays an interface element (`DIALOG` , `ALERT` , etc.) is not allowed and ends the method processing.

\$1 - URL extra data

The first parameter (`$1`) is the URL entered by users in the address area of their web browser, without the host address.

Let's use an intranet connection as an example. Suppose that the IP address of your 4D Web Server machine is 123.4.567.89. The following table shows the values of `$1` depending on the URL entered in the web browser:

URL entered in web browser	Value of parameter <code>\$1</code>
123.4.567.89	/
http://123.4.567.89	/
123.4.567.89/Customers	/Customers
http://123.4.567.89/Customers/Add	/Customers/Add
123.4.567.89/Do_This/If_OK/Do_That	/Do_This/If_OK/Do_That

Note that you are free to use this parameter at your convenience. 4D simply ignores the value passed beyond the host part of the URL. For example, you can establish a convention where the value `"/Customers/Add"` means "go directly to add a new record in the `[Customers]` table." By supplying the web users with a list of possible values and/or default bookmarks, you can provide shortcuts to different parts of your application. By supplying the web users with a list of possible values and/or default bookmarks, you can provide shortcuts to different parts of your application. This way, web users can quickly access resources of your website without going through the entire navigation path each time they make a new connection.

\$2 - Header and Body of the HTTP request

The second parameter (`$2`) is the header and the body of the HTTP request sent by the web browser. Note that this information is passed to your `On Web Connection` database method "as is". Its contents will vary depending on the nature of the web browser attempting the connection.

If your application uses this information, it is up to you to parse the header and the body. You can use the `WEB GET` `HTTP HEADER` and the `WEB GET HTTP BODY` commands.

For performance reasons, the size of data passing through the `$2` parameter must not exceed 32 KB. Beyond this size, they are truncated by the 4D HTTP server.

\$3 - Web client IP address

The `$3` parameter receives the IP address of the browser's machine. This information can allow you to distinguish between intranet and internet connections.

4D returns IPv4 addresses in a hybrid IPv6/IPv4 format written with a 96-bit prefix, for example `::ffff:192.168.2.34` for the IPv4 address 192.168.2.34. For more information, refer to the [IPv6 Support](#) section.

\$4 - Server IP address

The \$4 parameter receives the IP address requested by the 4D Web Server. 4D allows for multi-homing, which allows you to use machines with more than one IP address. For more information, please refer to the [Configuration page](#).

\$5 and \$6 - User Name and Password

The \$5 and \$6 parameters receive the user name and password entered by the user in the standard identification dialog box displayed by the browser, if applicable (see the [authentication page](#)).

If the user name sent by the browser exists in 4D, the \$6 parameter (the user's password) is not returned for security reasons.

/4DACTION

/4DACTION/MethodName***
*/4DACTION/*****MethodName/Param

Paramètres	Type		Description
MethodName	Text	->	Name of the 4D project method to be executed
Param	Text	->	Text parameter to pass to the project method

Usage: URL or Form action.

This URL allows you to call the *MethodName* 4D project method with an optional *Param* text parameter. The method will receive this parameter in \$1.

- The 4D project method must have been [allowed for web requests](#): the "Available through 4D tags and URLs (4DACTION...)" attribute value must have been checked in the properties of the method. If the attribute is not checked, the web request is rejected.
- When 4D receives a /4DACTION/MethodName/Param request, the `On Web Authentication` database method (if it exists) is called.

4DACTION/ can be associated with a URL in a static Web page:

```
<A HREF="/4DACTION/MyMethod/hello">Do Something</A>
```

The `MyMethod` project method should generally return a "reply" (sending of an HTML page using `WEB SEND FILE` or `WEB SEND TEXT`, etc.). Be sure to make the processing as short as possible in order not to block the browser.

A method called by `/4DACTION` must not call interface elements (`DIALOG`, `ALERT`, etc.).

Exemple

This example describes the association of the `/4DACTION` URL with an HTML picture object in order to dynamically display a picture in the page. You insert the following instructions in a static HTML page:

```
<IMG SRC="/4DACTION/getPhoto/smith">
```

The `getPhoto` method is as follows:

```

C_TEXT($1) // This parameter must always be declared
var $path : Text
var $PictVar : Picture
var $BlobVar : Blob

//find the picture in the Images folder within the Resources folder
$path:=Get 4D folder(Current resources folder)+"Images"+Folder separator+$1+".psd"

READ PICTURE FILE($path;$PictVar) //put the picture in the picture variable
PICTURE TO BLOB($PictVar;$BLOB;".png") //convert the picture to ".png" format
WEB SEND BLOB($BLOB;"image/png")

```

4DACTION to post forms

The 4D Web server also allows you to use “posted” forms, which are static HTML pages that send data to the Web server, and to easily retrieve all the values. The POST type must be associated to them and the form’s action must imperatively start with /4DACTION/MethodName.

A form can be submitted through two methods (both can be used with 4D):

- POST, usually used to send data to the Web server,
- GET, usually used to request data from the Web server.

When the Web server receives a posted form, it calls the `On Web Authentication` database method (if it exists).

In the called method, you must call the `WEB GET VARIABLES` command in order to [retrieve the names and values](#) of all the fields included in an HTML page submitted to the server.

Example to define the action of a form:

```
<FORM ACTION="/4DACTION/MethodName" METHOD=POST>
```

Exemple

In a Web application, we would like for the browsers to be able to search among the records by using a static HTML page. This page is called “search.htm”. The application contains other static pages that allow you to, for example, display the search result (“results.htm”). The POST type has been associated to the page, as well as the `/4DACTION/SEARCH` action.

Here is the HTML code that corresponds to this page:

```

<form action="/4action/processForm" method=POST>
<input type=text name=vName value=""><BR>
<input type=checkbox name=vExact value="Word">Whole word<BR>
<input type=submit name=OK value="Search">
</FORM>

```

During data entry, type “ABCD” in the data entry area, check the “Whole word” option and validate it by clicking the Search button. In the request sent to the Web server:

```

vName="ABCD"
vExact="Word"
OK="Search"

```

4D calls the `On Web Authentication` database method (if it exists), then the `processForm` project method is called, which is as follows:

```

C_TEXT($1) //mandatory for compiled mode
C_LONGINT($vName)
C_TEXT(vName;vLIST)
ARRAY TEXT($arrNames;0)
ARRAY TEXT($arrVals;0)
WEB GET VARIABLES($arrNames;$arrVals) //we retrieve all the variables of the form
$vName:=Find in array($arrNames;"vName")
vName:=$arrVals{$vName}
If(Find in array($arrNames;"vExact")=-1) //If the option has not been checked
    vName:=vName+"@"
End if
QUERY([Jockeys];[Jockeys]Name=vName)
FIRST RECORD([Jockeys])
While(Not(End selection([Jockeys])))
    vLIST:=vLIST+[Jockeys]Name+" "+[Jockeys]Tel+"  
"
    NEXT RECORD([Jockeys])
End while
WEB SEND FILE("results.htm") //Send the list to the results.htm form
//which contains a reference to the variable vLIST,
//for example <!--4DHTML vLIST-->
//...
End if

```

Getting values from HTTP requests

4D's Web server lets you recover data sent through POST or GET requests, using Web forms or URLs.

When the Web server receives a request with data in the header or in the URL, 4D can retrieve the values of any HTML objects it contains. This principle can be implemented in the case of a Web form, sent for example using `WEB SEND FILE` or `WEB SEND BLOB`, where the user enters or modifies values, then clicks on the validation button.

In this case, 4D can retrieve the values of the HTML objects found in the request using the `WEB GET VARIABLES` command. The `WEB GET VARIABLES` command retrieves the values as text.

Consider the following HTML page source code:

```

<html>
<head>
    <title>Welcome</title>
    <script language="JavaScript"><!--
function GetBrowserInformation(formObj){
formObj.vtNav_appName.value = navigator.appName
formObj.vtNav_appVersion.value = navigator.appVersion
formObj.vtNav_appCodeName.value = navigator.appCodeName
formObj.vtNav_userAgent.value = navigator.userAgent
return true
}
function LogOn(formObj){
if(formObj.vtUserName.value!=""){
return true
} else {
alert("Enter your name, then try again.")
return false
}
}
//--></script>
</head>
<body>
<form action="/4DACTION/WWW_STD_FORM_POST" method="post"
name="frmWelcome"
onsubmit="return GetBrowserInformation(frmWelcome)">
<h1>Welcome to Spiders United</h1>

```

```

<p><b>Please enter your name:</b>
<input name="vtUserName" value="" size="30" type="text"></p>
<p>
<input name="vsbLogOn" value="Log On" onclick="return LogOn(frmWelcome)" type="submit">
<input name="vsbRegister" value="Register" type="submit">
<input name="vsbInformation" value="Information" type="submit"></p>
<p>
<input name="vtNav_appName" value="" type="hidden">
<input name="vtNav_appVersion" value="" type="hidden">
<input name="vtNav_appCodeName" value="" type="hidden">
<input name="vtNav_userAgent" value="" type="hidden"></p>
</form>
</body>
</html>
return false
}
}
//--></script>
</head>
<body>
<form action="/4DACTION/WWW_STD_FORM_POST" method="post"
name="frmWelcome"
onsubmit="return GetBrowserInformation(frmWelcome)">
<h1>Welcome to Spiders United</h1>
<p><b>Please enter your name:</b>
<input name="vtUserName" value="" size="30" type="text"></p>
<p>
<input name="vsbLogOn" value="Log On" onclick="return LogOn(frmWelcome)" type="submit">
<input name="vsbRegister" value="Register" type="submit">
<input name="vsbInformation" value="Information" type="submit"></p>
<p>
<input name="vtNav_appName" value="" type="hidden">
<input name="vtNav_appVersion" value="" type="hidden">
<input name="vtNav_appCodeName" value="" type="hidden">
<input name="vtNav_userAgent" value="" type="hidden"></p>
</form>
</body>
</html>

```

When 4D sends the page to a Web Browser, it looks like this:

The main features of this page are:

- It includes three Submit buttons: `vsbLogOn`, `vsbRegister` and `vsbInformation`.
- When you click Log On, the submission of the form is first processed by the JavaScript function `LogOn`. If no name is entered, the form is not even submitted to 4D, and a JavaScript alert is displayed.
- The form has a POST 4D method as well as a Submit script (`GetBrowserInformation`) that copies the browser properties to the four hidden objects whose names starts with `vtNav_App`. It also includes the `vtUserName` object.

Let's examine the 4D method `WWW_STD_FORM_POST` that is called when the user clicks on one of the buttons on the HTML form.

```

// Retrieval of value of variables
ARRAY TEXT($arrNames;0)
ARRAY TEXT($arrValues;0)
WEB GET VARIABLES($arrNames;$arrValues)
C_TEXT($user)

Case of

// The Log On button was clicked
:(Find in array($arrNames;"vsbLogOn")#-1)
$user :=Find in array($arrNames;"vtUserName")
QUERY([WWW Users];[WWW Users]UserName=$arrValues{$user})
$0:=(Records in selection([WWW Users])>0)
If($0)
    WWW POST EVENT("Log On";WWW Log information)
// The WWW POST EVENT method saves the information in a database table
Else

    $0:=WWW Register
// The WWW Register method lets a new Web user register
End if

// The Register button was clicked
:(Find in array($arrNames;"vsbRegister")#-1)
$0:=WWW Register

// The Information button was clicked
:(Find in array($arrNames;"vsbInformation")#-1)
WEB SEND FILE("userinfos.html")
End case

```

The features of this method are:

- The values of the variables *vtNav_appName*, *vtNav_appVersion*, *vtNav_appCodeName*, and *vtNav_userAgent* (bound to the HTML objects having the same names) are retrieved using the `WEB GET VARIABLES` command from HTML objects created by the *GetBrowserInformation* JavaScript script.
- Out of the *vsbLogOn*, *vsbRegister* and *vsbInformation* variables bound to the three Submit buttons, only the one corresponding to the button that was clicked will be retrieved by the `WEB GET VARIABLES` command. When the submit is performed by one of these buttons, the browser returns the value of the clicked button to 4D. This tells you which button was clicked.

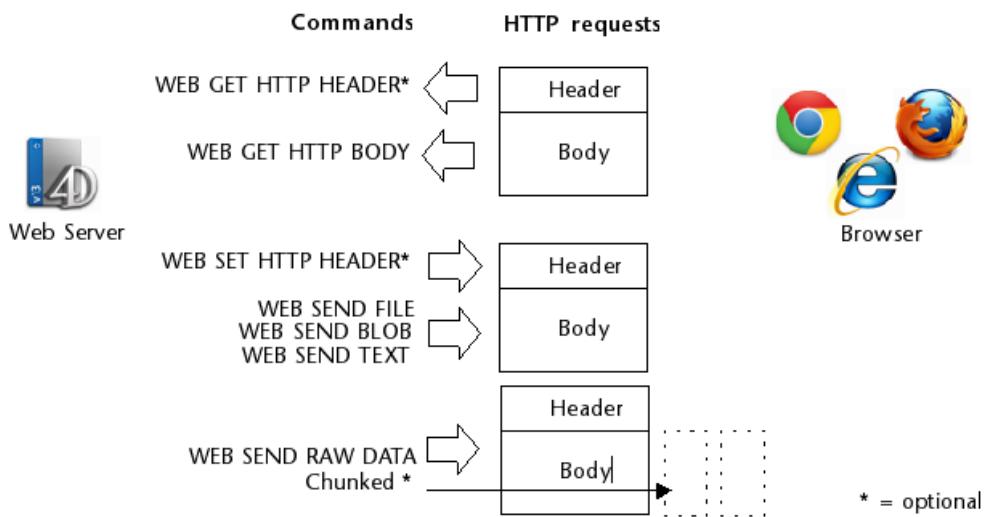
Keep in mind that with HTML, all objects are text objects. If you use a SELECT object, it is the value of the highlighted element in the object that is returned in the `WEB GET VARIABLES` command, and not the position of the element in the array as in 4D. `WEB GET VARIABLES` always returns values of the Text type.

Other Web Server Commands

The 4D web server provides several low-level web commands allowing you to develop custom processing of requests:

- the `WEB GET HTTP BODY` command returns the body as raw text, allowing any parsing you may need
- the `WEB GET HTTP HEADER` command return the headers of the request. It is useful to handle custom cookies, for example (along with the `WEB SET HTTP HEADER` command).
- the `WEB GET BODY PART` and `WEB Get body part count` commands to parse the body part of a multi-part request and retrieve text values, but also files posted, using BLOBs.

These commands are summarized in the following graphic:



The 4D web server supports files uploaded in chunked transfer encoding from any Web client. Chunked transfer encoding is a data transfer mechanism specified in HTTP/1.1. It allows data to be transferred in a series of "chunks" (parts) without knowing the final data size. The 4D Web Server also supports chunked transfer encoding from the server to Web clients (using `WEB SEND RAW DATA`).

COMPILER_WEB Project Method

The `COMPILER_WEB` method, if it exists, is systematically called when the HTTP server receives a dynamic request and calls the 4D engine. This is the case, for example, when the 4D Web server receives a posted form or a URL to process in [On Web Connection](#). This method is intended to contain typing and/or variable initialization directives used during Web exchanges. It is used by the compiler when the application is compiled. The `COMPILER_WEB` method is common to all the Web forms. By default, the `COMPILER_WEB` method does not exist. You must explicitly create it.

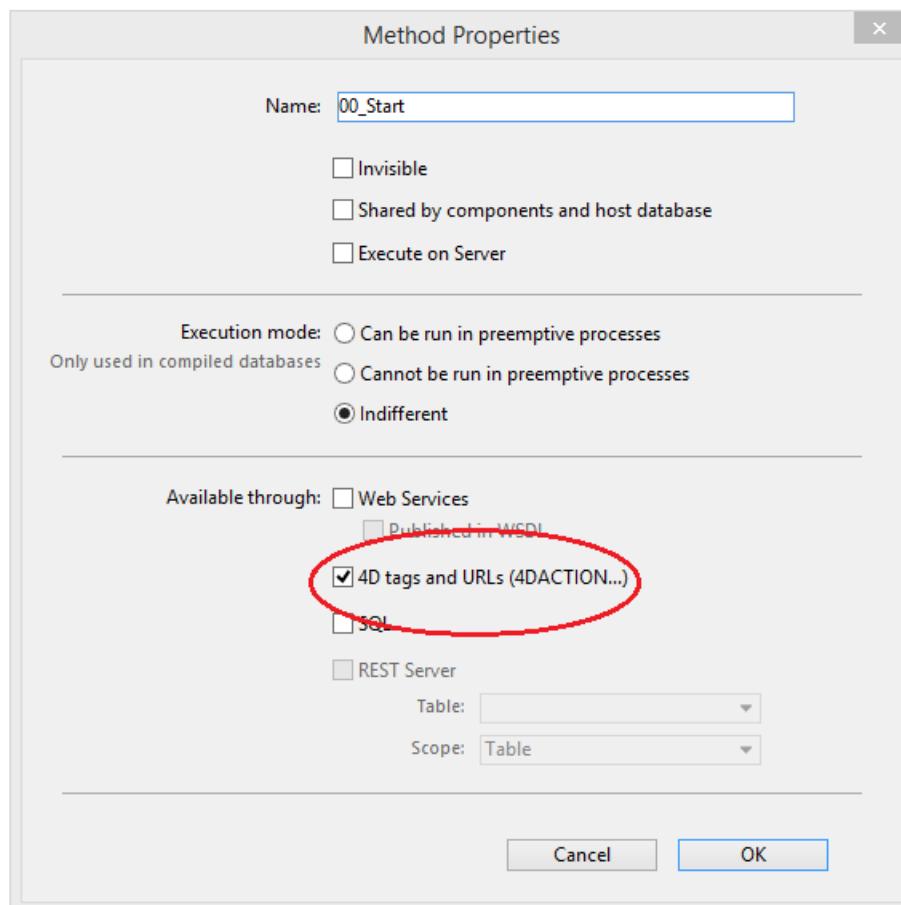
The `COMPILER_WEB` project method is also called, if it exists, for each SOAP request accepted.

Allowing project methods

Les balises 4D telles que `4DEVAL`, `4DTEXT`, `4DHTML`, etc. ainsi que l'[URL /4ACTION](#) vous permettent de déclencher l'exécution de toute méthode projet d'un projet 4D publié sur le Web. Par exemple, la requête <http://www.server.com/4ACTION/login> entraîne l'exécution de la méthode projet `login`, si elle existe.

Ce mécanisme présente donc un risque de sécurité pour l'application, notamment si un internaute déclenche intentionnellement (ou non) une méthode non destinée à être exécutée via le web. Vous pouvez éviter ce risque comme suit :

- Filtrez les méthodes appelées via les URL à l'aide de la méthode base [On Web Authentication](#). Inconvénients : si la base de données comprend un grand nombre de méthodes, ce système peut être difficile à gérer.
- Utilisez l'option Available through 4D tags and URLs (4ACTION...) de la boîte de dialogue Propriétés de la méthode :



This option is used to individually designate each project method that can be called using the `4ACTION` special URL, or the `4DTEXT`, `4DHTML`, `4DEVAL`, `4SCRIPT`, `4DIF`, `4ELSEIF` or `4LOOP` tags. When it is not checked, the project method concerned cannot be directly executed through an HTTP request. Conversely, it can be executed using other types of calls (formulas, other methods, etc.).

This option is unchecked by default. Methods that can be executed through `4ACTION` or specific tags must be specifically indicated.

In the Explorer, Project methods with this property are given a specific icon:



Pages d'erreur HTTP personnalisées

4D Web Server vous permet de personnaliser les pages d'erreur HTTP envoyées aux clients, en fonction du code d'état de la réponse du serveur. Les pages d'erreur font référence à :

- les codes d'état commençant par 4 (erreurs du client), par exemple 404
- les codes d'état commençant par 5 (erreurs du serveur), par exemple 501.

Pour une description complète des codes d'état d'erreur HTTP, vous pouvez vous reporter à la [liste des codes d'état HTTP](#) (Wikipedia).

Remplacement des pages par défaut

Pour remplacer les pages d'erreur par défaut de 4D Web Server par vos propres pages, il vous suffit de :

- placer des pages HTML personnalisées au premier niveau du dossier web de l'application,
- nommer les pages personnalisées "{statusCode}.html" (par exemple, "404.html").

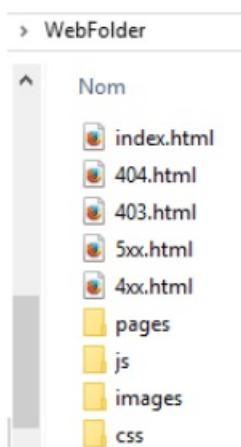
Vous pouvez définir une page d'erreur par code d'état et/ou une page d'erreur générique pour une série d'erreurs, nommée "{number}xx.html". Par exemple, vous pouvez créer "4xx.html" pour les erreurs génériques du client. 4D Web Server recherchera d'abord une page {statusCode}.html puis, si elle n'existe pas, une page générique.

Par exemple, lorsqu'une réponse HTTP retourne un code d'état 404 :

1. 4D Web Server essaie d'envoyer une page "404.html" située dans le dossier web de l'application.
2. Si elle n'est pas trouvée, 4D Web Server essaie d'envoyer une page "4xx.html" située dans le dossier Web de l'application.
3. Si elle n'est pas trouvée, 4D Web Server utilise alors sa page d'erreur par défaut.

Exemple

Si vous définissez les pages personnalisées suivantes dans votre dossier web :



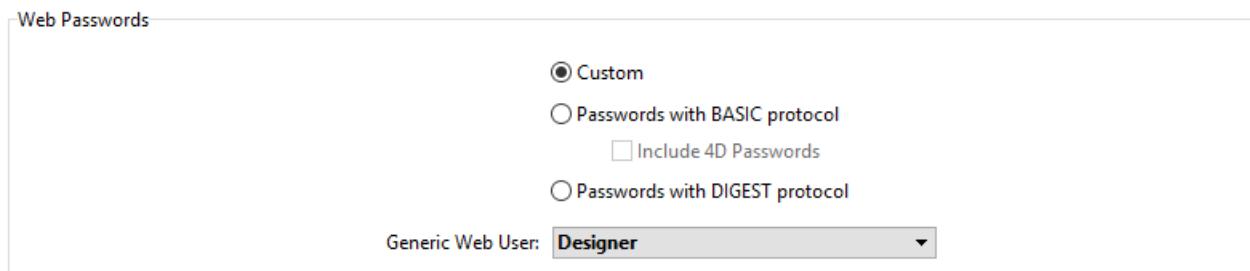
- les pages "403.html" ou "404.html" seront servies lorsque des réponses HTTP 403 ou 404 sont retournées respectivement,
- la page "4xx.html" sera servie pour tout autre état d'erreur 4xx (400, 401, etc.),
- la page "5xx.html" sera servie pour tout état d'erreur 5xx.

Authentification

L'authentification est nécessaire lorsque vous souhaitez fournir des droits d'accès spécifiques aux utilisateurs Web. L'authentification désigne la manière dont les informations concernant les références de l'utilisateur (généralement le nom et le mot de passe) sont collectées et traitées.

Modes d'authentification

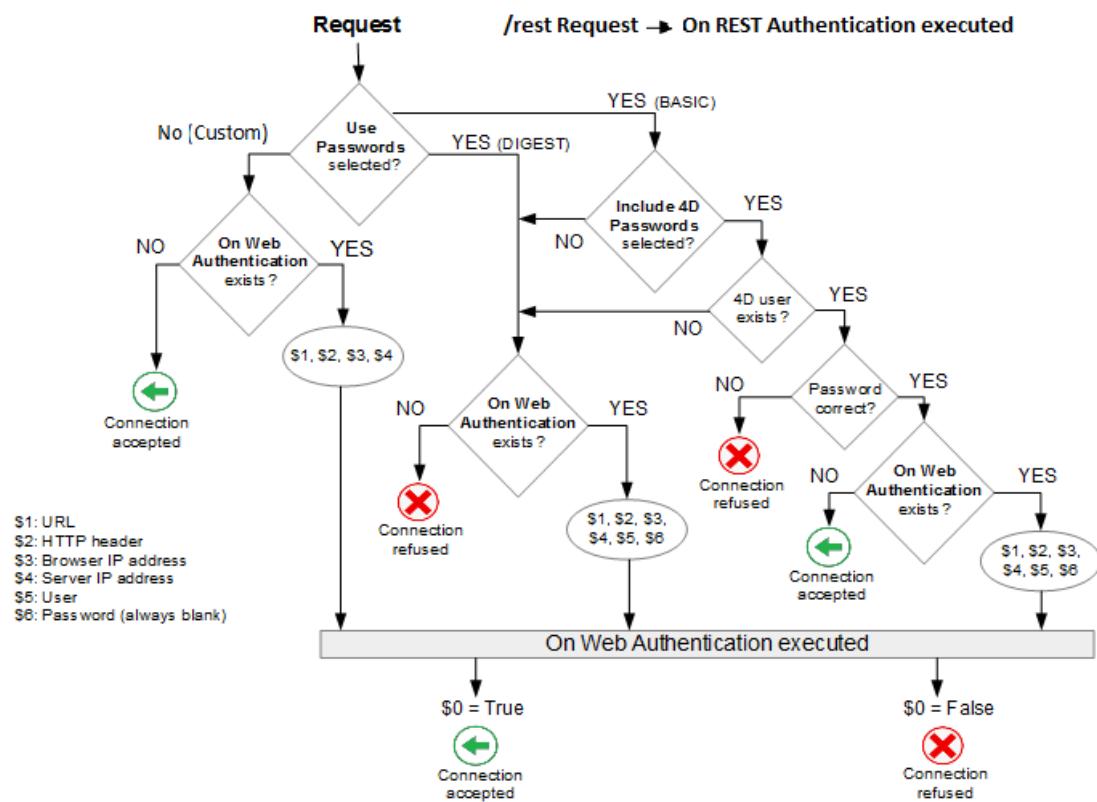
Le serveur web 4D propose trois modes d'authentification que vous pouvez sélectionner sur la page Web/Options (I) de la fenêtre des Propriétés :



Il est recommandé d'utiliser une authentification personnalisée.

Aperçu

Le schéma suivant résume le système d'accès au serveur web 4D :



Les requêtes qui commencent par `rest/` sont gérées directement par le [serveur REST](#).

Authentification personnalisée (par défaut)

Dans ce mode, c'est au développeur de définir comment authentifier les utilisateurs. 4D évalue uniquement les requêtes HTTP [qui nécessitent une authentification](#).

Ce mode d'authentification est le plus flexible car il permet de :

- soit déléguer l'authentification de l'utilisateur à une application tierce (par exemple, un réseau social, un SSO);
- ou alors, fournir une interface à l'utilisateur (par exemple, un formulaire web) pour qu'il puisse créer son compte dans votre base de données clients ; ensuite, vous pouvez authentifier les utilisateurs avec n'importe quel algorithme personnalisé (voir [cette exemple](#) dans le chapitre "Sessions Utilisateur"). L'important est de ne jamais stocker le mot de passe en clair, en utilisant du code tel que :

```
//... user account creation  
ds.webUser.password:=Generate password hash($password)  
ds.webUser.save()
```

Voir également [cet exemple](#) du chapitre "Prise en main".

Si aucune authentification personnalisée n'est fournie, 4D appelle la méthode de base de données [On Web Authentication](#) (si elle existe). En plus de \$1 et \$2, seules les adresses IP du navigateur et du serveur (\$3 et \$4) sont fournies, le nom d'utilisateur et le mot de passe (\$5 et \$6) sont vides. La méthode doit retourner True dans \$0 si l'utilisateur est authentifié avec succès. Ensuite, la ressource qui fait l'objet de la requête est fournie. Si l'authentification échoue, False est retourné dans \$0.

Attention: Si la méthode de base de données [On Web Authentication](#) n'existe pas, les connexions sont automatiquement acceptées (mode test).

Protocole BASIC

Lorsqu'un utilisateur se connecte au serveur, une boîte de dialogue standard apparaît sur son navigateur afin qu'il saisisse son nom d'utilisateur et son mot de passe.

Le nom et le mot de passe saisis par l'utilisateur sont envoyés en clair dans l'en-tête de la requête HTTP. Ce mode requiert généralement le protocole HTTPS pour assurer la confidentialité.

Les valeurs saisies sont ensuite évaluées :

- Si l'option Inclure les mots de passe 4D est cochée, les informations d'identification des utilisateurs seront d'abord évaluées par rapport à la [table interne des utilisateurs 4D](#).
 - Si le nom d'utilisateur envoyé par le navigateur existe dans la table des utilisateurs 4D et que le mot de passe est correct, la connexion est acceptée. Si le mot de passe est incorrect, la connexion est refusée.
 - Si le nom de l'utilisateur n'existe pas dans la table des utilisateurs 4D, la méthode base [On Web Authentication](#) est appelée. Si la méthode base [On Web Authentication](#) n'existe pas, les connexions sont rejetées.
- Si l'option Inclure les mots de passe 4D n'est pas cochée, les informations d'identification des utilisateurs sont envoyées à la méthode base [On Web Authentication](#) avec les autres paramètres de connexion (adresse et port IP, URL...) afin de pouvoir les traiter. Si la méthode base [On Web Authentication](#) n'existe pas, les connexions sont rejetées.

Avec le serveur Web du client 4D, gardez à l'esprit que tous les sites publiés par les machines 4D Client partageront la même table d'utilisateurs. La validation des utilisateurs/mots de passe est effectuée par l'application 4D Server.

Mots de passe protocole DIGEST

Ce mode offre un niveau de sécurité plus élevé car les informations d'authentification sont traitées par un processus à sens unique appelé "hashing" qui rend leur contenu impossible à déchiffrer.

Comme en mode BASIC, l'utilisateur doit saisir son nom et mot de passe lors de la connexion. Ensuite, la méthode base [On Web Authentication](#) est appelée. Lorsque le mode DIGEST est activé, le paramètre \$6 (mot de passe) est toujours retourné vide. En effet, lors de l'utilisation de ce mode, ces informations ne passent pas par le réseau en texte clair (non chiffré). Il est donc impératif dans ce cas d'évaluer les demandes de connexion à l'aide de la commande [WEB Validate digest](#).

Vous devez redémarrer le serveur web pour que les modifications apportées à ces paramètres soient prises en compte.

On Web Authentication

La méthode de base de données [On Web Authentication](#) est chargée de gérer l'accès au moteur du serveur web. Elle est appelée par 4D ou 4D Server lorsqu'une requête HTTP dynamique est reçue.

Database method calls

The [On Web Authentication](#) database method is automatically called when a request or processing requires the execution of some 4D code (except for REST calls). It is also called when the web server receives an invalid static URL (for example, if the static page requested does not exist).

The [On Web Authentication](#) database method is therefore called:

- when the web server receives a URL requesting a resource that does not exist
- when the web server receives a URL beginning with `4DACTION/`, `4DCGI/` ...
- when the web server receives a root access URL and no home page has been set in the Settings or by means of the [WEB SET HOME PAGE](#) command
- when the web server processes a tag executing code (e.g. `4DSCRIPT`) in a semi-dynamic page.

The [On Web Authentication](#) database method is NOT called:

- when the web server receives a URL requesting a valid static page.
- when the web server receives a URL beginning with `rest/` and the REST server is launched (in this case, the authentication is handled through the [On REST Authentication database method](#) or [Structure settings](#)).

Syntaxe

`On Web Authentication($1 : Text ; $2 : Text ; $3 : Text ; $4 : Text ; $5 : Text ; $6 : Text) -> $0 : Boolean`

Paramètres	Type		Description
\$1	Text	<-	Variable URL
\$2	Text	<-	HTTP headers + HTTP body (up to 32 kb limit)
\$3	Text	<-	IP address of the web client (browser)
\$4	Text	<-	Adresse IP du serveur
\$5	Text	<-	Nom d'utilisateur
\$6	Text	<-	Password
\$0	Booléen	->	True = request accepted, False = request rejected

You must declare these parameters as follows:

```
//On Web Authentication database method
```

```
C_TEXT($1;$2;$3;$4;$5;$6)  
C_BOOLEAN($0)
```

```
//Code for the method
```

Alternatively, you can use the [named parameters](#) syntax:

```
// On Web Authentication database method  
#DECLARE ($url : Text; $header : Text; \  
$BrowserIP : Text; $ServerIP : Text; \  
$user : Text; $password : Text) \  
-> $RequestAccepted : Boolean
```

All the `On Web Authentication` database method's parameters are not necessarily filled in. The information received by the database method depends on the selected [authentication mode](#)).

\$1 - URL

The first parameter (`$1`) is the URL received by the server, from which the host address has been removed.

Let's take the example of an Intranet connection. Suppose that the IP address of your 4D Web Server machine is 123.45.67.89. The following table shows the values of `$1` depending on the URL entered in the Web browser:

URL entered in web browser	Value of parameter \$1
123.45.67.89	/
http://123.45.67.89	/
123.45.67.89/Customers	/Customers
http://123.45.67.89/Customers/Add	/Customers/Add
123.45.67.89/Do_This/If_OK/Do_That	/Do_This/If_OK/Do_That

\$2 - Header and Body of the HTTP request

The second parameter (`$2`) is the header and the body of the HTTP request sent by the web browser. Note that this information is passed to your `On Web Authentication` database method as it is. Its contents will vary depending on the nature of the web browser which is attempting the connection.

If your application uses this information, it is up to you to parse the header and the body. You can use the `WEB GET HTTP HEADER` and the `WEB GET HTTP BODY` commands.

For performance reasons, the size of data passing through the `$2` parameter must not exceed 32 KB. Beyond this size, they are truncated by the 4D HTTP server.

\$3 - Web client IP address

The `$3` parameter receives the IP address of the browser's machine. This information can allow you to distinguish between intranet and internet connections.

4D returns IPv4 addresses in a hybrid IPv6/IPv4 format written with a 96-bit prefix, for example `::ffff:192.168.2.34` for the IPv4 address `192.168.2.34`. For more information, refer to the [IPv6 Support](#) section.

\$4 - Server IP address

The \$4 parameter receives the IP address used to call the web server. 4D allows for multi-homing, which allows you to exploit machines with more than one IP address. Pour plus d'informations, veuillez consulter la [Page Configuration](#).

\$5 and \$6 - User Name and Password

The \$5 and \$6 parameters receive the user name and password entered by the user in the standard identification dialog box displayed by the browser. This dialog box appears for each connection, if **basic** or **digest** authentication is selected.

If the user name sent by the browser exists in 4D, the \$6 parameter (the user's password) is not returned for security reasons.

\$0 parameter

The **On Web Authentication** database method returns a boolean in \$0:

- If \$0 is True, the connection is accepted.
- If \$0 is False, the connection is refused.

The **On Web Connection** database method is only executed if the connection has been accepted by **On Web Authentication**.

WARNING

If no value is set to \$0 or if \$0 is not defined in the **On Web Authentication** database method, the connection is considered as accepted and the **On Web Connection** database method is executed.

- Do not call any interface elements in the **On Web Authentication** database method (**ALERT** , **DIALOG** , etc.) because otherwise its execution will be interrupted and the connection refused. The same thing will happen if an error occurs during its processing.

Exemple

Example of the **On Web Authentication** database method in **DIGEST mode**:

```
// On Web Authentication Database Method
#DECLARE ($url : Text; $header : Text; $ipB : Text; $ipS : Text; \
$user : Text; $pw : Text) -> $valid : Boolean

var $found : cs.WebUserSelection
$valid:=False

$found:=ds.WebUser.query("User === :1";$user)
If($found.length=1) // User is found
  $valid:=WEB Validate digest($user;[WebUser]password)
Else
  $valid:=False // User does not exist
End if
```

Sessions utilisateur

Le serveur Web de 4D offre des fonctions intégrées pour la gestion des sessions utilisateur. Creating and maintaining user sessions allows you to control and improve the user experience on your web application. When user sessions are enabled, web clients can reuse the same server context from one request to another.

Web server user sessions allow to:

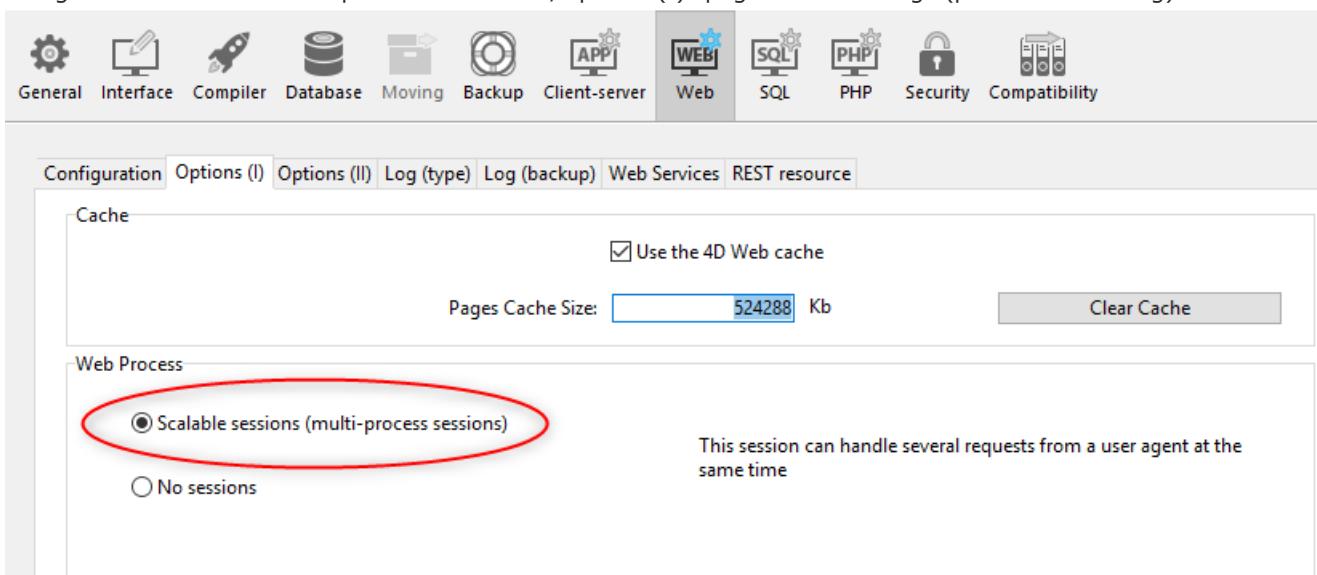
- handle multiple requests simultaneously from the same web client through an unlimited number of preemptive processes (web server sessions are scalable),
- share data between the processes of a web client,
- associate privileges to user sessions,
- gérer l'accès via un objet `Session` et l'[API Session](#).

Note : l'implémentation actuelle n'est que la première étape d'une fonctionnalité complète à venir qui permet aux développeurs de gérer les autorisations utilisateur hiérarchiques via des sessions dans l'ensemble de l'application Web.

Activation des sessions

The session management feature can be enabled and disabled on your 4D web server. There are different ways to enable session management:

- Using the Scalable sessions option on the "Web/Options (I)" page of the Settings (permanent setting):



This option is selected by default in new projects. It can however be disabled by selecting the `No sessions` option, in which case the web session features are disabled (no `Session` object is available).

- Utilisation de la propriété `.scalableSession` de l'objet Web Server (pour passer le paramètre `settings` de la fonction `.start()`). In this case, this setting overrides the option defined in the Settings dialog box for the Web Server object (it is not stored on disk).

The `WEB SET OPTION` command can also set the session mode for the main Web server.

In any cases, the setting is local to the machine; so it can be different on the 4D Server Web server and the Web servers of remote 4D machines.

Compatibility: A Legacy sessions option is available in projects created with a 4D version prior to 4D v18 R6 (for more information, please refer to the [doc.4d.com](#) web site).

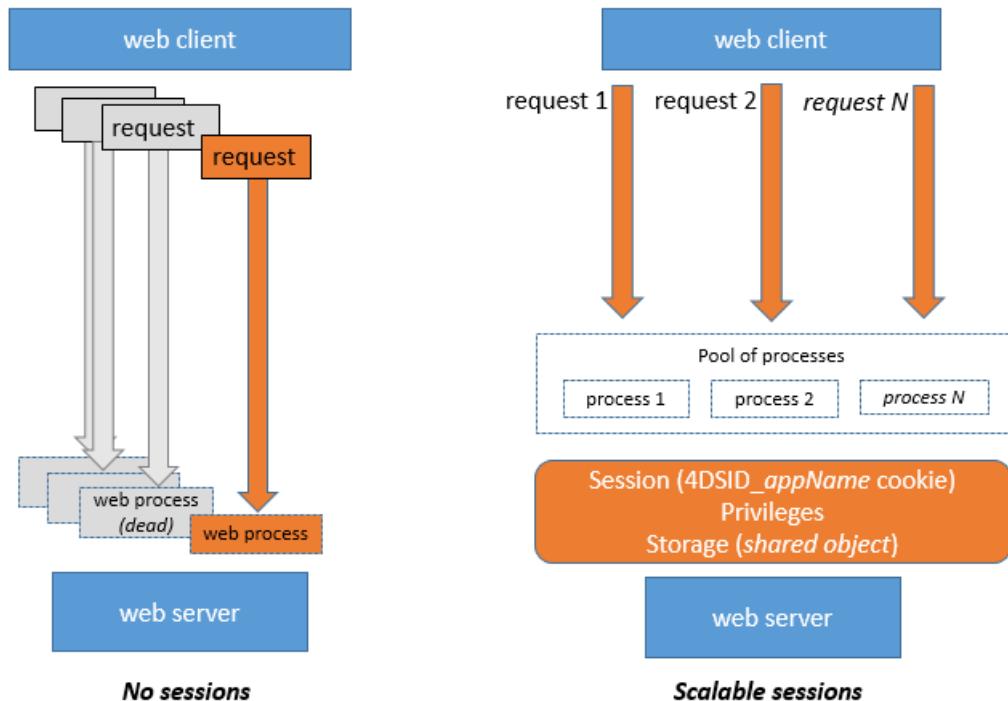
Session implementation

When [sessions are enabled](#), automatic mechanisms are implemented, based upon a private cookie set by 4D itself: "4DSID_AppName", where *AppName* is the name of the application project. This cookie references the current web session for the application.

Le nom du cookie peut être obtenu à l'aide de la propriété `.sessionCookieName`.

1. In each web client request, the Web server checks for the presence and the value of the private "4DSID_AppName" cookie.
2. If the cookie has a value, 4D looks for the session that created this cookie among the existing sessions; if this session is found, it is reused for the call.
3. If the client request does not correspond to an already opened session:
 - a new session with a private "4DSID_AppName" cookie is created on the web server
 - a new Guest `Session` object is created and is dedicated to the scalable web session.

L'objet `Session` courant est alors accessible via la commande `Session` dans le code de n'importe quel processus Web.



Web processes usually do not end, they are recycled in a pool for efficiency. When a process finishes executing a request, it is put back in the pool and made available for the next request. Since a web process can be reused by any session, [process variables](#) must be cleared by your code at the end of its execution (using `CLEAR VARIABLE` for example). This cleanup is necessary for any process related information, such as a reference to an opened file. This is the reason why it is recommended to use the `Session` object when you want to keep session related information.

Mode préemptif

Sur 4D Server, les sessions du serveur Web sont automatiquement gérées par des process préemptifs, y compris en mode interprété. Vous devez vous assurer que le code de votre serveur web est [conforme à une exécution préemptive](#).

Pour déboguer le code web interprété sur la machine serveur, assurez-vous que le débogueur est [rattaché au serveur ou à une machine distante](#). Les process Web passent alors en mode coopératif et le code du serveur Web peut être débogué.

Avec 4D monoposte, le code interprété s'exécute toujours en mode coopératif.

Partage d'informations

Chaque objet `Session` fournit une propriété `.storage` qui est un `objet partagé`. This property allows you to share information between all processes handled by the session.

Session lifetime

A scalable web session is closed when:

- the web server is stopped,
- the timeout of the session cookie has been reached.

The lifespan of an inactive cookie is 60 minutes by default, which means that the web server will automatically close inactive sessions after 60 minutes.

Ce timeout peut être défini à l'aide de la propriété `.idleTimeout` de l'objet `Session` (le timeout ne peut pas être inférieur à 60 minutes).

Lorsqu'une session Web évolutive est fermée, si la commande `Session` est appelée par la suite :

- the `Session` object does not contain privileges (it is a Guest session)
- la propriété `.storage` est vide
- a new session cookie is associated to the session

Privileges

Privileges can be associated to sessions. On the web server, you can provide specific access or features depending on the privileges of the session.

Vous pouvez attribuer des privilèges à l'aide de la fonction `.setPrivileges()`. Dans votre code, vous pouvez vérifier les privilèges de la session pour autoriser ou refuser l'accès à l'aide de la fonction `.hasPrivilege()`. Par défaut, les nouvelles sessions n'ont aucun privilège : ce sont des sessions invité (la fonction `.isGuest()` retourne true).

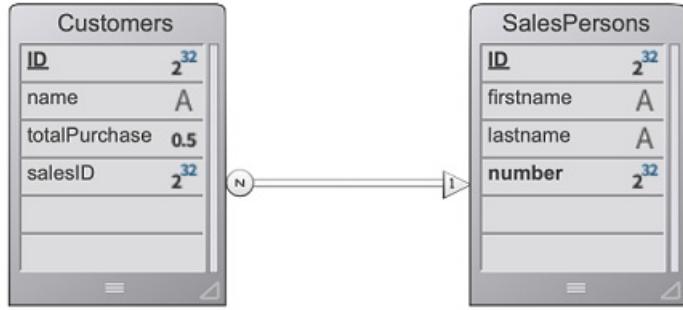
Dans l'implémentation actuelle, seul le privilège "WebAdmin" est disponible.

Exemple :

```
If (Session.hasPrivilege("WebAdmin"))
    //Access is granted, do nothing
Else
    //Display an authentication page
End if
```

Exemple

In a CRM application, each salesperson manages their own client portfolio. The datastore contains at least two linked dataclasses: Customers and SalesPersons (a salesperson has several customers).



We want a salesperson to authenticate, open a session on the web server, and have the top 3 customers be loaded in the session.

1. We run this URL to open a session:

```
http://localhost:8044/authenticate.shtml
```

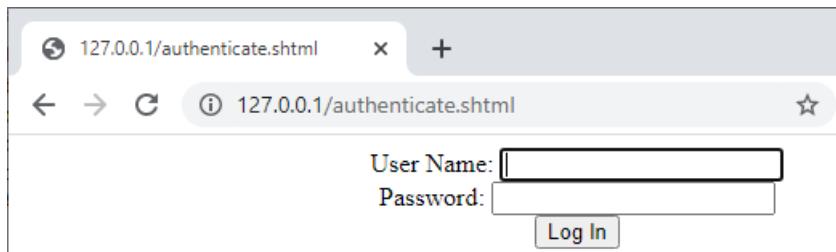
Dans un environnement de production, il est nécessaire d'utiliser une [connexion HTTPS](#) pour éviter la circulation d'informations non chiffrées sur le réseau.

2. The `authenticate.shtml` page is a form containing `userId` et `password` input fields and sending a 4DACTION POST action:

```

<!DOCTYPE html>
<html>
<body bgcolor="#ffffff">
<FORM ACTION="/4DACTION/authenticate" METHOD=POST>
    UserId: <INPUT TYPE=TEXT NAME=userId VALUE=""><BR>
    Password: <INPUT TYPE=TEXT NAME=password VALUE=""><BR>
<INPUT TYPE=SUBMIT NAME=OK VALUE="Log In">
</FORM>
</body>
</html>

```



3. The authenticate project method looks for the `userID` person and validates the password against the hashed value already stored in the `SalesPersons` table:

```

var $indexUserId; $indexPassword; $userId : Integer
var $password : Text
var $userTop3; $sales; $info : Object

ARRAY TEXT($anames; 0)
ARRAY TEXT($avalues; 0)

WEB GET VARIABLES($anames; $avalues)

$indexUserId:=Find in array($anames; "userId")
$userId:=Num($avalues{$indexUserId})

$indexPassword:=Find in array($anames; "password")
$password:=$avalues{$indexPassword}

$sales:=ds.SalesPersons.query("userId = :1"; $userId).first()

If ($sales#Null)
    If (Verify password hash($password; $sales.password))
        $info:=New object()
        $info.userName:=$sales.firstname+" "+$sales.lastname
        Session.setPrivileges($info)
        Use (Session.storage)
            If (Session.storage.myTop3=Null)
                $userTop3:=$sales.customers.orderBy("totalPurchase desc").slice(0; 3)
                Session.storage.myTop3:=$userTop3
            End if
        End use
        WEB SEND HTTP REDIRECT("/authenticationOK.shtml")
    Else
        WEB SEND TEXT("This password is wrong")
    End if
Else
    WEB SEND TEXT("This userId is unknown")
End if

```

Using preemptive web processes

Le Web Server de 4D vous permet de tirer pleinement parti des ordinateurs multi-coeurs en utilisant des process Web préemptifs dans vos applications compilées. Vous pouvez configurer votre code lié au Web, y compris les balises 4D, les méthodes base Web ou les fonctions de classe REST de ORDA, afin qu'il s'exécute simultanément sur le plus grand nombre de coeurs possibles.

Pour plus d'informations sur la fonctionnalité des process préemptifs dans 4D, veuillez vous référer à la section *Process 4D préemptifs* du [manuel de language 4D](#).

Availability of preemptive mode for web processes

Le tableau suivant permet d'indiquer si l'utilisation du mode préemptif pour les process Web est disponible dans le contexte d'exécution suivant :

4D Server	Interprété (associé au débogueur)	Interprété (non associé au débogueur)	Compilé
Serveur REST	coopératif	préemptif	préemptif
Serveur Web	coopératif	<i>paramètre web</i>	<i>paramètre web</i>
Server Web Services	coopératif	<i>paramètre web</i>	<i>paramètre web</i>

4D distant/monoposte	Interprété	Compilé
Serveur REST	coopératif	préemptif
Serveur Web	coopératif	<i>paramètre web</i>
Server Web Services	coopératif	<i>paramètre web</i>

- Serveur REST : gère les [fonctions de classe du modèle de données ORDA](#)
- Serveur Web : gère les [modèles Web](#), [4DACTION](#) et les [méthodes base](#)
- Serveur de services Web : gère les requêtes SOAP
- *web setting* signifie que le mode préemptif dépend d'une valeur de réglage :
 - lorsque l'option [Scalable sessions est sélectionnée](#), le [mode préemptif est automatiquement utilisé](#) pour les process web.
 - otherwise, the [Use preemptive processes](#) option is taken into account.
 - regarding Web service processes (server or client), preemptive mode is supported at method level. You just have to select "Can be run in preemptive processes" property for published SOAP server methods (see [Publishing a Web Service with 4D](#)) or proxy client methods (see [Subscribing to a Web Service in 4D](#)) and make sure they are confirmed thread-safe by the compiler.

Ecrire du code serveur Web thread-safe

Tout le code 4D exécuté par le serveur Web doit être thread-safe si vous souhaitez que les process Web soient lancés en mode préemptif. Lorsque l'option [Utiliser des process préemptifs](#) est cochée dans le dialogue des Propriétés, les parties de l'application listées ci-dessous sont automatiquement évaluées par 4D Compiler :

- Toutes les méthodes base liées au web :
 - [On Web Authentication](#)
 - [On Web Connection](#)
 - [On REST Authentication](#)
 - [On Mobile App Authentication](#) et [On Mobile App Action](#)

- La méthode projet `compiler_web` (indépendamment de sa propriété "Mode d'exécution") ;
- Tout code traité par la commande `PROCESS 4D TAGS` en contexte Web, par exemple via des pages `.shtml`
- Toute méthode projet comportant l'attribut "Available through 4D tags and URLs" (`4DACTION`, etc.)"
- Triggers pour les tables comportant l'attribut "Expose as REST resource"
- [Fonctions de classe du modèle de données ORDA](#) appelées via REST

Pour chacune de ces méthodes ou parties de code, le compilateur vérifiera si les règles thread-safe sont respectées, et retournera une erreur en cas de problème. Pour plus d'informations à propos des règles thread-safe, veuillez vous référer au paragraphe *Ecrire une méthode thread-safe* dans le chapitre *Process* du manuel de [Langage 4D](#).

Code web 4D thread-safe

La plupart des commandes et fonctions 4D, des méthodes base et des URL 4D sont thread-safe et peuvent être utilisées en mode préemptif.

4D commands and database methods

Toutes les commandes 4D relatives au Web sont thread-safe, à savoir :

- toutes les commandes du thème *Web Server*
- toutes les commandes du thème *Client HTTP*.

Les méthodes base ci-dessous sont thread-safe et peuvent être utilisées en mode préemptif (voir ci-dessus) : `On Web Authentication`, `On Web Connection`, `On REST Authentication` ...).

Bien sûr, le code exécuté par ces méthodes doit aussi être thread-safe.

Web Server URLs

Les URLs Web Server ci-dessous sont thread-safe et peuvent être utilisées en mode préemptif :

- `4daction/` (la méthode projet appelée doit également être à thread-safe)
- `4dcgi/` (les méthodes base appelées doivent également être thread-safe)
- `4dwebtest/`
- `4dblank/`
- `4dstats/`
- `4dhtmlstats/`
- `4dcache-clear/`
- `rest/`
- `4dimgfield/` (généré par `PROCESS 4D TAGS` pour les requêtes web relatives aux champs images)
- `4dimg/` (généré par `PROCESS 4D TAGS` pour les requêtes web sur les variables image)

Preemptive web process icon

Dans l'Explorateur d'exécution et dans la fenêtre d'administration de 4D Server, une icône spécifique s'affiche pour les process Web préemptifs :

Type de process	Icône
Méthode Web (process préemptif)	

Prise en main

4D vous fournit un serveur REST puissant, qui permet d'accéder directement aux données stockées dans vos applications 4D.

Le serveur REST est inclus dans 4D et 4D Server et automatiquement disponible dans vos applications 4D [une fois configuré](#).

Cette section est destinée à vous familiariser avec la fonctionnalité REST à l'aide d'un exemple simple. Nous allons :

- créer et configurer un projet d'application 4D basique
- accéder aux données du projet 4D via REST à l'aide d'un navigateur standard.

Pour simplifier l'exemple, nous allons utiliser 4D et un navigateur qui s'exécutent sur la même machine. Bien entendu, vous pouvez également utiliser une architecture distante.

Créer et configurer le projet 4D

1. Lancez votre application 4D ou 4D Server et créez un nouveau projet. Vous pouvez, par exemple, le nommer "Emp4D".
2. Dans l'éditeur de structure, créez une table [Employees] et ajoutez-y les champs suivants :
 - Lastname (Alpha)
 - Firstname (Alpha)
 - Salary (Longint)

Employees	
ID	2 ³²
Lastname	A
Firstname	A
Salary	2 ³²

L'option "Exposer une ressource REST" est cochée par défaut pour la table et pour chaque champ ; ne modifiez pas ce paramètre.

3. Créez des formulaires, puis créez quelques employés :

Emp4D - Employees: 3 of 3			
ID :	Lastname :	Firstname :	Salary :
1	Brown	Michael	25000
2	Jones	Maryanne	35000
3	Smithers	Jack	41000

4. Affichez la page Web > Web Features de la boîte de dialogue des Propriétés et [cochez l'option Exposer en tant que serveur REST](#).
5. Dans le menu Exécuter, sélectionnez Démarrer le serveur Web (si nécessaire), puis sélectionnez Tester le serveur Web.

4D affiche la page d'accueil par défaut du serveur Web 4D.

Accéder aux données 4D avec le navigateur

Vous pouvez désormais lire et modifier des données dans 4D uniquement via les requêtes REST.

Toute requête d'URL 4D REST commence par `/ rest`, pour être insérée après la zone `adress:port`. Par exemple, pour voir le contenu du datastore 4D, vous pouvez écrire :

```
http://127.0.0.1/rest/$catalog
```

Le serveur REST répond :

```
{
  "__UNIQID": "96A49F7EF2ABDE44BF32059D9ABC65C1",
  "dataClasses": [
    {
      "name": "Employees",
      "uri": "/rest/$catalog/Employees",
      "dataURI": "/rest/Employees"
    }
  ]
}
```

Cela signifie que le datastore contient le dataclass Employees. Vous pouvez voir les attributs de la dataclass en tapant :

```
/rest/$catalog/Employees
```

Si vous souhaitez obtenir toutes les entités de la dataclass Employee, vous pouvez écrire :

```
/rest/Employees
```

Réponse :

```
{
  "__entityModel": "Employees",
  "__GlobalStamp": 0,
  "__COUNT": 3,
  "__FIRST": 0,
  "__ENTITIES": [
    {
      "__KEY": "1",
      "__TIMESTAMP": "2020-01-07T17:07:52.467Z",
      "__STAMP": 2,
      "ID": 1,
      "Lastname": "Brown",
      "Firstname": "Michael",
      "Salary": 25000
    },
    {
      "__KEY": "2",
      "__TIMESTAMP": "2020-01-07T17:08:14.387Z",
      "__STAMP": 2,
      "ID": 2,
      "Lastname": "Jones",
      "Firstname": "Maryanne",
      "Salary": 35000
    },
    {
      "__KEY": "3",
      "__TIMESTAMP": "2020-01-07T17:08:34.844Z",
      "__STAMP": 2,
      "ID": 3,
      "Lastname": "Smithers",
      "Firstname": "Jack",
      "Salary": 41000
    }
  ],
  "__SENT": 3
}
```

Il existe plusieurs possibilités pour filtrer les données à recevoir. Par exemple, pour obtenir uniquement la valeur de l'attribut "Lastname" de la 2ème entité, vous pouvez simplement écrire :

```
/rest/Employees(2)/Lastname
```

Réponse :

```
{
  "__entityModel": "Employees",
  "__KEY": "2",
  "__TIMESTAMP": "2020-01-07T17:08:14.387Z",
  "__STAMP": 2,
  "Lastname": "Jones"
}
```

L'[API REST](#) de 4D fournit plusieurs commandes pour interagir avec les applications 4D.

Configuration du serveur

À l'aide de requêtes HTTP standard, le serveur 4D REST permet aux applications externes d'accéder directement aux données de votre application, c'est-à-dire de récupérer des informations sur les dataclass de votre projet, de manipuler des données, de vous connecter à votre application Web et bien plus encore.

Pour commencer à utiliser les fonctionnalités REST, vous devez démarrer et configurer le serveur 4D REST.

- Sur 4D Server, l'ouverture d'une session REST nécessite une licence client 4D disponible.
- On 4D single-user, you can open up to three REST sessions for testing purposes.
- You need to manage the [session](#) for your requesting application.

Démarrage du serveur REST

Pour des raisons de sécurité, par défaut, 4D ne répond pas aux requêtes REST. Si vous souhaitez démarrer le serveur REST, cochez l'option Exposer en tant que serveur REST dans la page Web > Web Features des paramètres de la structure afin que les requêtes REST soient traitées.

The screenshot shows the 'Structure Settings' window for a project named 'Git'. The top navigation bar has tabs for General, Interface, Compiler, Database, Moving, Backup, Client-server, Web (which is selected and highlighted in grey), SQL, PHP, Security, and Compatibility. Below this is a sub-navigation bar with tabs for Configuration, Options (I), Options (II), Log (type), Log (backup), Web Services, and Web features (which is selected). The main content area has two sections: 'Publishing' and 'Access'. In the 'Publishing' section, there is a note: 'NOTE: This service is only active on 4D Developer Professional and 4D Server.' followed by a checked checkbox labeled 'Expose as REST server'. In the 'Access' section, there is a note: 'NOTE: This setting is only taken into account when the 4D password access system is activated (the Designer has been assigned a password) and the database method "On REST Authentication" does not exist.' followed by a dropdown menu labeled 'Read/Write: <Anyone>'.

Les services REST utilisent le serveur HTTP 4D; vous devez donc vous assurer que le serveur web 4D est lancé.

Le message d'avertissement "Attention, vérifiez les privilèges d'accès" s'affiche lorsque vous cochez cette option, pour attirer votre attention sur le fait que lorsque les services REST sont activés, l'accès par défaut aux objets de base de données est gratuit tant que les accès REST n'ont pas été configurés.

You must restart the 4D application for your changes to take effect.

Configuration de l'accès REST

Par défaut, les accès REST sont ouverts à tous les utilisateurs, ce qui n'est évidemment pas recommandé pour des raisons de sécurité et de contrôle de l'utilisation des licences clientes.

Vous pouvez configurer les accès REST de l'une des manières suivantes :

- en attribuant un groupe d'utilisateurs en lecture/écriture aux services REST dans la page "Web > Web Features" des paramètres de structure ;

- saisir d'une méthode base `On REST Authentication` pour intercepter et gérer chaque demande REST initiale.

Vous ne pouvez pas utiliser les deux fonctionnalités simultanément. Une fois qu'une méthode base `On REST Authentication` a été définie, 4D lui donne entièrement le contrôle des requêtes REST : tout paramètre effectué à l'aide du menu "Lecture/Ecriture" de la page de Web > Fonctionnalités Web des paramètres de la structure est ignoré.

Utilisation des Paramètres de la Structure

Le menu Lecture/Écriture de la page "Web > Web Features" des paramètres de la structure indique un groupe d'utilisateurs 4D autorisé à établir le lien vers l'application 4D à l'aide des requêtes REST.

Par défaut, le menu affiche `<Anyone>`, ce qui signifie que les accès REST sont ouverts à tous les utilisateurs. Une fois que vous avez spécifié un groupe, seul un compte utilisateur 4D appartenant à ce groupe peut être utilisé pour [accéder à 4D via une requête REST](#). Si un compte utilisé n'appartient pas à ce groupe, 4D renvoie une erreur d'authentification à l'expéditeur de la requête.

Pour que ce paramètre prenne effet, la méthode base `On REST Authentication` ne doit pas être définie. S'il existe, 4D ignore les paramètres d'accès définis dans les propriétés de la structure.

Méthode base On REST Authentication

La méthode base `On REST Authentication` vous permet de contrôler de manière personnalisée l'ouverture des sessions REST sur 4D. Cette méthode base est automatiquement appelée lorsqu'une nouvelle session est ouverte à l'aide d'une requête REST. Lorsqu'une [requête d'ouverture de session REST](#) est reçue, les identifiants de connexion sont fournis dans l'en-tête de la requête. La méthode base `On REST Authentication` est appelée afin de vous permettre d'évaluer ces identifiants. Pour obtenir plus d'informations, veuillez vous reporter à la [documentation](#) de la méthode base `On REST Authentication`. Pour obtenir plus d'informations, veuillez vous reporter à la [documentation](#) de la méthode base `On REST Authentication`.

Exposer les tables et les champs

Une fois les services REST sont activés dans l'application 4D, une session REST peut par défaut accéder à toutes les tables et à tous les champs de la base de données 4D via l'[interface du datastore](#). Ainsi, elle peut utiliser leurs données. Par exemple, si votre base de données contient une table [Employee], il est possible d'écrire :

```
http://127.0.0.1:8044/rest/Employee/?$filter="salary>10000"
```

Cette requête retournera tous les employés dont le champ "salary" est supérieur à 10 000.

Les tables et/ou champs 4D dont l'attribut est "invisible" sont également exposés par défaut dans REST.

Si vous souhaitez personnaliser les objets du datastore accessibles via REST, vous devez désactiver l'exposition de chaque table et/ou champ que vous souhaitez masquer. Lorsqu'une requête REST tente d'accéder à une ressource non autorisée, 4D retourne une erreur.

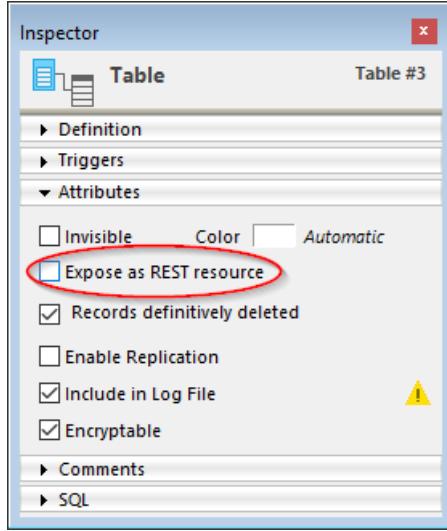
Exposer des tables

Par défaut, toutes les tables sont exposées dans REST.

Pour des raisons de sécurité, vous pouvez choisir d'exposer uniquement certaines tables du datastore aux appels REST. Par exemple, si vous avez créé une table [Users] stockant les noms d'utilisateur et les mots de passe, il serait préférable de ne pas l'exposer.

Pour supprimer l'exposition REST d'une table :

1. Affichez l'Inspecteur de table dans l'Editeur de structure et sélectionnez la table à modifier.
2. Décochez l'option Exposer en tant que ressource REST :



Procédez ainsi pour chaque table dont l'exposition doit être modifiée.

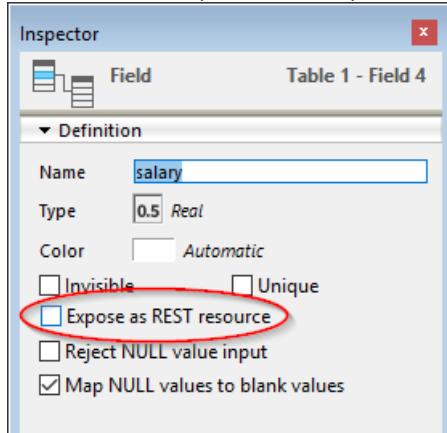
Exposer des champs

Par défaut, tous les champs d'une base 4D sont exposés dans REST.

Vous pouvez choisir d'exposer certains champs de vos tables à REST. Par exemple, si vous ne souhaitez pas exposer le champ [Employees]Salary.

Pour supprimer l'exposition REST d'un champ :

1. Affichez l'Inspecteur de champ dans l'Editeur de structure et sélectionnez le champ à modifier.
2. Décochez la case Exposer en tant que ressource REST pour le champ.



Répétez cette opération pour chaque champ dont l'exposition doit être modifiée.

Pour qu'un champ soit accessible via REST, la table parente doit l'être également. Si la table parente n'est pas exposée, aucun de ses champs ne le sera, quel que soit leur statut.

Sessions et utilisateurs

REST requests can benefit from [web user sessions](#), providing extra features such as multiple requests handling, data sharing between the web client processes, and user privileges.

La première étape à suivre pour ouvrir une session REST sur le serveur 4D, consiste à authentifier l'utilisateur qui envoie la requête.

Authentification des utilisateurs

You log in a user to your application by calling `$directory/login` in a POST request including the user's name and password in the header. This request calls the `On REST Authentication` database method (if it exists), where you can check the user's credentials (see example below).

Ouverture des sessions

Lorsque les [sessions évolutives sont activées](#) (recommandé), si la méthode base `On REST Authentication` retourne `true`, une session utilisateur est alors automatiquement ouverte et vous pouvez la gérer via l'objet `Session` et l'[API Session](#). Subsequent REST requests will reuse the same session cookie.

If the `On REST Authentication` database method has not been defined, a `guest` session is opened.

Exemple

In this example, the user enters their email and password in an html page that requests `$directory/login` in a POST (it is recommended to use an HTTPS connection to send the html page). The `On REST Authentication` database method is called to validate the credentials and to set the session.

The HTML login page:

Email:

Password:

```

<html><body bgcolor="#ffffff">

<div id="demo">
  <FORM name="myForm">
    Email: <INPUT TYPE=TEXT NAME=userId VALUE=""><BR>
    Password: <INPUT TYPE=TEXT NAME=password VALUE=""><BR>
    <button type="button" onclick="onClick()">
      Login
    </button>
  </div id="authenticationFailed" style="visibility:hidden;">Authentication failed</div>
</FORM>
</div>

<script>
function sendData(data) {
  var XHR = new XMLHttpRequest();

  XHR.onreadystatechange = function() {
    if (this.status == 200) {
      window.location = "authenticationOK.shtml";
    }
    else {
      document.getElementById("authenticationFailed").style.visibility = "visible";
    }
  };
}

XHR.open('POST', 'http://127.0.0.1:8044/rest/$directory/login'); //rest server address

XHR.setRequestHeader('username-4D', data.userId);
XHR.setRequestHeader('password-4D', data.password);
XHR.setRequestHeader('session-4D-length', data.timeout);

XHR.send();
};

function onClick()
{
sendData({userId:document.forms['myForm'].elements['userId'].value , password:document.forms['myForm'].e
}
</script></body></html>

```

When the login page is sent to the server, the `On REST Authentication` database method is called:

```

//On REST Authentication

#DECLARE($userId : Text; $password : Text) -> $Accepted : Boolean
var $sales : cs.SalesPersonsEntity

$Accepted:=False

//A '/rest' URL has been called with headers username-4D and password-4D
If ($userId#"")
  $sales:=ds.SalesPersons.query("email = :1"; $userId).first()
  If ($sales#Null)
    If (Verify password hash($password; $sales.password))
      fillSession($sales)
      $Accepted:=True
    End if
  End if
End if

```

As soon as it has been called and returned `True`, the `On REST Authentication` database method is no longer called in the session.

The `fillSession` project method initializes the user session, for example:

```
#DECLARE($sales : cs.SalesPersonsEntity)
var $info : Object

$info:=New object()
$info.userName:=$sales.firstname" "+$sales.lastname

Session.setPrivileges($info)

Use (Session.storage)
If (Session.storage.myTop3=NULL)
    Session.storage.myTop3:=$sales.customers.orderBy("totalPurchase desc").slice(0; 3)
End if
End use
```

Obtenir des informations du serveur

Vous pouvez obtenir plusieurs informations du serveur REST :

- les datastores exposées et leurs attributs
- le contenu du cache du serveur REST, y compris les sessions utilisateur.

Catalogue

Utilisez les paramètres `$catalog`, `$catalog/{dataClass}`, ou `$catalog/$all` pour obtenir la liste [des dataclasses exposées et leurs attributs](#).

Pour obtenir la collection de toutes les dataclass exposées avec leurs attributs :

```
GET /rest/$catalog/$all
```

Informations sur le cache

Utilisez le paramètre `$info` pour obtenir des informations sur les sélections d'entités stockées dans le cache du 4D Server et sur l'exécution des sessions utilisateur.

queryPath et queryPlan

Les sélections d'entité générées par les requêtes peuvent avoir les deux propriétés suivantes : `queryPlan` et `queryPath`. Pour calculer et retourner ces propriétés, il vous suffit d'ajouter `$queryPlan` et/ou `$queryPath` dans la demande REST.

Par exemple :

```
GET /rest/People/$filter="employer.name=acme AND lastName=Jones"&$queryplan=true&$querypath=true
```

Ces propriétés sont des objets contenant des informations sur la façon dont le serveur exécute les requêtes composites en interne via des dataclass et des relations :

- `queryPlan` : objet contenant la description détaillée de la requête juste avant son exécution (c'est-à-dire la requête planifiée).
- `queryPath` : objet contenant la description détaillée de la requête telle qu'elle a été réellement effectuée.

Les informations enregistrées incluent le type de requête (indexé et séquentiel) et chaque sous-requête nécessaire ainsi que les opérateurs de conjonction. Les chemins d'accès des requêtes contiennent également le nombre d'entités identifiées et la durée d'exécution de chaque critère de recherche. Il peut être utile d'analyser ces informations lors du développement de votre application. Généralement, la description du plan de requête et son chemin d'accès sont identiques mais ils peuvent différer car 4D peut intégrer des optimisations dynamiques lorsqu'une requête est exécutée, afin d'améliorer les performances. Par exemple, le moteur 4D peut convertir dynamiquement une requête indexée en requête séquentielle s'il estime que ce processus est plus rapide. Ce cas particulier peut se produire lorsque le nombre d'entités recherchées est faible.

Manipulation des données

Tous les attributs, dataclasses exposés et toutes les fonctions sont accessibles via REST. Les noms de dataclass, d'attributs et de fonctions sont sensibles à la casse; contrairement aux données des requêtes.

Rechercher des données

Pour rechercher directement des données, vous pouvez utiliser la fonction `$filter`. Par exemple, pour trouver une personne nommée "Smith", vous pouvez écrire :

```
http://127.0.0.1:8081/rest/Person/?$filter="lastName=Smith"
```

Adding, modifying, and deleting entities

Avec l'API REST, vous pouvez effectuer toutes les manipulations de données souhaitées dans 4D.

Pour ajouter et modifier des entités, vous pouvez appeler `$method=update`. Si vous souhaitez supprimer une ou plusieurs entités, vous pouvez utiliser `$method=delete`.

Outre la récupération d'un attribut dans une dataclass à l'aide de `{dataClass}({key})`, vous pouvez également écrire une `class function` qui retourne une entity selection (ou une collection).

Avant de retourner la sélection, vous pouvez également la trier en utilisant `$orderby` un ou plusieurs attributs (même les attributs de relation).

Navigating data

Ajoutez le `$skip` (pour définir avec quelle entité commencer) et `$top/$limit` (pour définir le nombre d'entités à retourner) des requêtes REST à vos requêtes ou entity selections pour parcourir la collection d'entités.

Creating and managing entity set

Un ensemble d'entités (également appelé *entity set* ou <0>entity selection</0>) est une collection d'entités obtenue via une requête REST stockée dans le cache de 4D Server. L'utilisation d'un entity set vous empêche de lancer continuellement des requêtes à votre application pour obtenir les mêmes résultats. L'accès à un entity set est beaucoup plus rapide et peut améliorer la vitesse de votre application.

Pour créer un entity set, appelez `$method=entityset` dans votre requête REST. Par mesure de sécurité, vous pouvez également utiliser `$savedfilter` et/ou `$savedorderby` lorsque vousappelez `$filter` et/ou `$orderby` afin que l'entité définie puisse être rapidement récupérée avec le même ID que précédemment, dans le cas où elle expireait ou serait supprimée du serveur.

Pour accéder à l'entity set, vous devez utiliser `$entityset/{entitySetID}`, par exemple :

```
/rest/People/$entityset/0AF4679A5C394746BFEB68D2162A19FF
```

Par défaut, un entity set est stocké pendant deux heures; cependant, vous pouvez modifier le timeout en passant une nouvelle valeur à `$timeout`. Le timeout est continuellement réinitialisé à la valeur définie (soit par défaut soit à celle que vous définissez) chaque fois que vous l'utilisez.

Si vous souhaitez supprimer un entity set du cache de 4D Server, vous pouvez utiliser `$method=release`.

Si vous modifiez l'un des attributs de l'entité dans l'entity set, les valeurs seront mises à jour. Toutefois, si vous modifiez une valeur qui faisait partie de la requête exécutée pour créer l'entity set, elle ne sera pas supprimée de l'entity set même si elle ne correspond plus aux critères de recherche. Bien entendu, les entités que vous supprimez ne feront plus partie de l'entity set.

Si l'ensemble d'entités ne se trouve plus dans le cache de 4D Server, il sera recréé avec un nouveau timeout de 10

minutes. L'ensemble d'entités sera actualisé (certaines entités peuvent être incluses tandis que d'autres peuvent être supprimées) depuis la dernière fois qu'il a été créé, s'il n'existe plus avant de le recréer.

Using `$entityset/{entitySetID}?$logicOperator... &$otherCollection`, you can combine two entity sets that you previously created. Vous pouvez combiner les résultats dans les deux, retourner uniquement ce qui est commun aux deux, ou renvoyer ce qui n'est pas commun aux deux.

Une nouvelle sélection d'entités est renvoyée; vous pouvez néanmoins créer un nouvel ensemble d'entités en appelant `$method=entityset` à la fin de la requête REST.

Calculating data

En utilisant `$compute`, vous pouvez calculer la moyenne, le nombre, le min, le max ou la somme pour un attribut spécifique d'une dataclass. Vous pouvez également calculer toutes les valeurs avec le mot clé `$all`.

Par exemple, pour obtenir le salaire le plus élevé :

```
/rest/Employee/salary/?$compute=max
```

Pour calculer toutes les valeurs et retourner un objet JSON :

```
/rest/Employee/salary/?$compute=$all
```

Appeler les fonctions de classe du modèle de données

Vous pouvez appeler des [fonctions de classe utilisateurs](#) ORDA du modèle de données via des requêtes POST, afin de pouvoir bénéficier de l'API de l'application ciblée. Par exemple, si vous avez défini une fonction `getCity()` dans la dataclass City, vous pouvez l'appeler à l'aide de la requête suivante :

```
/rest/City/getCity
```

avec des données contenues dans le corps de la requête : `["Paris"]`

Les appels aux méthodes projet 4D exposées en tant que service REST sont toujours pris en charge mais sont obsolètes.

Selecting Attributes to get

Vous pouvez toujours définir les attributs à retourner dans la réponse REST après une requête initiale en passant leur chemin d'accès dans la requête (par exemple, `Company(1)/name,revenues/`)

Vous pouvez appliquer ce filtre comme suit :

Object	Syntaxe	Exemple
Dataclass	<code>{dataClass}/{att1,att2...}</code>	<code>/People/firstName,lastName</code>
Collection d'entités	<code>{dataClass}/{att1,att2...}/?\$filter="{filter}"</code>	<code>/People/firstName,lastName/?\$filter="lastName=</code>
Entité spécifique	<code>{dataClass}({ID})/{att1,att2...}</code>	<code>/People(1)/firstName,lastName</code>
	<code>{dataClass}:{attribute}(value)/{att1,att2...}/</code>	<code>/People:firstName(Larry)/firstName,lastName/</code>
Entity selection	<code>{dataClass}/{att1,att2...}/\$entityset/{entitySetID}</code>	<code>/People/firstName/\$entityset/528BF90F1089491</code>

Les attributs doivent être délimités par une virgule, c'est-à-dire `/Employee/firstName,lastName,salary`. Des attributs de stockage ou des attributs relationnels peuvent être transmis.

Exemples

Voici quelques exemples vous permettant d'indiquer les attributs à retourner en fonction de la méthode employée pour récupérer les entités.

Vous pouvez appliquer cette méthode à :

- Dataclass (tout ou une collection d'entités dans une dataclass)
- Entités spécifiques
- Entity sets

Exemple avec une dataclass

Les requêtes suivantes retournent uniquement le prénom et le nom de la dataclass People (soit la dataclass entière, soit une sélection d'entités basée sur la recherche définie dans `$filter`).

```
GET /rest/People/firstName,lastName/
```

Résultat :

```
{
  __entityModel: "People",
  __COUNT: 4,
  __SENT: 4,
  __FIRST: 0,
  __ENTITIES: [
    {
      __KEY: "1",
      __STAMP: 1,
      firstName: "John",
      lastName: "Smith"
    },
    {
      __KEY: "2",
      __STAMP: 2,
      firstName: "Susan",
      lastName: "Leary"
    },
    {
      __KEY: "3",
      __STAMP: 2,
      firstName: "Pete",
      lastName: "Marley"
    },
    {
      __KEY: "4",
      __STAMP: 1,
      firstName: "Beth",
      lastName: "Adams"
    }
  ]
}
```

```
GET /rest/People/firstName,lastName/?$filter="lastName='A@'" /
```

Résultat :

```
{
  __entityModel: "People",
  __COUNT: 1,
  __SENT: 1,
  __FIRST: 0,
  __ENTITIES: [
    {
      __KEY: "4",
      __STAMP: 4,
      firstName: "Beth",
      lastName: "Adams"
    }
  ]
}
```

Exemple d'entité

La requête suivante retourne uniquement les attributs de prénom et nom à partir d'une entité spécifique de la dataclass People :

```
GET /rest/People(3)/firstName,lastName/
```

Résultat :

```
{
  __entityModel: "People",
  __KEY: "3",
  __STAMP: 2,
  firstName: "Pete",
  lastName: "Marley"
}
```

```
GET /rest/People(3)/
```

Résultat :

```
{
  __entityModel: "People",
  __KEY: "3",
  __STAMP: 2,
  ID: 3,
  firstName: "Pete",
  lastName: "Marley",
  salary: 30000,
  employer: {
    __deferred: {
      uri: "http://127.0.0.1:8081/rest/Company(3)",
      __KEY: "3"
    }
  },
  fullName: "Pete Marley",
  employerName: "microsoft"
}
```

Exemple d'ensemble d'entités

Once you have [created an entity set](#), you can filter the information in it by defining which attributes to return:

```
GET /rest/People/firstName,employer.name/$entityset/BDCD8AABE13144118A4CF8641D5883F5?$expand=employer
```

Affichage d'un attribut d'image

Si vous souhaitez afficher intégralement un attribut d'image, saisissez ce qui suit :

```
GET /rest/Employee(1)/photo?$imageformat=best&$version=1&$expand=photo
```

Pour plus d'informations sur les formats d'image, reportez-vous à `$imageformat`. Pour plus d'informations sur le paramètre de version, reportez-vous à `$version`.

Enregistrement d'un attribut BLOB sur le disque

Si vous souhaitez enregistrer un BLOB stocké dans votre dataclass, vous pouvez écrire ce qui suit :

```
GET /rest/Company(11)/blobAtt?$binary=true&$expand=blobAtt
```

Récupérer une seule entité

Vous pouvez utiliser la syntaxe `{dataClass}:{attribute}(value)` lorsque vous souhaitez récupérer une seule entité. C'est particulièrement utile lorsque vous souhaitez lancer une recherche associée qui n'est pas créée sur la clé primaire de la dataclass. Par exemple, vous pouvez écrire :

```
GET /rest/Company:companyCode("Acme001")
```

Appeler des fonctions de classe ORDA

Vous pouvez appeler les [fonctions de classe de modèles de données](#) définies pour le modèle de données ORDA via vos requêtes REST, afin de bénéficier de l'API de l'application 4D ciblée.

Les fonctions sont simplement appelées dans les requêtes POST sur l'interface ORDA appropriée, sans (). Par exemple, si vous avez défini une fonction `getCity()` dans la dataclass City, vous pouvez l'appeler à l'aide de la requête suivante :

```
/rest/City/getCity
```

avec des données contenues dans le corps de la requête POST : `["Paris"]`

Dans le langage 4D, cet appel équivaut à :

```
$city:=ds.City.getCity("Aguada")
```

Seules les fonctions contenant le mot-clé `exposed` peuvent être directement appelées à partir de requêtes REST. See [Exposed vs non-exposed functions](#) section.

Appeler des fonctions

Les fonctions doivent toujours être appelées à l'aide des requêtes POST (une requête GET recevra une erreur).

Les fonctions sont appelées sur l'objet correspondant au datastore du serveur.

Fonction de classe	Syntaxe
<code>datastore class</code>	<code>/rest/\$catalog/DataStoreClassFunction</code>
<code>dataclass class</code>	<code>/rest/{dataClass}/DataClassClassFunction</code>
<code>entitySelection class</code>	<code>/rest/{dataClass}/EntitySelectionClassFunction</code>
	<code>/rest/{dataClass}/EntitySelectionClassFunction/\$entityset/entitySetNumber</code>
	<code>/rest/{dataClass}/EntitySelectionClassFunction/\$filter</code>
	<code>/rest/{dataClass}/EntitySelectionClassFunction/\$orderby</code>
<code>entity class</code>	<code>/rest/{dataClass}(key)/EntityClassFunction/</code>

`/rest/{dataClass}/Function` peut être utilisé pour appeler une fonction de dataclass ou de sélection d'entité (`/rest/{dataClass}` retourne toutes les entités de la DataClass en tant que sélection d'entité).

La fonction est d'abord recherchée dans la classe de sélection d'entité. Si elle n'est pas trouvée, elle est recherchée dans la dataclass. En d'autres termes, si une fonction portant le même nom est définie à la fois dans la classe DataClass et la classe EntitySelection, la fonction de classe de dataclass ne sera jamais exécutée.

All 4D code called from REST requests must be thread-safe if the project runs in compiled mode, because the REST Server always uses preemptive processes in this case (the [Use preemptive process setting value](#) is ignored by the REST Server).

Paramètres

Vous pouvez envoyer des paramètres aux fonctions définies dans les classes utilisateurs ORDA. Côté serveur, ils seront

reçus dans les fonctions de classe dans les paramètres normaux \$1, \$2, etc.

Les règles suivantes s'appliquent :

- Les paramètres doivent être passés dans le corps de la requête POST
- Les paramètres doivent être inclus dans une collection (format JSON)
- Tous les types de données scalaires pris en charge dans les collections JSON peuvent être passés en tant que paramètres.
- La sélection d'entité et l'entité peuvent être passées en tant que paramètres. L'objet JSON doit contenir des attributs spécifiques utilisés par le serveur REST pour affecter des données aux objets ORDA correspondants : __DATACLASS, __ENTITY, __ENTITIES, __DATASET.

Voir [cet exemple](#) et [cet exemple](#).

Paramètre de valeur scalaire

Le(s) paramètre(s) doivent simplement être incluse dans une collection définie dans le corps. Par exemple, avec une fonction de dataclass `getCities()` qui reçoit des paramètres de type texte : `/rest/City/getCities`

Paramètres dans le corps : `["Aguada","Paris"]`

Tous les types de données JSON sont pris en charge dans les paramètres, y compris les pointeurs JSON. Les dates peuvent être passées sous forme de chaînes au format de date ISO 8601 (par exemple, "2020-08-22T22:00:00Z").

Paramètre d'entité

Les entités passées en paramètres sont référencées sur le serveur via leur clé (c'est-à-dire la propriété __KEY). Si le paramètre clé est omis dans une requête, une nouvelle entité est chargée en mémoire du serveur. Vous pouvez également transmettre des valeurs pour tous les attributs de l'entité. Ces valeurs seront automatiquement utilisées pour l'entité traitée sur le serveur.

Si la requête envoie des valeurs d'attribut modifiées pour une entité existante sur le serveur, la fonction de modèle de données ORDA appelée sera automatiquement exécutée sur le serveur avec des valeurs modifiées. Cette fonctionnalité vous permet, par exemple, de vérifier le résultat d'une opération sur une entité, après avoir appliqué toutes les règles métier, depuis l'application cliente. Vous pouvez alors décider de sauvegarder ou non l'entité sur le serveur.

Propriétés	Type	Description
Attributs de l'entité	mixte	Optionnelle - Valeurs à modifier
__DATACLASS	Chaine	Obligatoire - Indique la Dataclass de l'entité
__ENTITY	Booléen	Obligatoire - Vrai pour indiquer au serveur que le paramètre est une entité
__KEY	mixte (type identique à celui de la clé primaire)	Optionnel - clé primaire de l'entité

- Si __KEY n'est pas fourni, une nouvelle entité est créée sur le serveur avec les attributs donnés.
- Si __KEY est fourni, l'entité correspondant à __KEY est chargée sur le serveur avec les attributs donnés

Voir les exemples de [création](#) ou de [mise à jour](#) des entités.

Paramètre d'entité associé

Mêmes propriétés que pour un [paramètre d'entité](#). De plus, l'entité associée doit exister et est référencée par __KEY, qui contient sa clé primaire.

Reportez-vous aux exemples de [création](#) ou de [mise à jour](#) des entités avec des entités associées.

Paramètre de sélection d'entité

La sélection d'entité doit avoir été définie au préalable à l'aide de `$method=entityset`.

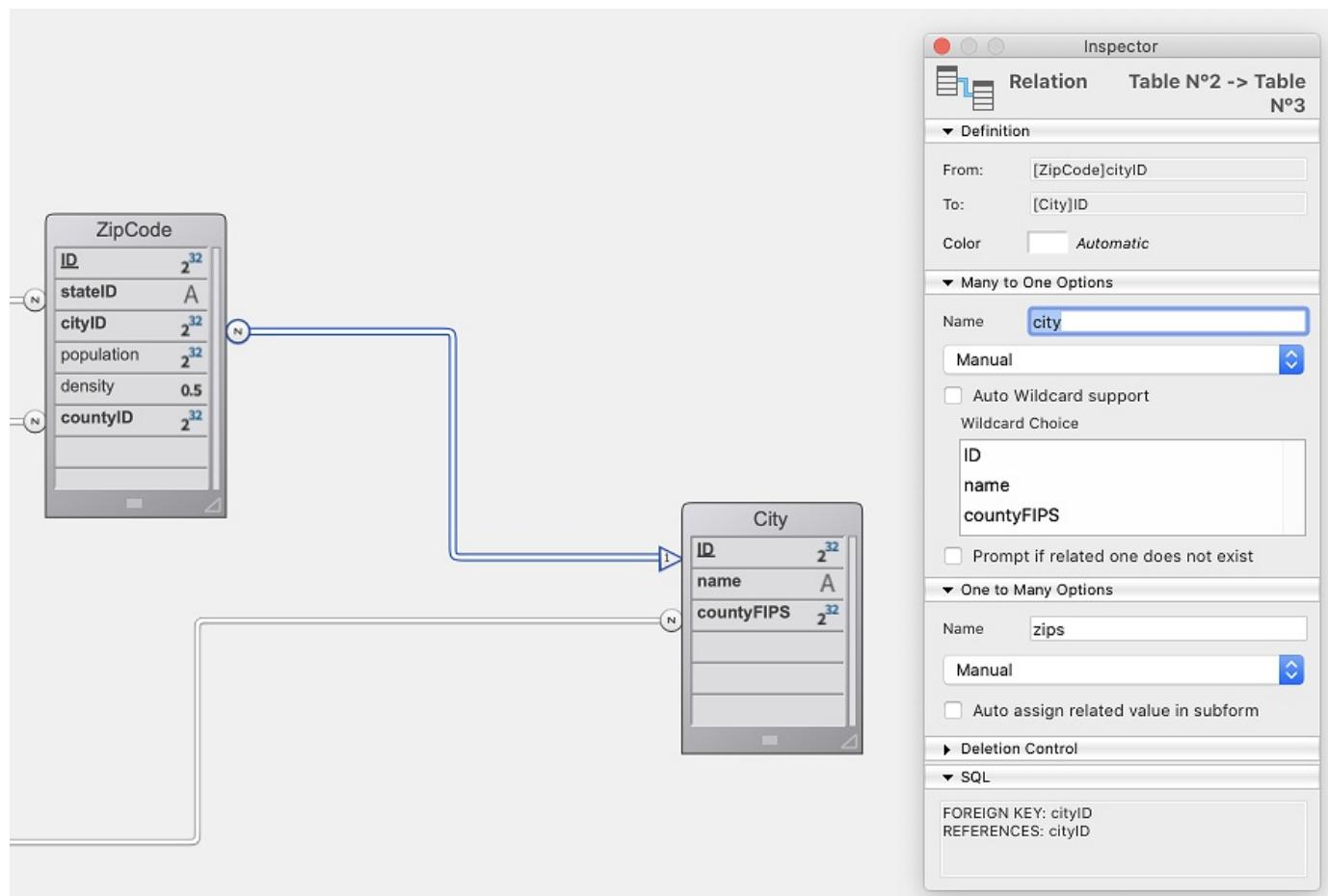
Si la requête envoie une sélection d'entité modifiée au serveur, la fonction de modèle de données ORDA appelée sera automatiquement exécutée sur le serveur avec la sélection d'entité modifiée.

Propriétés	Type	Description
Attributs de l'entité	mixte	Optionnelle - Valeurs à modifier
<code>__DATASET</code>	Chaine	Obligatoire - entitySetID (UUID) de la sélection d'entité
<code>__ENTITIES</code>	Booléen	Obligatoire - Vrai pour indiquer au serveur que le paramètre est une sélection d'entité

Reportez-vous aux exemples de [réception d'une sélection d'entité](#).

Exemples de requêtes

Cette base de données est exposée comme un datastore distant sur localhost (port 8111) :



Utiliser une fonction de classe de datastore

La classe de `DataStore` `US_Cities` fournit une API :

```
// DataStore class

Class extends DataStoreImplementation

exposed Function getName()
    $0:="US cities and zip codes manager"
```

Vous pouvez lancer cette requête :

```
POST 127.0.0.1:8111/rest/$catalog/getName
```

Résultat

```
{  
  "result": "US cities and zip codes manager"  
}
```

Utiliser une fonction de classe de dataclass

La classe de Dataclass `City` fournit une API qui retourne une entité de ville à partir du nom passé en paramètre :

```
// City class  
  
Class extends DataClass  
  
exposed Function getCity()  
  var $0 : cs.CityEntity  
  var $1,$nameParam : text  
  $nameParam:=$1  
  $0:=This.query("name = :1";$nameParam).first()
```

Vous pouvez lancer cette requête :

```
POST 127.0.0.1:8111/rest/City/getCity
```

Requête : ["Paris"]

Résultat

Le résultat est une entité :

```
{  
  "__entityModel": "City",  
  "__DATACLASS": "City",  
  "__KEY": "1",  
  "__TIMESTAMP": "2020-03-09T08:03:19.923Z",  
  "__STAMP": 1,  
  "ID": 1,  
  "name": "Paris",  
  "countyFIPS": 72003,  
  "county": {  
    "__deferred": {  
      "uri": "/rest/County(72003)",  
      "__KEY": "72003"  
    }  
  },  
  "zips": {  
    "__deferred": {  
      "uri": "/rest/City(1)/zips?$expand=zips"  
    }  
  }  
}
```

Utiliser une fonction de classe d'une entité

La classe d'entité `CityEntity` fournit une API :

```
// CityEntity class

Class extends Entity

exposed Function getPopulation()
$0:=This.zips.sum("population")
```

Vous pouvez lancer cette requête :

POST 127.0.0.1:8111/rest/City(2)/getPopulation

Résultat

```
{
    "result": 48814
}
```

Utiliser une fonction de classe d'une sélection d'entité

La classe de sélection d'entité `CityEntity` fournit une API :

```
// CitySelection class

Class extends EntitySelection

exposed Function getPopulation()
$0:=This.zips.sum("population")
```

Vous pouvez lancer cette requête :

POST 127.0.0.1:8111/rest/City/getPopulation/?\$filter="ID<3"

Résultat

```
{
    "result": 87256
}
```

Utiliser une fonction de classe de sélection d'entité et un ensemble d'entité

La classe `StudentsSelection` a une fonction `getAgeAverage` :

```
// StudentsSelection Class

Class extends EntitySelection

exposed Function getAgeAverage
C_LONGINT($sum;$0)
C_OBJECT($s)

$sum:=0
For each ($s;This)
    $sum:=$sum+$s.age()
End for each
$0:=$sum/This.length
```

Une fois que vous avez créé un ensemble d'entité, vous pouvez lancer cette requête :

```
POST 127.0.0.1:8044/rest/Students/getAgeAverage/$entityset/17E83633FFB54ECDBF947E5C620BB532
```

Résultat

```
{  
    "result": 34  
}
```

Utiliser une fonction de classe de sélection d'entité et un "orderBy"

La classe `StudentsSelection` a une fonction `getLastSummary` :

```
// StudentsSelection Class  
  
Class extends EntitySelection  
  
exposed Function getLastSummary  
    C_TEXT($0)  
    C_OBJECT($last)  
  
    $last:=This.last()  
    $0:=$last.firstname + " - +" + $last.lastname + " is ... " + String($last.age())
```

Vous pouvez lancer cette requête :

```
POST 127.0.0.1:8044/rest/Students/getLastSummary/$entityset/?$filter=lastname=b@&$orderby="lastname"
```

Résultat

```
{  
    "result": "Wilbert - Bull is ... 21"  
}
```

Utiliser une entité à créer sur le serveur

La classe de Dataclass `Students` possède la fonction `pushData()` qui reçoit une entité contenant les données du client. La méthode `checkData()` effectue quelques contrôles. Si elles sont valides, l'entité est sauvegardée et retournée.

```

// Students Class

Class extends DataClass

exposed Function pushData
    var $1, $entity, $status, $0 : Object

    $entity:=$1

    $status:=checkData($entity) // $status est un objet avec une propriété booléenne "success"

    $0:=$status

    If ($status.success)
        $status:=$entity.save()
        If ($status.success)
            $0:=$entity
    End if
End if

```

Lancez cette requête :

POST <http://127.0.0.1:8044/rest/Students/pushData>

Corps de la requête :

```
[{
    "__DATACLASS": "Students",
    "__ENTITY": true,
    "firstname": "Ann",
    "lastname": "Brown"
}]
```

Si aucune `_KEY` n'est donnée, une nouvelle entité Students est chargée sur le serveur avec les attributs du client. Parce que la fonction `pushData()` exécute une action `save()`, la nouvelle entité est créée.

Résultat

```
{
    "__entityModel": "Students",
    "__DATACLASS": "Students",
    "__KEY": "55",
    "__TIMESTAMP": "2020-06-16T10:54:41.805Z",
    "__STAMP": 1,
    "ID": 55,
    "firstname": "Ann",
    "lastname": "BROWN",
    "schoolID": null,
    "school": null
}
```

Utiliser une entité à mettre à jour sur le serveur

Description semblable à la précédente, avec l'attribut `_KEY`

Lancez cette requête :

POST: <http://127.0.0.1:8044/rest/Students/pushData>

Corps de la requête :

```
[{  
    "__DATACLASS": "Students",  
    "__ENTITY": true,  
    "lastname": "Brownie",  
    "__KEY": 55  
}]
```

Si aucune `__KEY` n'est donnée, l'entité Students est chargée avec la clé primaire 55 avec la valeur lastname reçue par le client. Parce que la fonction exécute une action `save()`, la nouvelle entité est mise à jour.

Résultat

```
{  
    "__entityModel": "Students",  
    "__DATACLASS": "Students",  
    "__KEY": "55",  
    "__TIMESTAMP": "2020-06-16T11:10:21.679Z",  
    "__STAMP": 3,  
    "ID": 55,  
    "firstname": "Ann",  
    "lastname": "BROWNIE",  
    "schoolID": null,  
    "school": null  
}
```

Créer une entité avec une entité liée

Dans cet exemple, nous créons une nouvelle entité Students avec l'entité Schools ayant la clé primaire 2.

Lancez cette requête :

POST: <http://127.0.0.1:8044/rest/Students/pushData>

Corps de la requête :

```
[{  
    "__DATACLASS": "Students",  
    "__ENTITY": true,  
    "firstname": "John",  
    "lastname": "Smith",  
    "school": {"__KEY": 2}  
}]
```

Résultat

```
{
    "__entityModel": "Students",
    "__DATACLASS": "Students",
    "__KEY": "56",
    "__TIMESTAMP": "2020-06-16T11:16:47.601Z",
    "__STAMP": 1,
    "ID": 56,
    "firstname": "John",
    "lastname": "SMITH",
    "schoolID": 2,
    "school": {
        "__deferred": {
            "uri": "/rest/Schools(2)",
            "__KEY": "2"
        }
    }
}
```

Mettre à jour une entité avec une entité liée

Dans cet exemple, nous associons une école existante à l'entité Students. La classe `StudentsEntity` possède une API :

```
// StudentsEntity class

Class extends Entity

exposed Function putToSchool()
    var $1, $school , $0, $status : Object

        // $1 is a Schools entity
    $school:=$1
        // Associe l'entité reliée "school" à l'entité courante "Students"
    This.school:=$school

    $status:=This.save()

    $0:=$status
```

You run this request, called on a Students entity : POST `http://127.0.0.1:8044/rest/Students(1)/putToSchool` Body of the request:

```
[{
    "__DATACLASS":"Schools",
    "__ENTITY":true,
    "__KEY":2
}]
```

Résultat

```
{
    "result": {
        "success": true
    }
}
```

Recevoir une sélection d'entité comme paramètre

Dans la classe de Dataclass `Students` , la fonction `setFinalExam()` met à jour une sélection d'entité reçue (\$1). Elle met à jour l'attribut `finalExam` avec la valeur reçue (\$2). Elle retourne les clés primaires des entités mises à jour.

```
// Students class

Class extends DataClass

exposed Function setFinalExam()

    var $1, $es, $student, $status : Object
    var $2, $examResult : Text

    var $keys, $0 : Collection

        //Entity selection
$es:=$1

$examResult:=$2

$keys:=New collection()

//Boucle sur la sélection d'entité
For each ($student;$es)
    $student.finalExam:=$examResult
    $status:=$student.save()
    If ($status.success)
        $keys.push($student.ID)
    End if
End for each

$0:=$keys
```

Un ensemble d'entité est d'abord créé avec cette requête :

```
http://127.0.0.1:8044/rest/Students/?$filter="ID<3"&$method=entityset
```

Vous pouvez ensuite exécuter cette requête :

```
POST http://127.0.0.1:8044/rest/Students/setFinalExam
```

Corps de la requête :

```
[
{
  "__ENTITIES":true,
  "__DATASET":"9B9C053A111E4A288E9C1E48965FE671"
},
"Passed"
]
```

Résultat

Les entités ayant les clés primaires sont 1 et 2 ont été mises à jour.

```
{
  "result": [
    1,
    2
  ]
}
```

Utiliser une sélection d'entité mise à jour sur le client

A l'aide de la fonction `getAgeAverage()` définie ci-dessus.

```
var $remoteDS, $newStudent, $students : Object
var $ageAverage : Integer

$remoteDS:=Open datastore(New object("hostname";"127.0.0.1:8044");"students")

// $newStudent est une entité "student" à traiter
$newStudent:=...
$students:=$remoteDS.Students.query("school.name = :1";"Math school")
// Nous avons ajouté une entité à la sélection d'entité $students sur le client
$students.add($newStudent)

// Nous appelons une fonction sur la classe StudentsSelection qui retourne l'âge moyen des étudiants de
// La fonction est utilisée sur le serveur sur la sélection d'entité $students mise à jour, qui inclut la nouvelle entité ajoutée
$ageAverage:=$students.getAgeAverage()
```

A propos des requêtes REST

Les structures suivantes sont prises en charge par les requêtes REST :

URI	Resource (Input)	/? ou &{filter} (Output)
http://{servername}:{port}/rest/	{dataClass}	\$filter, \$attributes, \$skip, \$method=.....
	{dataClass}/\$entityset/{entitySetID}	\$method=...
	{dataClass}({clé})	\$attributes
	{dataClass}:{attribute}(value)	

Si toutes les requêtes REST doivent contenir les paramètres URI et Resource, les filtres d'Output (qui filtrent les données retournées) sont facultatifs.

Comme pour tous les URI, le premier paramètre est délimité par un «?» et tous les paramètres suivants par un «&». Par exemple :

```
GET /rest/Person/?$filter="lastName!=Jones"&$method=entityset&$timeout=600
```

Vous pouvez placer toutes les valeurs entre guillemets en cas de doute. Par exemple, dans notre exemple ci-dessus, nous aurions pu saisir la valeur du nom de famille entre guillemets simples "lastName!='Jones'".

Les paramètres vous permettent de manipuler des données dans des dataclass de votre projet 4D. Outre la récupération de données à l'aide des méthodes HTTP `GET`, vous pouvez également ajouter, mettre à jour et supprimer des entités d'une dataclass à l'aide des méthodes HTTP `POST`.

Si vous souhaitez que les données soient retournées dans un tableau au lieu d'un JSON, utilisez le paramètre `$asArray`.

Statut et réponse REST

À chaque requête REST, le serveur retourne l'état et une réponse (avec ou sans erreur).

Statut de la requête

Avec chaque requête REST, vous obtenez le statut et la réponse. Voici quelques exemples de statuts :

Statut	Description
0	Requête non traitée (le serveur n'est peut-être pas été lancé).
200 OK	Requête traitée sans erreur.
401 Unauthorized	Erreur d'autorisation (vérifiez les autorisations de l'utilisateur).
402 No session	Nombre maximal de sessions ayant été atteint.
404 Not Found	La data n'est pas accessible via REST ou bien l'ensemble d'entités n'existe pas.
500 Internal Server Error	Erreur lors du traitement de la requête REST.

Réponse

La réponse (au format JSON) varie en fonction de la requête.

Si une erreur survient, elle sera envoyée avec la réponse du serveur ou bien ce sera la réponse du serveur.

\$catalog

Le catalogue décrit toutes les dataclass et les attributs disponibles dans le datastore.

Syntaxe

Syntaxe	Exemple	Description
\$catalog	/\$catalog	Retourne une liste des dataclasse de votre projet avec deux URI
\$catalog/\$all	/\$catalog/\$all	Retourne des informations sur toutes les dataclasse de votre projet et leurs attributs
\$catalog/{dataClass}	/\$catalog/Employee	Renvoie des informations sur une dataclass et ses attributs

\$catalog

Retourne une liste de dataclass dans votre projet avec deux URI : une pour accéder aux informations sur sa structure et une pour récupérer les données de la dataclass

Description

Lorsque vous appelez `$catalog`, une liste des dataclass est renvoyée avec deux URI pour chaque dataclass dans le datastore de votre projet.

Seules les dataclass exposées apparaissent dans cette liste pour le datastore de votre projet. Pour plus d'informations, reportez-vous à la section [Exposition des tableaux et des champs](#).

Voici une description des propriétés renvoyées pour chaque dataclass dans le datastore de votre projet :

Propriété	Type	Description
name	Chaine	Nom de la dataclass.
uri	Chaine	Un URI vous permettant d'obtenir des informations sur la dataclass et ses attributs.
dataURI	Chaine	URI vous permettant d'afficher les données dans la dataclass.

Exemple

```
GET /rest/$catalog
```

Résultat :

```
{
  dataClasses: [
    {
      name: "Company",
      uri: "http://127.0.0.1:8081/rest/$catalog/Company",
      dataURI: "http://127.0.0.1:8081/rest/Company"
    },
    {
      name: "Employee",
      uri: "http://127.0.0.1:8081/rest/$catalog/Employee",
      dataURI: "http://127.0.0.1:8081/rest/Employee"
    }
  ]
}
```

\$catalog/\$all

Retourne des informations sur toutes les dataclasse de votre projet et leurs attributs

Description

En appelant `$catalog/$all`, vous pouvez recevoir des informations détaillées sur les attributs de chacune des dataclasses du modèle de votre projet.

Pour plus d'informations sur ce qui est retourné pour chaque dataclass et ses attributs, utilisez `\$catalog/{dataClass}`.

Exemple

```
GET /rest/$catalog/$all
```

Résultat :

```
{
  "dataClasses": [
    {
      "name": "Company",
      "className": "Company",
      "collectionName": "CompanySelection",
      "tableNumber": 2,
      "scope": "public",
      "dataURI": "/rest/Company",
      "attributes": [
        {
          "name": "ID",
          "kind": "storage",
          "fieldPos": 1,
          "scope": "public",
          "indexed": true,
          "type": "long",
          "identifying": true
        },
        {
          "name": "name",
          "kind": "storage",
          "fieldPos": 2,
          "scope": "public",
          "type": "string"
        },
        {
          "name": "revenues",
          "kind": "storage",
          "fieldPos": 3,
          "scope": "public",
          "type": "double"
        }
      ]
    }
  ]
}
```

```
        "kind": "storage",
        "fieldPos": 3,
        "scope": "public",
        "type": "number"
    },
    {
        "name": "staff",
        "kind": "relatedEntities",
        "fieldPos": 4,
        "scope": "public",
        "type": "EmployeeSelection",
        "reversePath": true,
        "path": "employer"
    },
    {
        "name": "url",
        "kind": "storage",
        "scope": "public",
        "type": "string"
    }
],
"key": [
    {
        "name": "ID"
    }
]
},
{
    "name": "Employee",
    "className": "Employee",
    "collectionName": "EmployeeSelection",
    "tableNumber": 1,
    "scope": "public",
    "dataURI": "/rest/Employee",
    "attributes": [
        {
            "name": "ID",
            "kind": "storage",
            "scope": "public",
            "indexed": true,
            "type": "long",
            "identifying": true
        },
        {
            "name": "firstname",
            "kind": "storage",
            "scope": "public",
            "type": "string"
        },
        {
            "name": "lastname",
            "kind": "storage",
            "scope": "public",
            "type": "string"
        },
        {
            "name": "employer",
            "kind": "relatedEntity",
            "scope": "public",
            "type": "Company",
            "path": "Company"
        }
    ],
    "key": [
        {
            "name": "ID"
        }
    ]
}
```

```

        "name": "ID"
    }
}
]
}
}
```

\$catalog/{dataClass}

Renvoie des informations sur une dataclass et ses attributs

Description

L'appel de `$catalog/{dataClass}` pour une dataclass spécifique retournera les informations suivantes sur la dataclass et les attributs qu'elle contient. L'appel de `$catalog/{dataClass}` pour une dataclass spécifique retournera les informations suivantes sur la dataclass et les attributs qu'elle contient.

Les informations que vous récupérez concernent :

- Dataclass
- Attribut(s)
- Méthode(s) le cas échéant
- Clé primaire

DataClass

Les propriétés suivantes sont retournées pour une dataclass exposée :

Propriété	Type	Description
name	Chaine	Nom de la dataclass
collectionName	Chaine	Nom d'une entity selection dans la dataclass
tableNumber	Nombre	Numéro de la table dans la base 4D
scope	Chaine	Étendue de la dataclass (à noter que seules les dataclasses dont l'étendue (scope) est publique sont affichées)
dataURI	Chaine	Un URI aux données de la dataclass

Attribut(s)

Voici les propriétés de chaque attribut exposé qui sont retournées :

Propriété	Type	Description
name	Chaine	Le nom de l'attribut.
kind	Chaine	Type d'attribut (stockage ou relatedEntity).
fieldPos	Nombre	Position du champ dans la table de la base.
scope	Chaine	Portée de l'attribut (seuls les attributs dont la portée est publique apparaîtront).
indexed	Chaine	Si un type d'index a été sélectionné, cette propriété retournera true. Sinon, cette propriété n'apparaîtra pas.
type	Chaine	Type d'attribut de chaîne (booléen, blob, octet, date, durée, image, long, long64, numérique, chaîne, uuid ou mot) ou la dataclasse pour un attribut de relation N-> 1.
identifying	Booléen	Cette propriété retourne True si l'attribut est la clé primaire. Sinon, cette propriété n'apparaîtra pas.
path	Chaine	Name of the dataclass for a relatedEntity attribute, or name of the relation for a relatedEntities attribute.
foreignKey	Chaine	Pour un attribut relatedEntity, nom de l'attribut associé.
inverseName	Chaine	Nom de la relation opposée pour un attribut relatedEntity ou relatedEntities.

Clé primaire

L'objet clé retourne le nom de l'attribut (name) défini comme clé primaire pour la dataclass.

Exemple

Vous pouvez récupérer les informations concernant une dataclass spécifique.

```
GET /rest/$catalog/Employee
```

Résultat :

```
{
  name: "Employee",
  className: "Employee",
  collectionName: "EmployeeCollection",
  scope: "public",
  dataURI: "http://127.0.0.1:8081/rest/Employee",
  defaultTopSize: 20,
  extraProperties: {
    panelColor: "#76923C",
    __CDATA: "\n\n\t\t\n",
    panel: {
      isOpen: "true",
      pathVisible: "true",
      __CDATA: "\n\n\t\t\t\n",
      position: {
        X: "394",
        Y: "42"
      }
    }
  },
  attributes: [
    {
      name: "ID",
      kind: "storage",
      scope: "public",
      indexed: true,
      type: "long",
      identifying: true
    }
  ]
}
```

```
,  
{  
    name: "firstName",  
    kind: "storage",  
    scope: "public",  
    type: "string"  
},  
{  
    name: "lastName",  
    kind: "storage",  
    scope: "public",  
    type: "string"  
},  
{  
    name: "fullName",  
    kind: "calculated",  
    scope: "public",  
    type: "string",  
    readOnly: true  
},  
{  
    name: "salary",  
    kind: "storage",  
    scope: "public",  
    type: "number",  
    defaultFormat: {  
        format: "$###,###.00"  
    }  
},  
{  
    name: "photo",  
    kind: "storage",  
    scope: "public",  
    type: "image"  
},  
{  
    name: "employer",  
    kind: "relatedEntity",  
    scope: "public",  
    type: "Company",  
    path: "Company"  
},  
{  
    name: "employerName",  
    kind: "alias",  
    scope: "public",  
  
    type: "string",  
    path: "employer.name",  
    readOnly: true  
},  
{  
    name: "description",  
    kind: "storage",  
    scope: "public",  
    type: "string",  
    multiLine: true  
},  
,  
key: [  
    {  
        name: "ID"  
    }  
]  
}
```


\$directory

Le répertoire gère l'accès des utilisateurs via les requêtes REST.

\$directory/login

Ouvre une session REST sur votre application 4D et connecte l'utilisateur.

Description

Utilisez `$directory/login` pour ouvrir une session dans votre application 4D via REST et connectez un utilisateur. Vous pouvez également modifier le timeout par défaut de la session 4D.

Tous les paramètres doivent être passés dans les en-têtes d'une méthode POST :

Clé de l'en-tête	Valeur de l'en-tête
username-4D	Utilisateur - Non obligatoire
password-4D	Mot de passe - Non obligatoire
hashed-password-4D	Mot de passe haché - Non obligatoire
session-4D-length	Timeout d'inactivité de la session (en minutes). Ne peut pas être inférieur à 60 - Non obligatoire

Exemple

```
C_TEXT($response;$body_t)
ARRAY TEXT($hKey;3)
ARRAY TEXT($hValues;3)
$hKey{1}:="username-4D"
$hKey{2}:="hashed-password-4D"
$hKey{3}:="session-4D-length"
$hValues{1}:="john"
$hValues{2}:=Generate digest("123";4D digest)
$hValues{3}:=120
$httpStatus:=HTTP Request(HTTP POST method;"app.example.com:9000/rest/$directory/login";$body_t;$respon
```

Résultat :

Si la connexion a réussi, le résultat sera le suivant :

```
{
  "result": true
}
```

Sinon, la réponse sera la suivante :

```
{
  "result": false
}
```


\$info

Renvoie des informations sur les ensembles d'entités stockés couramment dans le cache de 4D Server ainsi que sur les sessions utilisateur

Description

En appelant cette requête pour votre projet, vous récupérez des informations dans les propriétés suivantes :

Propriété	Type	Description
cacheSize	Nombre	Taille du cache du serveur 4D.
usedCache	Nombre	La quantité de cache du serveur 4D utilisée.
entitySetCount	Nombre	Nombre de sélections d'entités.
entitySet	Collection	Une collection dans laquelle chaque objet contient des informations sur chaque sélection d'entités.
ProgressInfo	Collection	Une collection contenant des informations sur les indicateurs de progression.
sessionInfo	Collection	Une collection dans laquelle chaque objet contient des informations sur chaque session utilisateur.

entitySet

Pour chaque sélection d'entités stocké dans le cache de 4D Server, les informations retournées sont les suivantes :

Propriété	Type	Description
id	Chaine	Un UUID qui référence l'ensemble d'entités.
dataClass	Chaine	Nom de la dataclass.
selectionSize	Nombre	Nombre d'entités dans la sélection d'entités.
sorted	Booléen	Retourne vrai si l'ensemble a été trié (à l'aide de <code>\$orderby</code>) ou faux s'il n'est pas trié.
refreshed	Date	Date de création de l'ensemble d'entités ou de la dernière utilisation.
expires	Date	Date d'expiration de l'ensemble d'entités (Cette date/heure change chaque fois que l'ensemble d'entités est actualisé). La différence entre actualisé et expire est le timeout d'un ensemble d'entités. Cette valeur correspond soit à deux heures par défaut, soit à la valeur que vous avez définie à l'aide de <code>\$timeout</code> .

Pour plus d'informations sur la création d'une sélection d'entités, reportez-vous à `$method=entityset`. Si vous souhaitez supprimer la sélection d'entités du cache de 4D Server, utilisez `$method=release`.

4D crée également ses propres sélections d'entités à des fins d'optimisation, de sorte que ceux que vous créez avec `$method=entityset` ne soient pas les seuls à être retournés. Les informations de l'indicateur de progression répertoriées après les sélections d'entités sont utilisées en interne par 4D.

sessionInfo

Pour chaque session utilisateur, les informations suivantes sont retournées dans la collection `sessionInfo` :

Propriété	Type	Description
sessionID	Chaine	Un UUID qui référence la session.
userName	Chaine	Nom de l'utilisateur qui lance la session.
lifeTime	Nombre	La durée d'une session utilisateur en secondes (3600 par défaut).
expiration	Date	Date et heure d'expiration courante de la session utilisateur.

Exemple

Retourne des informations sur les ensembles d'entités stockés couramment dans le cache de 4D Server ainsi que sur les sessions utilisateur:

```
GET /rest/$info
```

Résultat :

```

{
cacheSize: 209715200,
usedCache: 3136000,
entitySetCount: 4,
entitySet: [
{
id: "1418741678864021B56F8C6D77F2FC06",
tableName: "Company",
selectionSize: 1,
sorted: false,
refreshed: "2011-11-18T10:30:30Z",
expires: "2011-11-18T10:35:30Z"
},
{
id: "CAD79E5BF339462E85DA613754C05CC0",
tableName: "People",
selectionSize: 49,
sorted: true,
refreshed: "2011-11-18T10:28:43Z",
expires: "2011-11-18T10:38:43Z"
},
{
id: "F4514C59D6B642099764C15D2BF51624",
tableName: "People",
selectionSize: 37,
sorted: false,
refreshed: "2011-11-18T10:24:24Z",
expires: "2011-11-18T12:24:24Z"
}
],
ProgressInfo: [
{
UserInfo: "flushProgressIndicator",
sessions: 0,
percent: 0
},
{
UserInfo: "indexProgressIndicator",
sessions: 0,
percent: 0
}
],
sessionInfo: [
{
sessionID: "6657ABBCEE7C3B4089C20D8995851E30",
userID: "36713176D42DB045B01B8E650E8FA9C6",
userName: "james",
lifeTime: 3600,
expiration: "2013-04-22T12:45:08Z"
},
{
sessionID: "A85F253EDE90CA458940337BE2939F6F",
userID: "00000000000000000000000000000000",
userName: "default guest",
lifeTime: 3600,
expiration: "2013-04-23T10:30:25Z"
}
]
}

```

Les informations de l'indicateur de progression répertoriées après les sélections d'entités sont utilisées en interne par 4D.

\$upload

Retourne un ID du fichier téléchargé sur le serveur

Description

Publiez cette requête lorsque vous souhaitez télécharger un fichier sur le serveur. S'il s'agit d'une image, passez `$rawPict=true`. Pour tous les autres fichiers, passez `$binary=true`.

Vous pouvez modifier le timeout, qui est par défaut de 120 secondes, en passant une valeur au paramètre `$timeout`.

Scénario de téléchargement

Supposons que vous souhaitez télécharger une image pour mettre à jour l'attribut image d'une entité.

Pour télécharger une image (ou tout autre fichier binaire), sélectionnez d'abord le fichier dans l'application cliente. Le fichier lui-même doit être transmis dans le corps de la requête.

Téléchargez ensuite l'image sélectionnée vers le serveur 4D à l'aide d'une requête telle que :

```
POST /rest/$upload?$rawPict=true
```

Par conséquent, le serveur retourne un ID qui identifie le fichier :

Réponse :

```
{ "ID": "D507BC03E613487E9B4C2F6A0512FE50" }
```

Utilisez ensuite cet ID pour l'ajouter à un attribut en utilisant `$method=update` pour ajouter l'image à une entité. La requête est la suivante :

```
POST /rest/Employee/?$method=update
```

Données POST :

```
{
  __KEY: "12",
  __STAMP: 4,
  photo: { "ID": "D507BC03E613487E9B4C2F6A0512FE50" }
}
```

Réponse :

L'entité modifiée est retournée :

```
{
    "__KEY": "12",
    "__STAMP": 5,
    "uri": "http://127.0.0.1:8081/rest/Employee(12)",
    "ID": 12,
    "firstName": "John",
    "lastName": "Smith",
    "photo":
    {
        "__deferred":
        {
            "uri": "/rest/Employee(12)/photo?$imageformat=best&$version=1&$expand=photo",
            "image": true
        }
    }
},}
```

Exemple avec un client 4D HTTP

L'exemple suivant montre comment télécharger un fichier .pdf vers le serveur à l'aide du client 4D HTTP.

```
var $params : Text
var $response : Object
var $result : Integer
var $blob : Blob

ARRAY TEXT($headerNames; 1)
ARRAY TEXT($headerValues; 1)

$url:="localhost:80/rest/$upload?$binary=true" //préparer une requête the REST

$headerNames{1}:="Content-Type"
$headerValues{1}:="application/octet-stream"

DOCUMENT TO BLOB("c:\\invoices\\inv003.pdf"; $blob) //Charger le binaire

//Execute the first POST request to upload the file
$result:=HTTP Request(HTTP POST method; $url; $blob; $response; $headerNames; $headerValues)

If ($result=200)
    var $data : Object
    $data:=New object
    $data.__KEY:="3"
    $data.__STAMP:="3"
    $data.pdf:=New object("ID"; String($response.ID))

    $url:="localhost:80/rest/Invoices?$method=update" //seconde requête pour mettre à jour l'entité

    $headerNames{1}:="Content-Type"
    $headerValues{1}:="application/json"

    $result:=HTTP Request(HTTP POST method; $url; $data; $response; $headerNames; $headerValues)
Else
    ALERT(String($result)+" Error")
End if
```

{dataClass}

Les noms de dataclass peuvent être utilisés directement dans les requêtes REST pour opérer avec des entités, des sélections d'entités (entity selections) ou des fonctions de classe (class functions) de la dataclass.

Syntaxe

Syntaxe	Exemple	Description
{dataClass}	/Employee	Renvoie toutes les données (par défaut les 100 premières entités) de la dataclass
{dataClass}({clé})	/Employee(22)	Renvoie les données de l'entité spécifique définie par la clé primaire de la dataclass
{dataClass}:{attribute}(value)	/Employee:firstName(John)	Renvoie les données d'une entité dans laquelle la valeur de l'attribut est définie
{dataClass}/{DataClassClassFunction}	/City/getCity	Exécute une fonction de classe d'une dataclass
{dataClass}({EntitySelectionClassFunction})	/City/getPopulation/?\$filter="ID<3"	Exécute une fonction de classe d'une sélection d'entité
{dataClass}({key})/{EntityClassFunction}	City(2)/getPopulation	Exécute une fonction de classe d'une entité

Les appels de fonction sont détaillés dans la section [Appeler des fonctions de classe ORDA](#).

{dataClass}

Renvoie toutes les données (par défaut, les 100 premières entités) pour une dataclass spécifique (par exemple, Société)

Description

Lorsque vous appelez ce paramètre dans votre requête REST, les 100 premières entités sont renvoyées, sauf si vous avez spécifié une valeur à l'aide de `$top/$limit`.

Voici une description des données renvoyées :

Propriété	Type	Description
__entityModel	Chaine	Nom de la dataclass.
__COUNT	Nombre	Nombre d'entités dans la dataclass.
__SENT	Nombre	Nombre d'entités envoyées par la requête REST. Ce nombre peut être le nombre total d'entités s'il est inférieur à la valeur définie par <code>\$top/\$limit</code> .
__FIRST	Nombre	Numéro d'entité à partir duquel la sélection commence. Soit 0 par défaut soit la valeur définie par <code>\$skip</code> .
__ENTITIES	Collection	Cette collection d'objets contient un objet pour chaque entité avec tous ses attributs. Tous les attributs relationnels sont renvoyés en tant qu'objets avec un URI pour obtenir des informations concernant le parent.

Chaque entité contient les propriétés suivantes :

Propriété	Type	Description
__KEY	Chaine	Valeur de la clé primaire définie pour la dataclass.
__TIMESTAMP	Date	Horodatage de la dernière modification de l'entité
__STAMP	Nombre	Tampon interne qui est nécessaire lors de la modification des valeurs de l'entité lors de l'utilisation de <code>\$method=update</code> .

Si vous souhaitez indiquer les attributs à retourner, définissez-les à l'aide de la syntaxe suivante `{attribut1, attribut2, ...}`. Par exemple :

```
GET /rest/Company{name, address}
```

Exemple

Retourne toutes les données d'une dataclass spécifique.

```
GET /rest/Company
```

Résultat :

```
{
    "__entityModel": "Company",
    "__GlobalStamp": 51,
    "__COUNT": 250,
    "__SENT": 100,
    "__FIRST": 0,
    "__ENTITIES": [
        {
            "__KEY": "1",
            "__TIMESTAMP": "2020-04-10T10:44:49.927Z",
            "__STAMP": 1,
            "ID": 1,
            "name": "Adobe",
            "address": null,
            "city": "San Jose",
            "country": "USA",
            "revenues": 500000,
            "staff": {
                "__deferred": {
                    "uri": "http://127.0.0.1:8081/rest/Company(1)/staff?$expand=staff"
                }
            }
        },
        {
            "__KEY": "2",
            "__TIMESTAMP": "2018-04-25T14:42:18.351Z",
            "__STAMP": 1,
            "ID": 2,
            "name": "Apple",
            "address": null,
            "city": "Cupertino",
            "country": "USA",
            "revenues": 890000,
            "staff": {
                "__deferred": {
                    "uri": "http://127.0.0.1:8081/rest/Company(2)/staff?$expand=staff"
                }
            }
        },
        {
            "__KEY": "3",
            "__TIMESTAMP": "2018-04-25T14:42:18.351Z",
            "__STAMP": 1,
            "ID": 3,
            "name": "Microsoft",
            "address": null,
            "city": "Redmond",
            "country": "USA",
            "revenues": 850000,
            "staff": {
                "__deferred": {
                    "uri": "http://127.0.0.1:8081/rest/Company(3)/staff?$expand=staff"
                }
            }
        }
    ]
}
```

```

    "__TIMESTAMP": "2018-04-23T09:03:49.021Z",
    "__STAMP": 2,
    "ID": 3,
    "name": "4D",
    "address": null,
    "city": "Clichy",
    "country": "France",
    "revenues": 700000,
    "staff": {
        "__deferred": {
            "uri": "http://127.0.0.1:8081/rest/Company(3)/staff?$expand=staff"
        }
    }
},
{
    "__KEY": "4",
    "__TIMESTAMP": "2018-03-28T14:38:07.430Z",
    "__STAMP": 1,
    "ID": 4,
    "name": "Microsoft",
    "address": null,
    "city": "Seattle",
    "country": "USA",
    "revenues": 650000,
    "staff": {
        "__deferred": {
            "uri": "http://127.0.0.1:8081/rest/Company(4)/staff?$expand=staff"
        }
    }
}
....//plus d'entités ici
]
}

```

{dataClass}({clé})

Renvoie les données de l'entité spécifique définie par la clé primaire de la dataclass, par exemple, `Company (22)` ou `Company ("IT0911AB2200")`

Description

En passant la dataclass et une clé, vous pouvez récupérer toutes les informations publiques de cette entité. En passant la dataclass et une clé, vous pouvez récupérer toutes les informations publiques de cette entité. Pour plus d'informations sur la définition d'une clé primaire, reportez-vous à la section Modification de la clé primaire dans l'éditeur de modèle de données.

Pour plus d'informations sur les données rentrées, reportez-vous à [{DataStoreClass}](#).

Si vous souhaitez indiquer les attributs à retourner, définissez-les à l'aide de la syntaxe suivante `{attribut1, attribut2, ...}`. Par exemple :

```
GET /rest/Company(1)/name,address
```

Si vous souhaitez développer un attribut relationnel à l'aide de `$expand`, vous devez l'indiquer comme suit :

```
GET /rest/Company(1)/name,address,staff?$expand=staff
```

Exemple

La requête suivante retourne toutes les données publiques de la dataclass Company dont la clé est 1.

```
GET /rest/Company(1)
```

Résultat :

```
{  
    "__entityModel": "Company",  
    "__KEY": "1",  
    "__TIMESTAMP": "2020-04-10T10:44:49.927Z",  
    "__STAMP": 2,  
    "ID": 1,  
    "name": "Apple",  
    "address": Infinite Loop,  
    "city": "Cupertino",  
    "country": "USA",  
    "url": http://www.apple.com,  
    "revenues": 500000,  
    "staff": {  
        "__deferred": {  
            "uri": "http://127.0.0.1:8081/rest/Company(1)/staff?$expand=staff"  
        }  
    }  
}
```

{dataClass}:{attribute}(value)

Renvoie les données d'une entité dans laquelle la valeur de l'attribut est définie

Description

En passant la *dataClass* et un *attribut* avec une valeur, vous pouvez récupérer toutes les informations publiques de cette entité. La valeur est une valeur unique pour l'attribut, mais ce n'est pas la clé primaire.

```
GET /rest/Company:companyCode(Acme001)
```

Si vous souhaitez indiquer les attributs à retourner, définissez-les à l'aide de la syntaxe suivante `{attribut1, attribut2, ...}`. Par exemple :

```
GET /rest/Company:companyCode(Acme001)/name,address
```

Si vous souhaitez utiliser un attribut relationnel à l'aide de `$attributes`, vous devez l'indiquer comme suit :

```
GET /rest/Company:companyCode(Acme001)?$attributes=name,address,staff.name
```

Exemple

La requête suivante retourne toutes les données publiques de l'employé nommé "Jones".

```
GET /rest/Employee:lastname(Jones)
```

\$asArray

Retourne le résultat d'une requête sous forme de tableau (c'est-à-dire une collection) au lieu d'un objet JSON.

Description

Si vous souhaitez obtenir la réponse sous forme de tableau, il vous suffit d'ajouter `$asArray` à votre requête REST (ex : `, $asArray=true`).

Exemple

Voici un exemple pour obtenir une réponse sous forme de tableau.

```
GET /rest/Company/?$filter="name begin a"&$top=3&$asArray=true
```

Réponse :

```
[  
  {  
    "__KEY": 15,  
    "__STAMP": 0,  
    "ID": 15,  
    "name": "Alpha North Yellow",  
    "creationDate": "!!0000-00-00!!",  
    "revenues": 82000000,  
    "extra": null,  
    "comments": "",  
    "__GlobalStamp": 0  
,  
  {  
    "__KEY": 34,  
    "__STAMP": 0,  
    "ID": 34,  
    "name": "Astral Partner November",  
    "creationDate": "!!0000-00-00!!",  
    "revenues": 90000000,  
    "extra": null,  
    "comments": "",  
    "__GlobalStamp": 0  
,  
  {  
    "__KEY": 47,  
    "__STAMP": 0,  
    "ID": 47,  
    "name": "Audio Production Uniform",  
    "creationDate": "!!0000-00-00!!",  
    "revenues": 28000000,  
    "extra": null,  
    "comments": "",  
    "__GlobalStamp": 0  
}  
]
```

Les mêmes données au format JSON par défaut :

```
{
  "__entityModel": "Company",
  "__GlobalStamp": 50,
  "__COUNT": 52,
  "__FIRST": 0,
  "__ENTITIES": [
    {
      "__KEY": "15",
      "__TIMESTAMP": "2018-03-28T14:38:07.434Z",
      "__STAMP": 0,
      "ID": 15,
      "name": "Alpha North Yellow",
      "creationDate": "0!0!0",
      "revenues": 82000000,
      "extra": null,
      "comments": "",
      "__GlobalStamp": 0,
      "employees": {
        "__deferred": {
          "uri": "/rest/Company(15)/employees?$expand=employees"
        }
      }
    },
    {
      "__KEY": "34",
      "__TIMESTAMP": "2018-03-28T14:38:07.439Z",
      "__STAMP": 0,
      "ID": 34,
      "name": "Astral Partner November",
      "creationDate": "0!0!0",
      "revenues": 90000000,
      "extra": null,
      "comments": "",
      "__GlobalStamp": 0,
      "employees": {
        "__deferred": {
          "uri": "/rest/Company(34)/employees?$expand=employees"
        }
      }
    },
    {
      "__KEY": "47",
      "__TIMESTAMP": "2018-03-28T14:38:07.443Z",
      "__STAMP": 0,
      "ID": 47,
      "name": "Audio Production Uniform",
      "creationDate": "0!0!0",
      "revenues": 28000000,
      "extra": null,
      "comments": "",
      "__GlobalStamp": 0,
      "employees": {
        "__deferred": {
          "uri": "/rest/Company(47)/employees?$expand=employees"
        }
      }
    }
  ],
  "__SENT": 3
}
```

\$atomic/\$atonce

Autorise les actions d'une requête REST à faire partie d'une transaction. Si aucune erreur n'est générée, la transaction est validée. Sinon, la transaction est annulée.

Description

Lorsque plusieurs actions sont réunies, vous pouvez utiliser `$atomic/$atonce` pour vous assurer qu'aucune action ne se réalise si l'une d'elle échoue. Vous pouvez utiliser `$atomic` ou `$atonce`.

Exemple

Nous appelons la requête REST suivante dans une transaction.

```
POST /rest/Employee?method=update&$atomic=true
```

Données POST :

```
[  
{  
    "__KEY": "200",  
    "firstname": "John"  
},  
{  
    "__KEY": "201",  
    "firstname": "Harry"  
}]
```

Nous obtenons l'erreur suivante dans la deuxième entité ; la première entité n'est donc pas sauvegardée :

```
{  
    "__STATUS": {  
        "success": true  
    },  
    "__KEY": "200",  
    "__STAMP": 1,  
    "uri": "/rest/Employee(200)",  
    "__TIMESTAMP": "!!2020-04-03!!",  
    "ID": 200,  
    "firstname": "John",  
    "lastname": "Keeling",  
    "isWoman": false,  
    "numberOfKids": 2,  
    "addressID": 200,  
    "gender": false,  
    "address": {  
        "__deferred": {  
            "uri": "/rest/Address(200)",  
            "__KEY": "200"  
        }  
    },  
    "__ERROR": [  
        {  
            "message": "Cannot find entity with \"201\" key in the \"Employee\" dataclass",  
            "componentSignature": "dbmg",  
            "errCode": 1542  
        }  
    ]  
}
```

Même si le salaire de la première entité porte la valeur 45000, cette valeur n'a pas été sauvegardée sur le serveur et le timestamp (__STAMP)* n'a pas été modifié. Si nous rechargeons l'entité, la valeur précédente s'affichera.

\$attributes

Permet de sélectionner les attributs relationnels à obtenir à partir de la dataclass (par exemple, `Company(1)?$attributes=employees.lastname` or `Employee?$attributes=employer.name`).

Description

Lorsque vous avez des attributs relationnels dans une dataclass, utilisez `$attributes` pour définir le chemin des attributs dont vous souhaitez obtenir les valeurs pour l'entité ou les entités associées.

Vous pouvez appliquer des `$attributes` à une entité (par exemple, `People (1)`) ou à une entity selection (par exemple, `People/$entityset/0AF4679A5C394746BFEB68D2162A19FF`).

- Si `$attributes` n'est pas spécifié dans une requête, ou si la valeur "*" est passée, tous les attributs disponibles sont extraits. Les attributs d'entité relative sont extraits avec la forme simple : un objet avec la propriété `__KEY` (clé primaire) et `URI`. Les attributs des entités relatives ne sont pas extraits.
- Si `$attributes` est spécifié pour les attributs **d'entité relative** :
 - `$attributes=relatedEntity` : l'entité relative est retournée sous une forme simple (propriété `__KEY` différée (clé primaire)) et `URI`.
 - `$attributes=relatedEntity.*` : tous les attributs de l'entité relative sont retournés
 - `$attributes=relatedEntity.attributePath1, relatedEntity.attributePath2, ...` : seuls ces attributs de l'entité relative sont retournés.
- Si `$attributes` est spécifié pour les attributs d'entités relatives :
 - `$attributes=relatedEntities.*` : toutes les propriétés des entités relatives sont retournées
 - `$attributes=relatedEntities.attributePath1, relatedEntity.attributePath2, ...` : seuls ces attributs des entités relatives sont retournés.

Exemples avec plusieurs entités relatives

Si nous passons la requête REST suivante pour la dataclasse Company (qui possède un attribut de relation "employees"):

```
GET /rest/Company(1)/?$attributes=employees.lastname
```

Réponse :

```
{
    "__entityModel": "Company",
    "__KEY": "1",
    "__TIMESTAMP": "2018-04-25T14:41:16.237Z",
    "__STAMP": 2,
    "employees": {
        "__ENTITYSET": "/rest/Company(1)/employees?$expand=employees",
        "__GlobalStamp": 50,
        "__COUNT": 135,
        "__FIRST": 0,
        "__ENTITIES": [
            {
                "__KEY": "1",
                "__TIMESTAMP": "2019-12-01T20:18:26.046Z",
                "__STAMP": 5,
                "lastname": "ESSEAL"
            },
            {
                "__KEY": "2",
                "__TIMESTAMP": "2019-12-04T10:58:42.542Z",
                "__STAMP": 6,
                "lastname": "JONES"
            },
            ...
        ]
    }
}
```

Si vous souhaitez obtenir tous les attributs des employés :

```
GET /rest/Company(1)/?$attributes=employees.*
```

Si vous souhaitez obtenir le nom de famille et les attributs de nom de poste des employés :

```
GET /rest/Company(1)/?$attributes=employees.lastname,employees.jobname
```

Exemples avec une entité relative

Si nous passons la requête REST suivante pour la dataclass Employee (qui a plusieurs attributs relationnels, y compris "employer") :

```
GET /rest/Employee(1)?$attributes=employer.name
```

Réponse :

```
{
    "__entityModel": "Employee",
    "__KEY": "1",
    "__TIMESTAMP": "2019-12-01T20:18:26.046Z",
    "__STAMP": 5,
    "employer": {
        "__KEY": "1",
        "__TIMESTAMP": "2018-04-25T14:41:16.237Z",
        "__STAMP": 0,
        "name": "Adobe"
    }
}
```

Si vous souhaitez obtenir tous les attributs de l'employeur :

```
GET /rest/Employee(1)?$attributes=employer.*
```

Si vous souhaitez obtenir les noms de famille de tous les employés de l'employeur :

```
GET /rest/Employee(1)?$attributes=employer.employees.lastname
```

\$binary

Passez "true" pour enregistrer le BLOB en tant que document (vous devez également passer `$expand={blobAttributeName}`)

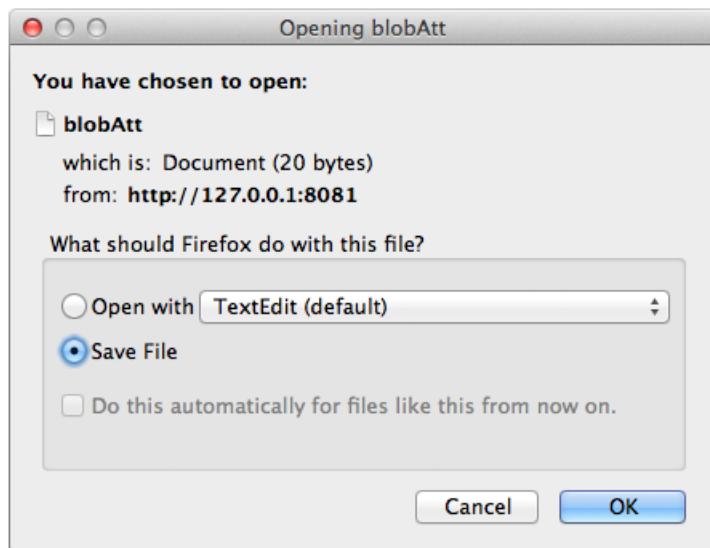
Description

`$binary` vous permet d'enregistrer le BLOB en tant que document. Vous devez également utiliser la commande `$expand`.

Lorsque vous faites la requête suivante :

```
GET /rest/Company(11)/blobAtt?$binary=true&$expand=blobAtt
```

Il vous sera demandé où enregistrer le BLOB sur le disque :



\$compute

Calculez des attributs spécifiques (par exemple, Employee/salary/?\$compute=sum) ou dans le cas d'un attribut Objet (par exemple, Employee/objectAtt.property1/?\$compute=sum)

Description

Ce paramètre vous permet de réaliser des calculs avec vos données.

Si vous souhaitez effectuer un calcul avec un attribut, saisissez ce qui suit :

```
GET /rest/Employee/salary/?$compute=$all
```

Si vous souhaitez passer un attribut Objet, vous devez passer l'une de ses propriétés. Par exemple :

```
GET /rest/Employee/objectAtt.property1/?$compute=$all
```

Vous pouvez utiliser l'un des mots clés suivants :

Mot-clé	Description
\$all	Un objet JSON qui définit toutes les fonctions de l'attribut (moyenne, nombre, min, max et somme pour les attributs de type Numérique et count, min et max pour les attributs de type Chaîne)
average	Obtenir la moyenne d'un attribut numérique
count	Obtenir le nombre total dans la collection ou la dataclass (dans les deux cas, vous devez spécifier un attribut)
min	Obtenir la valeur minimale d'un attribut numérique ou la plus petite valeur d'un attribut de type Chaîne
max	Obtenir la valeur maximale d'un attribut numérique ou la plus grande valeur d'un attribut de type Chaîne
sum	Obtenir la somme d'un attribut numérique

Exemple

Si vous souhaitez obtenir tous les calculs pour un attribut de type Numérique, vous pouvez écrire :

```
GET /rest/Employee/salary/?$compute=$all
```

Réponse :

```
{
  "salary": {
    "count": 4,
    "sum": 335000,
    "average": 83750,
    "min": 70000,
    "max": 99000
  }
}
```

Si vous souhaitez obtenir tous les calculs pour un attribut de type Chaîne, vous pouvez écrire :

```
GET /rest/Employee/firstName/?$compute=$all
```

Réponse :

```
{  
    "salary": {  
        "count": 4,  
        "min": Anne,  
        "max": Victor  
    }  
}
```

Si vous souhaitez obtenir un calcul avec un attribut, vous pouvez écrire ce qui suit :

```
GET /rest/Employee/salary/?$compute=sum
```

Réponse :

```
235000
```

Si vous souhaitez effectuer un calcul avec un attribut Objet, vous pouvez saisir ce qui suit :

```
GET /rest/Employee/objectAttribute.property1/?$compute=sum
```

Réponse :

```
45
```

\$distinct

Retourne les différentes valeurs d'un attribut spécifique dans une collection (par exemple, `Company/name? $filter="name=a*"$distinct=true`)

Description

`$distinct` vous permet de retourner une collection contenant les différentes valeurs d'une requête sur un attribut spécifique. Un seul attribut dans la dataclass peut être spécifié. Généralement, le type Chaîne est idéal; cependant, vous pouvez également l'utiliser sur n'importe quel type d'attribut pouvant contenir plusieurs valeurs.

Vous pouvez également utiliser `$skip` et `$top/$limit` si vous souhaitez parcourir la sélection avant qu'elle ne soit placée dans un tableau.

Exemple

Dans l'exemple ci-dessous, nous souhaitons récupérer les différentes valeurs d'un nom de société commençant par la lettre "a" :

```
GET /rest/Company/name?$filter="name=a*"$distinct=true
```

Réponse :

```
[  
  "Adobe",  
  "Apple"  
]
```

\$entityset

Après avoir [créé un ensemble d'entités](#) à l'aide de `$method=entityset`, vous pouvez ensuite l'utiliser ultérieurement.

Syntaxe

Syntaxe	Exemple	Description
<code>\$entityset/{entitySetID}</code>	<code>/People/\$entityset/0ANUMBER</code>	Récupère un ensemble d'entités existant
<code>\$entityset/{entitySetID}?\$operator...&\$otherCollection</code>	<code>/Employee/\$entityset/0ANUMBER?</code> <code>\$logicOperator=AND</code> <code>&\$otherCollection=C0ANUMBER</code>	Crée un nouvel ensemble d'entités à partir de la comparaison d'ensembles d'entités existants

\$entityset/{entitySetID}

Récupère un ensemble d'entités existant (e.g., `People/$entityset/0AF4679A5C394746BFEB68D2162A19FF`)

Description

Cette syntaxe vous permet d'exécuter toute opération sur un ensemble d'entités défini.

Étant donné que les ensembles d'entités sont limités par le temps (par défaut ou bien après avoir appelé `$timeout` pour définir la limite souhaitée), vous pouvez appeler `$savedfilter` et `$savedorderby` pour sauvegarder le filtre et trier par instructions lorsque vous créez un ensemble d'entités.

Lorsque vous récupérez un ensemble d'entités existant stocké dans le cache de 4D Server, vous pouvez également appliquer l'un des éléments suivants à l'ensemble d'entités : `$expand`, `$filter`, `$orderby`, `$skip`, et `$top/$limit`.

Exemple

Après avoir créé un ensemble d'entités, l'ID de l'ensemble d'entités est retourné avec les données. Vous appelez cet ID de la manière suivante :

```
GET /rest/Employee/$entityset/9718A30BF61343C796345F3BE5B01CE7
```

\$entityset/{entitySetID}?\$operator...&\$otherCollection

Créez un autre ensemble d'entités basé sur des ensembles d'entités préalablement créés

Paramètres	Type	Description
<code>\$operator</code>	Chaine	L'un des opérateurs logiques à tester avec l'autre ensemble d'entités
<code>\$otherCollection</code>	Chaine	ID de l'ensemble d'entités

Description

Après avoir créé un ensemble d'entités (ensemble d'entités n°1) à l'aide de `$method=entityset`, vous pouvez ensuite créer un autre ensemble d'entités en utilisant la syntaxe `$entityset/{entitySetID}?$operator...&$otherCollection`, la propriété `$operator` (dont les valeurs sont indiquées ci-dessous), et un autre ensemble d'entités (jeu d'entités n°2) défini par la propriété `$otherCollection`. Les deux ensembles d'entités doivent être dans la même dataclass.

Vous pouvez ensuite créer un autre ensemble d'entités contenant les résultats de cet appel en utilisant le `$method=entityset` à la fin de la requête REST.

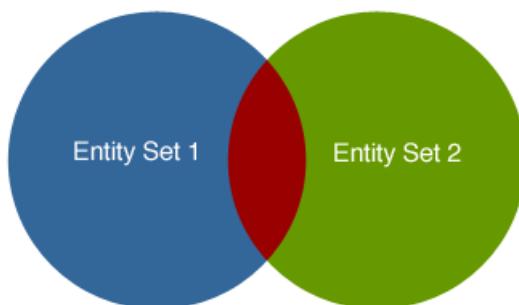
Voici les opérateurs logiques :

Opérateur	Description
AND	Retourne les entités communes aux deux ensembles d'entités
OR	Retourne les entités contenues dans les deux ensembles d'entités
EXCEPT	Retourne les entités de l'ensemble d'entités #1 moins celles de l'ensemble d'entités #2
INTERSECT	Retourne true ou false s'il existe une intersection des entités dans les deux ensembles d'entités (ce qui signifie qu'au moins une entité est commune aux deux ensembles d'entités)

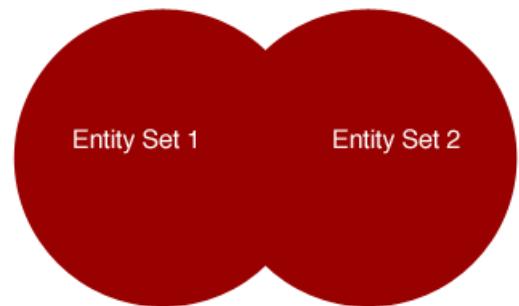
Les opérateurs logiques ne sont pas sensibles à la casse, vous pouvez donc écrire "AND" ou "and".

Vous trouverez ci-dessous une représentation des opérateurs logiques basés sur deux ensembles d'entités. La section rouge correspond à ce qui est retourné.

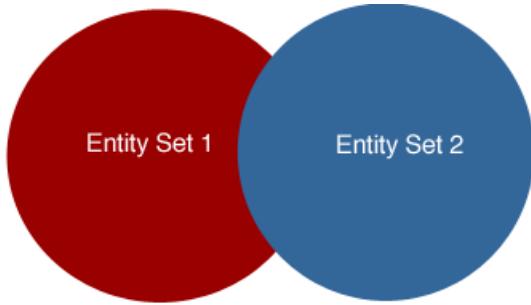
AND



OR



EXCEPT



La syntaxe est la suivante :

```
GET /rest/dataClass/$entityset/entitySetID?$logicOperator=AND&$otherCollection=entitySetID
```

Exemple

Dans l'exemple ci-dessous, nous retournons les entités qui se trouvent dans les deux ensembles d'entités puisque nous utilisons l'opérateur logique AND :

```
GET /rest/Employee/$entityset/9718A30BF61343C796345F3BE5B01CE7?
```

```
$logicOperator=AND&$otherCollection=C05A0D887C664D4DA1B38366DD21629B
```

Si nous souhaitons savoir si les deux ensembles d'entités se croisent, nous pouvons écrire ce qui suit :

```
GET /rest/Employee/$entityset/9718A30BF61343C796345F3BE5B01CE7?  
$logicOperator=intersect&$otherCollection=C05A0D887C664D4DA1B38366DD21629B
```

S'il existe une intersection, cette requête retourne true. Sinon, elle retourne false.

Dans l'exemple suivant, nous créons un nouvel ensemble d'entités qui combine toutes les entités des deux ensembles d'entités :

```
GET /rest/Employee/$entityset/9718A30BF61343C796345F3BE5B01CE7?  
$logicOperator=OR&$otherCollection=C05A0D887C664D4DA1B38366DD21629B&$method=entityset
```

\$expand

Développe une image stockée dans un attribut Image (par exemple, `Employee(1)/photo?`

`$imageformat=best&$expand=photo`)

ou

Développe un attribut BLOB pour l'enregistrer.

Compatibilité : pour des raisons de compatibilité, `$expand` peut être utilisé pour développer un attribut relationnel (par exemple, `Company(1)?$expand= staff` ou `EmployeeEmployee/?$filter="firstName BEGIN a"&$expand=employer`). Il est cependant recommandé d'utiliser `$attributes` pour cette fonctionnalité.

Affichage d'un attribut d'image

Si vous souhaitez afficher intégralement un attribut d'image, saisissez ce qui suit :

```
GET /rest/Employee(1)/photo?$imageformat=best&$version=1&$expand=photo
```

Pour plus d'informations sur les formats d'image, reportez-vous à `$imageformat`. Pour plus d'informations sur le paramètre de version, reportez-vous à `$version`.

Enregistrement d'un attribut BLOB sur le disque

Si vous souhaitez enregistrer un BLOB stocké dans votre dataclass, vous pouvez écrire ce qui suit en passant également "true" à `$binary` :

```
GET /rest/Company(11)/blobAtt?$binary=true&$expand=blobAtt
```

\$filter

Permet de rechercher les données d'une dataclass ou d'une méthode (par exemple, `$filter="firstName!='' AND salary>30000"`)

Description

Ce paramètre vous permet de définir le filtre de votre dataclass ou de votre méthode.

Utiliser un filtre simple

Un filtre est composé des éléments suivants :

{attribut} {comparateur} {valeur}

Par exemple : `$filter="firstName=john"` où `firstName` est l'attribut, `=` est le comparateur et `john` est la valeur.

Utiliser un filtre complexe

Un filtre plus complexe est composé des éléments suivants, qui joint deux requêtes :

{attribut} {comparateur} {valeur} {AND/OR/EXCEPT} {attribut} {comparateur} {valeur}

Par exemple : `$filter="firstName=john AND salary>20000"` où `firstName` et `salary` sont les attributs de la dataclasse "Employee".

Utiliser la propriété params

Vous pouvez également utiliser la propriété params de 4D.

{attribut} {comparateur} {placeholder} {AND/OR/EXCEPT} {attribut} {comparateur} {placeholder}&\$params='["{value1}","{value2}"]'"

Par exemple : `$filter="firstName=:1 AND salary>:2"&$params='["john",20000]'"` où `firstName` et `salary` sont les attributs de la dataclass "Employee".

Pour plus d'informations sur la façon de rechercher des données dans 4D, reportez-vous à la documentation de [dataClass.query\(\)](#).

Lorsque vous insérez des guillemets ('') ou des guillemets doubles (""), vous devez les échapper en utilisant leur code de caractère :

- Guillemets (): \u0027
- Guillemets doubles (""): \u0027

Par exemple, vous pouvez écrire ce qui suit lors du passage d'une valeur avec un guillemet lors de l'utilisation de la propriété `params` :

`http://127.0.0.1:8081/rest/Person/?$filter="lastName=:1"&$params='["\u0027Reilly"]'`

Si vous passez la valeur directement, vous pouvez écrire ce qui suit :

`http://127.0.0.1:8081/rest/Person/?$filter="lastName=O'Reilly"`

Attribut

Si l'attribut se trouve dans la même dataclass, vous pouvez simplement le passer directement (par exemple, `firstName`). Cependant, si vous souhaitez lancer une requête dans une autre dataclass, vous devez inclure le nom de l'attribut relationnel et le nom d'attribut, c'est-à-dire le chemin d'accès (par exemple, `employeur.nom`). Le nom

d'attribut est sensible à la casse (`firstName` n'est pas égal à `FirstName`).

Vous pouvez également rechercher des attributs de type Objet en utilisant la notation par points. Par exemple, si vous avez un attribut dont le nom est "objAttribute" avec la structure suivante :

```
{  
    prop1: "this is my first property",  
    prop2: 9181,  
    prop3: ["abc", "def", "ghi"]  
}
```

Vous pouvez rechercher dans l'objet en écrivant ce qui suit :

```
GET /rest/Person/?filter="objAttribute.prop2 == 9181"
```

Comparateur

Le comparateur doit être l'une des valeurs suivantes :

Comparateur	Description
=	est égal à
!=	différent de
>	supérieur à
>=	supérieur ou égal à
<	inférieur à
<=	inférieur ou égal à
begin	commence avec

Exemples

Dans l'exemple suivant, nous recherchons tous les employés dont le nom de famille commence par un "j" :

```
GET /rest/Employee?$filter="lastName begin j"
```

Dans cet exemple, nous recherchons dans la dataclass "Employee" tous les employés d'une entreprise autre que Acmedont et dont le salaire est supérieur à 20 000 :

```
GET /rest/Employee?$filter="salary>20000 AND  
employer.name!=acme"&$orderby="lastName,firstName"
```

Dans cet exemple, nous recherchons dans la dataclass "Person" toutes les personnes dont la propriété numérique, de l'attribut anotherobj de type Objet, est supérieure à 50 :

```
GET /rest/Person/?filter="anotherobj.mynum > 50"
```

\$imageformat

Définit le format d'image à utiliser pour récupérer des images (par exemple, `$imageformat=png`)

Description

Définissez le format à utiliser pour afficher les images. Par défaut, le meilleur format d'image sera choisi. Vous pouvez cependant sélectionner l'un des formats suivants :

Type	Description
GIF	Format GIF
PNG	Format PNG
JPEG	Format JPEG
TIFF	Format TIFF
best	Meilleur format basé sur l'image

Une fois que vous avez défini le format, vous devez passer l'attribut de l'image à `$expand` pour charger complètement la photo.

S'il n'y a pas d'image à charger ou si le format ne permet pas de charger l'image, la réponse sera vide.

Exemple

L'exemple suivant définit le format d'image au format JPEG, quel que soit le véritable type de la photo et passe le véritable numéro de version envoyé par le serveur :

```
GET /rest/Employee(1)/photo?$imageformat=jpeg&$version=3&$expand=photo
```

\$lock

Verrouille et déverrouille une entité en utilisant le [mécanisme pessimiste](#).

Syntaxe

Pour verrouiller une entité pour les autres sessions et processus 4D :

```
/?$lock=true
```

Pour déverrouiller l'entité pour les autres sessions et processus 4D :

```
/?$lock=false
```

La propriété `lockKindText` est "Locked by session".

Description

Les verrouillages déclenchés par l'API REST sont placés au niveau de la [session](#).

Une entité verrouillée est considérée comme *verrouillée* (c'est-à-dire que les actions de verrouillage / déverrouillage / mise à jour / suppression ne sont pas possibles) par :

- d'autres sessions REST
- les process 4D (client/serveur, datastore distant, monoposte) exécutés sur le serveur REST.

Une entité verrouillée par l'API REST peut être déverrouillée uniquement :

- par son cadenas, c'est-à-dire un `/?$lock=false` dans la session REST qui définit `/?$lock=true`
- ou si le [timeout d'inactivité](#) de la session est atteint (la session est fermée).

Réponse

Une requête `?$lock` retourne un objet JSON avec `"result":true` si l'opération de verrouillage est réussie et `"result":false` si elle échoue.

L'objet "`__STATUS`" retourné possède les propriétés suivantes :

Propriété		Type	Description
			<i>Disponible uniquement en cas de succès:</i>
success		boolean	vrai si l'action de verrouillage a été réussie (ou si l'entité est déjà verrouillée dans la session courante), sinon faux (non retourné dans ce cas).
			<i>Disponible uniquement en cas d'erreur :</i>
status		number	Code d'erreur, voir ci-dessous
statusText		Texte	Description de l'erreur, voir ci-dessous
lockKind		number	Code de verrouillage
lockKindText		Texte	"Locked by session" en cas de verrouillage par une session REST, "Locked by record" en cas de verrouillage par un process 4D
lockInfo		object	Information sur l'origine du verrouillage. Les propriétés retournées dépendent de l'origine du verrouillage (process 4D ou session REST).
			<i>Disponible uniquement pour un verrouillage par process 4D:</i>
	task_id	number	ID du process
	user_name	Texte	Nom d'utilisateur de la session sur la machine
	user4d_alias	Texte	Nom ou alias de l'utilisateur 4D
	user4d_id	number	Identifiant utilisateur dans le répertoire de la base 4D
	host_name	Texte	Nom de la machine
	task_name	Texte	Nom du process
	client_version	Texte	Version du client
			<i>Disponible uniquement pour un verrouillage par session REST :</i>
	host	Texte	URL d'origine du verrouillage de l'entité (ex : "127.0.0.1:8043")
	IPAddr	Texte	Adresse IP d'origine du verrouillage (ex. 127.0.0.1")
	recordNumber	number	Numéro de l'enregistrement verrouillé
	userAgent	Texte	userAgent de l'origine du verrouillage (ex : "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.110 Safari/537.36")

Les valeurs suivantes peuvent être retournées dans les propriétés *status* et *statusText* de l'objet *__STATUS* en cas d'erreur :

status	statusText	Commentaire
2	"Stamp has changed"	La valeur du marqueur interne (stamp) de l'entité ne correspond pas à celle de l'entité stockée dans les données (verrouillage optimiste).
3	"Already locked"	L'entité est verrouillée par un verrou pessimiste.
4	"Other error"	Une erreur critique peut être une erreur de bas niveau de la base de données (ex. clé dupliquée), une erreur matérielle, etc.
5	"Entity does not exist anymore"	L'entité n'existe plus dans les données.

Exemple

Nous verrouillons une entité dans un premier navigateur :

```
GET /rest/Customers(1)/?$lock=true
```

Réponse :

```
{  
    "result": true,  
    "__STATUS": {  
        "success": true  
    }  
}
```

Dans un second navigateur (autre session), nous envoyons la même requête.

Réponse :

```
{  
    "result":false,  
    "__STATUS":{  
        "status":3,  
        "statusText":"Already Locked",  
        "lockKind":7,  
        "lockKindText":"Locked By Session",  
        "lockInfo":{  
            "host":"127.0.0.1:8043",  
            "IPAddr":"127.0.0.1",  
            "recordNumber": 7,  
            "userAgent": """Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_3) AppleWebKit/537.36..."  
        }  
    }  
}
```

\$method

Ce paramètre vous permet de définir l'opération à exécuter avec l'entité ou la sélection d'entité (entity selection) retournée.

Syntaxe

Syntaxe	Exemple	Description
<code>\$method=delete</code>	<code>POST /Employee?\$filter="ID=11"&\$method=delete</code>	Supprime l'entité, collection d'entités ou sélection d'entité courante
<code>\$method=entityset</code>	<code>GET /People/?\$filter="ID>320"&\$method=entityset&\$timeout=600</code>	Crée un ensemble d'entités dans le cache de 4D Server basé sur la collection d'entités définies dans la requête REST
<code>\$method=release</code>	<code>GET /Employee/\$entityset/<entitySetID>?&\$method=release</code>	Affiche un ensemble d'entités existant stocké dans le cache de 4D Server
<code>\$method=subentityset</code>	<code>GET /Company(1)/staff?\$expand=staff&\$method=subentityset&\$subOrderby=lastName ASC</code>	Crée un ensemble d'entités basé sur la collection d'entités relatives définies dans la requête REST
<code>\$method=update</code>	<code>POST /Person/?\$method=update</code>	Met à jour et/ou crée une ou plusieurs entités

\$method=delete

Supprime l'entité, collection d'entités ou sélection d'entité courante (crée via REST)

Description

Avec `$method=delete`, vous pouvez supprimer une entité ou une collection d'entités entière. Vous pouvez définir la collection d'entités en utilisant, par exemple, `$filter` ou en spécifiant une directement à l'aide de `{dataClass}({key})` (par exemple, `/Employee(22)`).

Vous pouvez également supprimer les entités d'un ensemble d'entités en appelant `$entityset/{entitySetID}`.

Exemple

Vous pouvez ensuite saisir la requête REST suivante pour supprimer l'entité dont la clé porte le numéro 22 :

```
POST /rest/Employee(22)/?$method=delete
```

Vous pouvez également faire une requête en utilisant `$ filter` :

```
POST /rest/Employee?$filter="ID=11"&$method=delete
```

Vous pouvez également supprimer un ensemble d'entités utilisant `$entityset/{entitySetID}` :

```
POST /rest/Employee/$entityset/73F46BE3A0734EAA9A33CA8B14433570?&$method=delete
```

Réponse :

```
{  
    "ok": true  
}
```

\$method=entityset

Crée un ensemble d'entités dans le cache de 4D Server basé sur la collection d'entités définies dans la requête REST

Description

Lorsque vous créez une collection d'entités dans REST, vous pouvez également créer un ensemble d'entités qui sera enregistré dans le cache de 4D Server. L'ensemble d'entités aura un numéro de référence que vous pouvez passer à `$entityset/{entitySetID}` pour y accéder. Par défaut, il est valable deux heures; vous pouvez toutefois modifier cette durée en passant une valeur (en secondes) à `$timeout`.

Si vous avez utilisé `$savedfilter` et/ou `$savedorderby` (avec `$filter` et/ou `$orderby`) lors de la création de votre ensemble d'entités, vous pouvez le recréer avec le même ID de référence même s'il a été supprimé du cache de 4D Server.

Exemple

Pour créer un ensemble d'entités, qui sera enregistré dans le cache de 4D Server pendant deux heures, ajoutez `$method=entityset` à la fin de votre requête REST :

```
GET /rest/People/?$filter="ID>320"&$method=entityset
```

Vous pouvez créer un ensemble d'entités qui sera stocké dans le cache de 4D Server pendant seulement dix minutes en passant un nouveau timeout à `$timeout` :

```
GET /rest/People/?$filter="ID>320"&$method=entityset&$timeout=600
```

Vous pouvez également enregistrer le filtre et trier, en passant true à `$savedfilter` et `$savedorderby`.

`$skip` et `$top/$limit` ne sont pas pris en compte lors de l'enregistrement d'un ensemble d'entités.

Après avoir créé un ensemble d'entités, le premier élément, `__ENTITYSET` est ajouté à l'objet retourné et indique l'URI à utiliser pour accéder à l'ensemble d'entités :

```
__ENTITYSET: "http://127.0.0.1:8081/rest/Employee/$entityset/9718A30BF61343C796345F3BE5B01CE7"
```

\$method=release

Affiche un ensemble d'entités existant stocké dans le cache de 4D Server.

Description

Vous pouvez afficher un ensemble d'entités, que vous avez créé à l'aide de `$method=entityset`, à partir du cache de 4D Server.

Exemple

Affiche un ensemble d'entités existant :

```
GET /rest/Employee/$entityset/4C51204DD8184B65AC7D79F09A077F24?&$method=release
```

Réponse :

Si la requête a abouti, la réponse suivante est retournée :

```

{
  "ok": true
}
Si l'entite n'as pas été trouvée, une erreur est retournée :

{
  "__ERROR": [
    {
      "message": "Error code: 1802\nEntitySet \\"4C51204DD8184B65AC7D79F09A077F24\\" cannot be fou
      "componentSignature": "dbmg",
      "errCode": 1802
    }
  ]
}

```

\$method=subentityset

Crée un ensemble d'entités dans le cache de 4D Server basé sur la collection d'entités relatives définies dans la requête REST

Description

`$method=subentityset` vous permet de trier les données retournées par l'attribut relationnel défini dans la requête REST.

Pour trier les données, utilisez la propriété `$sub0rderby`. Pour chaque attribut, définissez l'ordre sur ASC (ou asc) pour l'ordre croissant et sur DESC (desc) pour l'ordre décroissant. Par défaut, les données sont triées par ordre croissant.

Si vous souhaitez spécifier plusieurs attributs, vous pouvez les délimiter avec une virgule, µ, `$sub0rderby="lastName desc, firstName asc"`.

Exemple

Si vous souhaitez récupérer uniquement les entités relatives pour une entité spécifique, vous pouvez lancer la requête REST suivante, dans laquelle "staff" est l'attribut relationnel dans la dataclass "Company" liée à la dataclass "Employee":

```
GET /rest/Company(1)/staff?$expand=staff&$method=subentityset&$sub0rderby=lastName ASC
```

Réponse :

```

{
    "__ENTITYSET": "/rest/Employee/$entityset/FF625844008E430B9862E5FD41C741AB",
    "__entityModel": "Employee",
    "__COUNT": 2,
    "__SENT": 2,
    "__FIRST": 0,
    "__ENTITIES": [
        {
            "__KEY": "4",
            "__STAMP": 1,
            "ID": 4,
            "firstName": "Linda",
            "lastName": "Jones",
            "birthday": "1970-10-05T14:23:00Z",
            "employer": {
                "__deferred": {
                    "uri": "/rest/Company(1)",
                    "__KEY": "1"
                }
            }
        },
        {
            "__KEY": "1",
            "__STAMP": 3,
            "ID": 1,
            "firstName": "John",
            "lastName": "Smith",
            "birthday": "1985-11-01T15:23:00Z",
            "employer": {
                "__deferred": {
                    "uri": "/rest/Company(1)",
                    "__KEY": "1"
                }
            }
        }
    ]
}

```

\$method=update

Met à jour et/ou crée une ou plusieurs entités

Description

`$method=update` vous permet de mettre à jour et/ou de créer une ou plusieurs entités dans un seul POST. Si vous mettez à jour et/ou créez une entité, cela s'effectue dans un objet avec, pour chaque propriété, un attribut et sa valeur, par exemple `{lastName: "Smith"}`. Si vous mettez à jour et/ou créez plusieurs entités, vous devez créer une collection d'objets.

In any cases, you must set the POST data in the body of the request.

Pour mettre à jour une entité, vous devez passer les paramètres `__KEY` et `__STAMP` dans l'objet avec tous les attributs modifiés. Si ces deux paramètres sont manquants, une entité sera ajoutée avec les valeurs de l'objet que vous envoyez dans le corps de votre POST.

Triggers are executed immediately when saving the entity to the server. La réponse contient toutes les données telles qu'elles existent sur le serveur.

Vous pouvez également placer ces requêtes pour créer ou mettre à jour des entités dans une transaction en appelant `$atomic/$atonce`. Si des erreurs se produisent lors de la validation des données, aucune des entités n'est

sauvegardée. Vous pouvez également utiliser `$method=validate` pour valider les entités avant de les créer ou de les mettre à jour.

Si un problème survient lors de l'ajout ou de la modification d'une entité, une erreur vous sera retournée avec ces informations.

A noter pour les types d'attributs spécifiques :

- Les dates doivent être exprimées au format JS : YYYY-MM-DDTHH:MM:SSZ (par exemple, "2010-10-05T23:00:00Z"). Si vous avez sélectionné la propriété Date uniquement pour votre attribut Date, le fuseau horaire et l'heure (heure, minutes et secondes) seront supprimés. Dans ce cas, vous pouvez également envoyer la date au format qui vous est retourné dd!mm!yyyy (par exemple, 05!10!2013).
- Les valeurs des booléens sont vrai ou faux.
- Les fichiers téléchargés à l'aide de `$upload` peuvent s'appliquer à un attribut de type Image ou BLOB en passant l'objet retourné au format suivant {"ID": "D507BC03E613487E9B4C2F6A0512FE50"}

Exemple

Pour mettre à jour une entité spécifique, utilisez l'URL suivante :

```
POST /rest/Person/?$method=update
```

Données POST :

```
{
  __KEY: "340",
  __STAMP: 2,
  firstName: "Pete",
  lastName: "Miller"
}
```

Les attributs `firstName` et `lastName` de l'entité indiquée ci-dessus seront modifiés en laissant inchangés tous les autres attributs (sauf ceux calculés sur la base de ces attributs).

Si vous souhaitez créer une entité, vous pouvez envoyer, via POST, les attributs à l'aide de cette URL :

```
POST /rest/Person/?$method=update
```

Données POST :

```
{
  firstName: "John",
  lastName: "Smith"
}
```

Vous pouvez également créer et mettre à jour plusieurs entités en même temps en utilisant la même URL ci-dessus en passant plusieurs objets d'un tableau au POST :

```
POST /rest/Person/?$method=update
```

Données POST :

```
[{
    "__KEY": "309",
    "__STAMP": 5,
    "ID": "309",
    "firstName": "Penelope",
    "lastName": "Miller"
}, {
    "firstName": "Ann",
    "lastName": "Jones"
}]
```

Réponse :

Lorsque vous ajoutez ou modifiez une entité, elle vous est retournée avec les attributs qui ont été modifiés. Par exemple, si vous créez le nouvel employé ci-dessus, les informations suivantes seront renvoyées :

```
{
    "__KEY": "622",
    "__STAMP": 1,
    "uri": "http://127.0.0.1:8081/rest/Employee(622)",
    "__TIMESTAMP": "!!2020-04-03!!",
    "ID": 622,
    "firstName": "John",
    "firstName": "Smith"
}
```

Si, par exemple, le tampon n'est pas correct, l'erreur suivante est retournée :

```
{
    "__STATUS": {
        "status": 2,
        "statusText": "Stamp has changed",
        "success": false
    },
    "__KEY": "1",
    "__STAMP": 12,
    "__TIMESTAMP": "!!2020-03-31!!",
    "ID": 1,
    "firstname": "Denise",
    "lastname": "O'Peters",
    "isWoman": true,
    "numberOfKids": 1,
    "addressID": 1,
    "gender": true,
    "imageAtt": {
        "__deferred": {
            "uri": "/rest/Persons(1)/imageAtt?$imageformat=best&$version=12&$expand=imageAtt",
            "image": true
        }
    },
    "extra": {
        "num": 1,
        "alpha": "I am 1"
    },
    "address": {
        "__deferred": {
            "uri": "/rest/Address(1)",
            "__KEY": "1"
        }
    },
    "__ERROR": [
        {
            "message": "Given stamp does not match current one for record# 0 of table Persons",
            "componentSignature": "dbmg",
            "errCode": 1263
        },
        {
            "message": "Cannot save record 0 in table Persons of database remote_dataStore",
            "componentSignature": "dbmg",
            "errCode": 1046
        },
        {
            "message": "The entity# 1 in the \"Persons\" dataclass cannot be saved",
            "componentSignature": "dbmg",
            "errCode": 1517
        }
    ]
}{}}
}
```

\$orderby

Trie les données retournées par l'attribut et l'ordre de tri définis (par exemple, `$orderby="lastName desc, salaire asc"`)

Description

`$orderby` ordonne les entités retournées par la requête REST. Pour chaque attribut, définissez l'ordre sur `ASC` (ou `asc`) pour l'ordre croissant et sur `DESC` (`desc`) pour l'ordre décroissant. Par défaut, les données sont triées par ordre croissant. Par défaut, les données sont triées par ordre croissant.

Exemple

Dans cet exemple, nous récupérons les entités et les trions en même temps :

```
GET /rest/Employee/?$filter="salary!=0"&$orderby="salary DESC, lastName ASC, firstName ASC"
```

L'exemple ci-dessous trie l'entité définie par l'attribut `lastName` dans l'ordre croissant :

```
GET /rest/Employee/$entityset/CB1BCC603DB0416D939B4ED379277F02?$orderby="lastName"
```

Résultat :

```
{
  __entityModel: "Employee",
  __COUNT: 10,
  __SENT: 10,
  __FIRST: 0,
  __ENTITIES: [
    {
      __KEY: "1",
      __STAMP: 1,
      firstName: "John",
      lastName: "Smith",
      salary: 90000
    },
    {
      __KEY: "2",
      __STAMP: 2,
      firstName: "Susan",
      lastName: "O'Leary",
      salary: 80000
    },
    // plus d'entités
  ]
}
```

\$querypath

Retourne la requête telle qu'elle a été exécutée par 4D Server (par exemple, `$querypath=true`)

Description

`$querypath` retourne la requête telle qu'elle a été exécutée par 4D Server. Si, par exemple, une partie de la requête passée ne retourne aucune entité, le reste de la requête n'est pas exécuté. La requête lancée est optimisée, comme vous pouvez le voir dans ce `$querypath`.

Pour plus d'informations sur les chemins de requête, reportez-vous à [queryPlan ete queryPath](#).

Dans la collection d'étapes, il existe un objet avec les propriétés suivantes qui définissent la requête exécutée :

Propriété	Type	Description
description	Chaine	Requête exécutée ou "AND" lorsqu'il existe plusieurs étapes
time	Nombre	Nombre de millisecondes nécessaires pour exécuter la requête
recordsfounds	Nombre	Nombre d'enregistrements trouvés
steps	Collection	Une collection avec un objet définissant l'étape suivante du chemin de la requête

Exemple

Si vous exécutez la requête suivante :

```
GET /rest/Employee/$filter="employer.name=acme AND lastName=Jones"&$querypath=true
```

Et si aucune entité n'a été trouvée, le chemin de la requête suivant sera retourné si vous saisissez ce qui suit :

```
GET /rest/$querypath
```

Réponse :

```

__queryPath: {

    steps: [
        {
            description: "AND",
            time: 0,
            recordsfounds: 0,
            steps: [
                {
                    description: "Join on Table : Company : People.employer = Company.ID",
                    time: 0,
                    recordsfounds: 0,
                    steps: [
                        {
                            steps: [
                                {
                                    description: "Company.name = acme",
                                    time: 0,
                                    recordsfounds: 0
                                }
                            ]
                        }
                    ]
                }
            ]
        }
    ]
}

```

En revanche, si la première requête retourne plus d'une entité, la seconde sera exécutée. Si nous exécutons la requête suivante :

```
GET /rest/Employee/$filter="employer.name=a* AND lastName!=smith"&$querypath=true
```

Si au moins une entité a été trouvée, le chemin de la requête suivant sera retourné si vous saisissez ce qui suit :

```
GET /rest/$querypath
```

Réponse :

```
"__queryPath": {
  "steps": [
    {
      "description": "AND",
      "time": 1,
      "recordsfounds": 4,
      "steps": [
        {
          "description": "Join on Table : Company : Employee.employer = Company.ID",
          "time": 1,
          "recordsfounds": 4,
          "steps": [
            {
              "steps": [
                {
                  "description": "Company.name LIKE a*",
                  "time": 0,
                  "recordsfounds": 2
                }
              ]
            }
          ]
        },
        {
          "description": "Employee.lastName # smith",
          "time": 0,
          "recordsfounds": 4
        }
      ]
    }
  ]
}
```

\$queryplan

Retourne la requête telle qu'elle a été passée au 4D Server (par exemple, `$queryplan=true`)

Description

\$queryplan retourne le plan de la requête telle qu'il a été exécuté par 4D Server.

Propriété	Type	Description
item	Chaine	Requête exécutée
subquery	Tableau	S'il existe une sous-requête, un objet supplémentaire contenant une propriété d'élément (comme celle indiquée ci-dessus)

Pour plus d'informations sur les plans de requête, reportez-vous à [queryPlan ete queryPath](#).

Exemple

Si vous passez la requête suivante :

```
GET /rest/People/$filter="employer.name=acme AND lastName=Jones"&$queryplan=true
```

Réponse :

```
__queryPlan: {
    And: [
        {
            item: "Join on Table : Company : People.employer = Company.ID",
            subquery: [
                {
                    item: "Company.name = acme"
                }
            ]
        },
        {
            item: "People.lastName = Jones"
        }
    ]
}
```

\$savedfilter

Enregistre le filtre défini par \$filter lors de la création d'un ensemble d'entités (par exemple, `$savedfilter=""{filter}"`)

Description

Lorsque vous créez un ensemble d'entités, vous pouvez, par sécurité, enregistrer le filtre utilisé pour sa création. Si l'ensemble d'entités que vous avez créé est supprimé du cache de 4D Server (en raison du timeout, du besoin d'espace sur le serveur ou de la suppression après avoir appelé `$method=release`).

Utilisez `$savedfilter` pour enregistrer le filtre que vous avez défini lors de la création de votre ensemble d'entités, puis passez `$savedfilter` avec votre appel, pour récupérer à chaque fois l'ensemble d'entités.

Si l'ensemble d'entités n'est plus dans le cache de 4D Server, il sera recréé avec un nouveau timeout de 10 minutes. L'ensemble d'entités sera actualisé (certaines entités peuvent être incluses tandis que d'autres peuvent être supprimées) depuis la dernière fois qu'il a été créé, s'il n'existe plus avant de le recréer.

Si vous avez utilisé à la fois `$savedfilter` et

Exemple

`$savedorderby` dans votre appel lors de la création d'un ensemble d'entités et que vous en omettez un, le nouvel ensemble d'entités, qui aura le même numéro de référence, le reflétera.

```
GET /rest/People/?$filter="employer.name=Apple"&$savedfilter="employer.name=Apple"&$method=entityset
```

Puis, lorsque vous accédez à votre ensemble d'entités, saisissez ce qui suit pour vous assurer que l'ensemble d'entités est toujours valide :

```
GET /rest/People/$entityset/AEA452C2668B4F6E98B6FD2A1ED4A5A8?&$savedfilter="employer.name=Apple"
```

\$savedorderby

Enregistre le tri défini par `$orderby` lors de la création d'un ensemble d'entités (par exemple, `$savedorderby="{$orderby}"`)

Description

Lorsque vous créez un ensemble d'entités, vous pouvez, par sécurité, enregistrer l'ordre de tri et le filtre utilisés pour sa création. Si l'ensemble d'entités que vous avez créé est supprimé du cache de 4D Server (en raison du timeout, du besoin d'espace sur le serveur ou de la suppression après avoir appelé `$method=release`).

Utilisez `$savedorderby` pour enregistrer l'ordre que vous avez défini lors de la création de votre ensemble d'entités, puis passez `$savedorderby` avec votre appel, pour récupérer à chaque fois l'ensemble d'entités.

Si l'ensemble d'entités n'est plus dans le cache de 4D Server, il sera recréé avec un nouveau timeout de 10 minutes. Si vous avez utilisé à la fois `$savedfilter` et `$savedorderby` dans votre appel lors de la création d'un ensemble d'entités et que vous en omettez un, le nouvel ensemble d'entités, qui aura le même numéro de référence, le reflétera.

Exemple

Appelez d'abord `$savedorderby`, dans l'appel initial, pour créer un ensemble d'entités :

```
GET /rest/People/?  
$filter="lastName!=""&$savedfilter="lastName!=""&$orderby="salary"&$savedorderby="salary"&$method=entity  
set
```

Ensuite, lorsque vous accédez à votre ensemble d'entités, écrivez ce qui suit (en utilisant à la fois `$savedfilter` et `$savedorderby`) pour vous assurer que le filtre et son ordre de tri existent toujours :

```
GET /rest/People/$entityset/AEA452C2668B4F6E98B6FD2A1ED4A5A8?  
$savedfilter="lastName!=""&$savedorderby="salary"
```

\$skip

Démarre l'entité définie par ce numéro dans la collection (par exemple, `$skip=10`)

Description

`$skip` définit l'entité de la collection par laquelle commencer. Par défaut, la collection envoyée commence par la première entité. Pour commencer avec la 10e entité de la collection, passez 10.

`$skip` est généralement utilisé avec

Exemple

`$top/$limit` pour naviguer dans une entity collection.

```
GET /rest/Employee/$entityset/CB1BCC603DB0416D939B4ED379277F02?$skip=20
```

\$timeout

Définit le nombre de secondes pour enregistrer un ensemble d'entités dans le cache de 4D Server (par exemple, `$timeout=1800`)

Description

`$method=entityset`, passez le nombre de secondes à `$timeout`. Par exemple, si vous souhaitez définir le timeout sur 20 minutes, passez 1200. Par défaut, le timeout est de deux (2) heures.

Une fois le timeout défini, chaque fois qu'un ensemble d'entités est appelé (via `$method=entityset`), le timeout est recalculé en fonction de l'heure courante et du timeout.

Une fois le timeout défini, chaque fois qu'un ensemble d'entités est appelé (via `$method=entityset`), le timeout est recalculé en fonction de l'heure courante et du timeout.

Exemple

Si un ensemble d'entités est supprimé puis recréé à l'aide de `$method=entityset` avec `$savedfilter`, le nouveau timeout par défaut est de 10 minutes, quel que soit le timeout que vous avez défini lors de l'appel de `$timeout`.

```
GET /rest/Employee/?$filter="salary!=0"&$method=entityset&$timeout=1200
```

\$top/\$limit

Limite le nombre d'entités à retourner (par exemple, `$top=50`)

Description

`$top/$limit` définit la limite des entités à retourner. Par défaut, leur nombre est limité à 100. Vous pouvez utiliser l'un des mots clés suivant : `$top` ou `$limit` .

Lorsqu'il est utilisé avec `$skip` , vous pouvez parcourir la sélection d'entités retournée par la requête REST.

Exemple

Dans l'exemple suivant, nous recherchons les dix entités qui suivent la 20e entité :

```
GET /rest/Employee/$entityset/CB1BCC603DB0416D939B4ED379277F02?$skip=20&$top=10
```

\$version

Numéro de version de l'image

Description

`$version` est le numéro de version de l'image retourné par le serveur. Le numéro de version, qui est envoyé par le serveur, fonctionne autour du cache du navigateur afin que vous soyez sûr de récupérer la bonne image.

La valeur du paramètre de version de l'image est modifiée par le serveur.

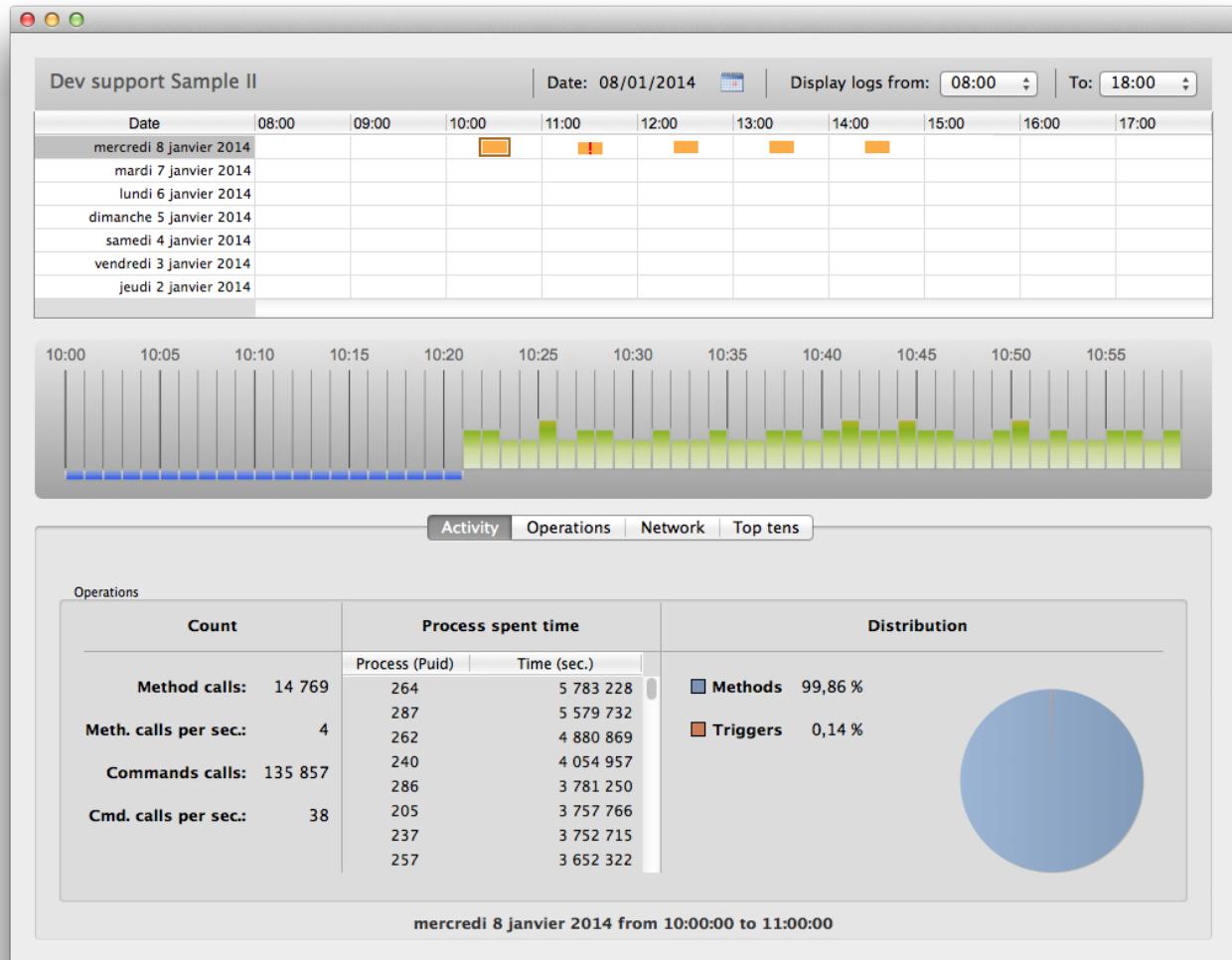
Exemple

L'exemple suivant définit le format d'image au format JPEG, quel que soit le véritable type de la photo et passe le véritable numéro de version envoyé par le serveur :

```
GET /rest/Employee(1)/photo?$imageformat=jpeg&$version=3&$expand=photo
```

A propos des formulaires 4D

Les formulaires fournissent l'interface par laquelle les informations sont saisies, modifiées et imprimées dans une application de bureau. A l'aide des formulaires, les utilisateurs peuvent interagir avec les données d'une base de données et imprimer des rapports. Les formulaires permettent de créer des boîtes de dialogue personnalisées, des palettes ou toute fenêtre personnalisée.



Les formulaires peuvent également contenir d'autres formulaires grâce aux fonctionnalités suivantes :

- [les objets sous-formulaires](#)
- [les formulaires hérités](#)

Création de formulaires

Vous pouvez ajouter ou modifier des formulaires 4D à l'aide des éléments suivants :

- L'interface 4D Developer : Créez de nouveaux formulaires à partir du menu Fichier ou de la fenêtre de l'Explorateur.
- L'éditeur de formulaires : Modifiez vos formulaires à l'aide de l'[éditeur de formulaires](#).
- Le code JSON : Créez et concevez vos formulaires à l'aide de JSON et enregistrez les fichiers de formulaire à l'[emplacement approprié](#). Exemple :

```
{
    "windowTitle": "Hello World",
    "windowMinWidth": 220,
    "windowMinHeight": 80,
    "method": "HWexample",
    "pages": [
        null,
        {
            "objects": {
                "text": {
                    "type": "text",
                    "text": "Hello World!",
                    "textAlign": "center",
                    "left": 50,
                    "top": 120,
                    "width": 120,
                    "height": 80
                },
                "image": {
                    "type": "picture",
                    "pictureFormat": "scaled",
                    "picture": "/RESOURCES/Images/HW.png",
                    "alignment": "center",
                    "left": 70,
                    "top": 20,
                    "width": 75,
                    "height": 75
                },
                "button": {
                    "type": "button",
                    "text": "OK",
                    "action": "Cancel",
                    "left": 60,
                    "top": 160,
                    "width": 100,
                    "height": 20
                }
            }
        }
    ]
}
```

Formulaire projet et formulaire table

Il existe deux catégories de formulaires :

- Les formulaires projet - Formulaires indépendants qui ne sont rattachés à aucune table. Ils sont destinés plus particulièrement à la création de boîtes de dialogue d'interface et de composants. Les formulaires projet peuvent être utilisés pour créer des interfaces facilement conformes aux normes du système d'exploitation.
- Les formulaires table - Rattachés à des tables spécifiques et bénéficient ainsi de fonctions automatiques utiles pour développer des applications basées sur des bases de données. En règle générale, une table possède des formulaires d'entrée et de sortie séparés.

En règle générale, vous sélectionnez la catégorie de formulaire lorsque vous créez le formulaire, mais vous pouvez la modifier par la suite.

Pages formulaire

Chaque formulaire est composé d'au moins deux pages :

- une page 1 : une page principale, affichée par défaut
- une page 0 : une page de fond, dont le contenu est affiché sur une page sur deux.

Vous pouvez créer plusieurs pages pour un formulaire d'entrée. Si le nombre de champs ou de variables est supérieur au nombre maximal supporté sur un écran, vous pouvez créer des pages supplémentaires pour les afficher. Plusieurs pages vous permettent d'effectuer les opérations suivantes :

- Placez les informations les plus importantes sur la première page et les informations les moins importantes sur les autres pages.
- Organisez chaque sujet sur sa propre page.
- Réduire ou éliminer le défilement pendant la saisie des données en définissant [l'ordre de saisie](#).
- Prévoyez de l'espace autour des éléments du formulaire pour un design d'écran attrayant.

Les pages multiples sont utiles uniquement pour les formulaires d'entrée. Elles ne sont pas destinées à être imprimées. Lorsqu'un formulaire de plusieurs pages est imprimé, seule la première page est imprimée.

Il n'y a aucune restriction sur le nombre de pages qu'un formulaire peut contenir. Le même champ peut apparaître en un nombre de fois illimité dans un formulaire et sur autant de pages que vous le souhaitez. Toutefois, plus vous aurez de pages dans un formulaire, plus il sera long à afficher.

Un formulaire multi-pages contient à la fois une page d'arrière-plan et plusieurs pages d'affichage. Les objets placés sur la page d'arrière-plan peuvent être visibles sur toutes les pages d'affichage, mais il ne peuvent être sélectionnés et modifiés que sur la page d'arrière-plan. Dans les formulaires multi-pages, vous devez placer votre palette de boutons sur la page d'arrière-plan. Vous devez également inclure un ou plusieurs objets sur la page d'arrière-plan qui fournissent à l'utilisateur des outils de navigation de page.

Formulaires hérités

Les formulaires 4D peuvent utiliser et être utilisés comme «formulaires hérités», ce qui signifie que tous les objets du *Formulaire A* peuvent être utilisés dans le *Formulaire B*. Dans ce cas, *Formulaire B* "hérite" des objets du *Formulaire A*.

Les références à un formulaire hérité est toujours active : si un élément d'un formulaire hérité est modifié (par exemple le style des boutons), tous les formulaires qui l'utilisent seront automatiquement modifiés.

Tous les formulaires (formulaires table et formulaires projet) peuvent être désignés comme un formulaire hérité. Cependant, les éléments qu'ils contiennent doivent être compatibles avec une utilisation dans différentes tables de base de données.

A l'exécution du formulaire, les objets sont chargés et combinés dans l'ordre suivant :

1. Page zéro du formulaire hérité
2. Page 1 du formulaire hérité
3. Page zéro du formulaire ouvert
4. Page courante du formulaire ouvert.

Cet ordre détermine [l'ordre de saisie](#) par défaut des objets dans le formulaire.

Seules les pages 0 et 1 du formulaire hérité peuvent apparaître dans les autres formulaires.

Les propriétés ainsi que la méthode d'un formulaire ne sont pas prises en compte lorsque celui-ci est utilisé comme formulaire hérité. En revanche, les méthodes des objets qu'il contient sont appelées.

Pour définir un formulaire hérité, les propriétés de [Inherited Form Name](#) et [Inherited Form Table](#) (pour les formulaires table) doivent être définies dans le formulaire qui héritera de quelque chose issue d'un autre formulaire.

Un formulaire peut hériter d'un formulaire projet, en définissant la propriété [Inherited Form Table](#) sur <None> dans la liste des propriétés (ou " " dans JSON).

Pour stopper l'héritage d'un formulaire, choisissez l'option <None> dans la Liste des propriétés (ou " " dans JSON) pour la propriété [Inherited Form Name](#).

Il est possible de définir un formulaire hérité dans un formulaire qui servira à son tour de formulaire hérité pour un troisième formulaire. La combinaison des objets s'effectue alors de manière récursive. 4D détecte les boucles

récurseuses (par exemple si le formulaire [table1]form1 est défini comme formulaire hérité de [table1]form1, c'est-à-dire de lui-même) et interrompt le chaînage des formulaires.

Propriétés prises en charge

[Associated Menu Bar](#) - [Fixed Height](#) - [Fixed Width](#) - [Form Break](#) - [Form Detail](#) - [Form Footer](#) - [Form Header](#) - [Form Name](#) - [Form Type](#) - [Inherited Form Name](#) - [Inherited Form Table](#) - [Maximum Height](#) - [Maximum Width](#) - [Method](#) - [Minimum Height](#) - [Minimum Width](#) - [Pages](#) - [Print Settings](#) - [Published as Subform](#) - [Save Geometry](#) - [Window Title](#)

Éditeur de formulaire

4D propose un éditeur de formulaires très complet qui vous permet de modifier votre formulaire jusqu'à ce que vous ayez atteint le résultat escompté. Dans l'éditeur de formulaires, vous pouvez créer et supprimer des objets, manipuler directement des objets et définir les propriétés des objets et des formulaires.

Interface

L'éditeur de formulaires affiche chaque formulaire JSON ouvert dans sa propre fenêtre, dotée d'une barre d'outils et d'une barre d'objets. Vous pouvez ouvrir plusieurs formulaires en même temps.

Display options

You can show or hide several interface elements on the current page of the form:

- Inherited Form: Inherited form objects (if there is an [inherited form](#)).
- Page 0: Objects from [page 0](#). Cette option vous permet de mieux visualiser et distinguer les objets de la page courante et ceux de la page 0.
- Paper: Borders of the printing page, which are shown as gray lines. This element can only be displayed by default in ["for printing" type forms](#).
- Rulers: Rulers of the Form editor's window.
- Markers: Output control lines and associated markers that show the limits of the form's different areas. This element can only be displayed by default in [list forms](#).
- Marker Labels: Marker labels, available only when the output control lines are displayed. This element can only be displayed by default in [list forms](#).
- Limits: Form's limits. Lorsque cette option est sélectionnée, le formulaire est affiché dans l'éditeur tel qu'il apparaîtra en mode Application. Cette possibilité est particulièrement intéressante pour ajuster un formulaire sans devoir tester le mode Application pour visualiser le résultat.

The [Size Based on](#), [Hor. margin](#) and [Vert. margin](#) settings of the form properties affect the form's limits.

Les limites du formulaire sont calculées en fonction des objets qui le composent. Lorsque vous déplacez ou agrandissez un objet placé près de la limite d'un formulaire, le rectangle de délimitation est modifié en conséquence.

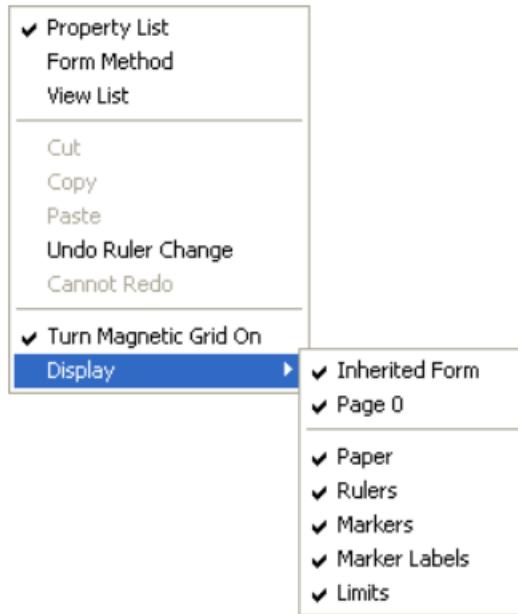
Default display

When a form is opened in the editor, interface elements are displayed or hidden by default, depending on:

- the New form default display options set in the Preferences - unchecked options cannot be displayed by default.
- the current [form type](#):
 - Markers and marker labels are always displayed by default on list forms
 - Paper is displayed by default on "for printing" forms.

Display/Hide elements

You can display or hide elements at any moment in the Form editor's current window by selecting [Display](#) from the Form menu or the Form editor's context menu:



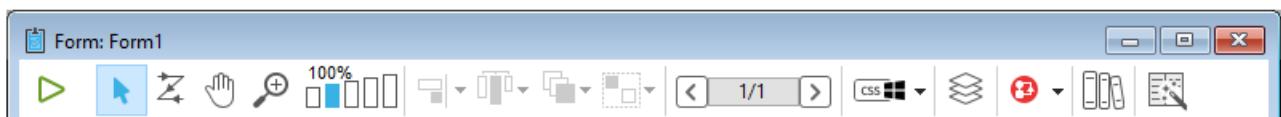
Rulers

Les règles situées sur le côté et en bas de cette fenêtre ont pour but de vous aider à placer les objets dans le formulaire. They can be [displayed or hidden](#).

Select Ruler definition... from the Form menu to change measurement units so that the form displays inches, centimeters, or pixels.

Barre d'outils

La barre d'outils de l'éditeur de formulaires propose un ensemble d'outils destinés à manipuler et modifier le formulaire. Chaque fenêtre dispose de sa propre barre d'outils.



La barre d'outils comporte les éléments suivants :

Icône	Nom	Description
▶	Exécuter le formulaire	Permet de tester l'exécution du formulaire. Lorsque vous cliquez sur ce bouton, 4D ouvre une nouvelle fenêtre et affiche le formulaire dans son contexte (liste d'enregistrements pour un formulaire liste et enregistrement courant en page pour un formulaire détaillé). Le formulaire est exécuté dans le process principal.
→	Flèche de sélection	Permet de sélectionner, déplacer et redimensionner les objets du formulaire. Note : Lorsqu'un objet de type Texte ou Zone de groupe est sélectionné, appuyer sur la touche Entrée permet de le passer en édition.
⇄	Ordre de saisie	Passe en mode "Ordre de saisie", dans lequel il est possible de visualiser et de modifier l'ordre de saisie courant du formulaire. A noter que vous pouvez également visualiser l'ordre de saisie courant tout en travaillant dans le formulaire.
👉	Déplacement	Passe en mode "Déplacement", dans lequel il est possible d'atteindre rapidement n'importe quelle partie du formulaire en le faisant directement glisser dans la fenêtre. Le curseur prend la forme d'une main. Ce mode de

Icône	Nom	Description
	Zoom	Permet de modifier l'échelle d'affichage du formulaire (100% par défaut). Vous pouvez passer en mode "Zoom" en cliquant sur le bouton loupe ou en cliquant directement sur la barre correspondant à l'échelle désirée. Cette fonction est détaillée dans le paragraphe précédent.
	Alignement	Ce bouton est associé à un menu permettant d'aligner les objets dans le formulaire. Il est activé (ou non) en fonction des objets sélectionnés. Disponible uniquement avec un aperçu CSS Aucun
	Répartition	Ce bouton est associé à un menu permettant de répartir les objets dans le formulaire. Il est activé (ou non) en fonction des objets sélectionnés. Disponible uniquement avec un aperçu CSS Aucun
	Niveau	Ce bouton est associé à un menu permettant de répartir les objets dans le formulaire. Il est activé (ou non) en fonction des objets sélectionnés.
	Groupement/Dégroupement	Ce bouton est associé à un menu permettant de grouper et dégrouper la sélection d'objets du formulaire. Il est activé (ou non) en fonction des objets sélectionnés.
	Affichage et gestion des pages	Cette zone permet de passer d'une page du formulaire à une autre et d'ajouter des pages. Pour naviguer parmi les pages du formulaire, cliquez sur les boutons fléchés ou cliquez sur la zone centrale et choisissez la page à afficher dans le menu qui apparaît. Si vous cliquez sur le bouton fléché de droite alors que vous êtes sur la dernière page du formulaire, 4D vous permet d'ajouter une page.
	CSS Preview	Ce bouton permet de sélectionner le mode CSS à utiliser.
	Gestion des vues	Ce bouton affiche ou masque alternativement la palette des vues. Cette fonction est détaillée dans la section Utiliser les vues d'objet.
	Affichage des badges	Chaque clic sur ce bouton provoque l'affichage successif de tous les types de badges de formulaire. Le bouton est également associé à un menu permettant de sélectionner directement le type de badge à afficher.
	Bibliothèque d'objets préconfigurés	Ce bouton affiche la fenêtre de la bibliothèque d'objets préconfigurée, proposant de nombreux objets auxquels des propriétés par défaut ont déjà été appliquées.
	Création de list box	Ce bouton crée de nouvelles list box de type entity selection.

Object bar

The object bar contains all the active and inactive objects that can be used in 4D forms. Some objects are grouped together by themes. Each theme includes several alternatives that you can choose between. When the object bar has the focus, you can select the buttons using the keys of the keyboard. The following table describes the object groups available and their associated shortcut key.

Bouton	Group	Key
	Text / Group Box	T
	Input	F
	Hierarchical List / List Box	L
	Combo Box / Drop-down List / Picture Pop-up Menu	P
	Button / Picture Button / Button Grid	B
	Bouton radio	R
	Case à cocher	C
	Progress Indicator / Ruler / Stepper / Spinner	I
	Rectangle / Line / Oval	S
	Splitter / Tab Control	D
	Plug-in Area / Subform / Web Area / 4D Write Pro / 4D View Pro	X

To draw an object type, select the corresponding button and then trace the object in the form. After creating an object, you can modify its type using the Property List. Hold down the Shift key as you draw to constrain the object to a regular shape. Lines are constrained to horizontal, 45°, or vertical, rectangles are constrained to squares, and ovals are constrained to circles.

The current variant of the theme is the object that will be inserted in the form. When you click the right side of a button, you access the variant menu:



You can click twice on the button so that it remains selected even after you have traced an object in the form (continual selection). This function makes creating several successive objects of the same type easier. To cancel a continual selection, click on another object or tool.

Liste de propriétés

Both forms and form objects have properties that control access to the form, the appearance of the form, and the behavior of the form when it is used. Form properties include, for example, the form's name, its menu bar, and its size. Object Properties include, for example, an object's name, its dimensions, its background color, and its font.

You can display and modify form and object properties using the Property List. It displays either form or objects properties depending on what you select in the editor window.

To display/hide the Property List, choose Property List from the Form menu or from the context menu of the Form editor. You can also display it by double-clicking in an empty area of the form.

Raccourcis de navigation

Vous pouvez naviguer dans la Liste des propriétés à l'aide des raccourcis suivants :

- Touches fléchées haut ou bas ↑ ↓ : déplacement de cellule en cellule.

- Touches fléchées gauche ou droite $\leftarrow \rightarrow$: déploie/contracte les thèmes ou les menus.
- PgUp et PgDn : sélectionne la première ou la dernière cellule visible de la liste affichée.
- Début et Fin : sélectionne la première ou la dernière cellule de la liste.
- Ctrl+clic (Windows) ou Commande+clic (Mac OS) sur un événement : sélectionne/désélectionne tous les événements, en fonction de l'état initial de l'événement sur lequel vous avez cliqué.
- Ctrl+clic (Windows) ou Commande+clic (Mac OS) sur un intitulé de thème : déploie/contracte tous les thèmes.

Manipulating Form Objects

Ajouter des objets

Vous pouvez ajouter des objets dans un formulaire de nombreuses manières :

- Par traçage d'un objet après sélection dans la barre d'objets (cf. paragraphe [Utiliser la barre d'objets](#))
- Par glisser-déposer depuis la barre d'objets
- Par glisser-déposer ou copier-coller depuis la [bibliothèque d'objets](#) préconfigurés
- Par glisser-déposer depuis un autre formulaire,
- Par glisser-déposer depuis l'Explorateur (champs) ou les éditeurs du mode Développement (énumérations, images, etc.)

Une fois l'objet inséré, vous pouvez modifier toutes ses caractéristiques dans l'éditeur de formulaires.

Vous pouvez travailler avec deux types d'objets dans vos formulaires :

- les objets statiques (filets, cadres, images d'arrière-plan, etc.) : ces objets sont généralement utilisés pour le décor, les libellés ou encore l'interface graphique. Ces objets sont accessibles via la barre d'objets de l'éditeur de formulaires. Vous pouvez définir leurs attributs graphiques (taille, couleur, police...) ainsi que leurs propriétés de redimensionnement à l'aide de la Liste de propriétés. A la différence des objets actifs, les objets statiques ne sont pas associés à des variables. A noter qu'il est possible d'insérer des éléments dynamiques dans les objets statiques.
- les objets actifs : un objet actif est un objet qui réalise une tâche ou une fonction de l'interface. Il existe de nombreux types d'objets actifs : champs, boutons, listes déroulantes, etc. Un objet actif est associé soit à un champ, soit à une variable.

Sélectionner des objets

Avant de pouvoir réaliser une opération sur un objet (comme le changement de l'épaisseur d'une ligne ou d'une police), il est nécessaire de sélectionner cet objet.

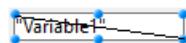
Pour sélectionner un objet à l'aide de la barre d'outils :

1. Cliquez sur l'outil Flèche dans la barre d'outils.



Lorsque vous le faites glisser au-dessus du formulaire, le pointeur prend la forme du pointeur standard.

2. Cliquez sur l'objet que vous souhaitez sélectionner. Des poignées de sélection identifient l'objet sélectionné.



Pour sélectionner un objet à l'aide de la Liste des propriétés :

1. Sélectionnez le nom de l'objet dans la liste de sélection située en haut de la palette.

De cette manière, vous pouvez sélectionner un objet masqué par d'autres objets ou situé en-dehors des limites de la fenêtre. Pour désélectionner un objet, cliquez hors de ses limites ou cliquez dessus en maintenant la touche Majuscule enfoncée.

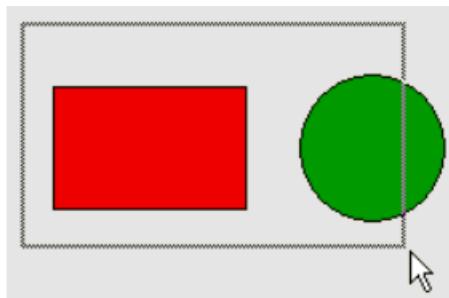
Il est également possible de sélectionner des objets en double-cliquant dans la fenêtre de résultat d'une recherche globale.

Selecting multiple objects

Il est souvent nécessaire de réaliser la même opération sur plusieurs objets d'un formulaire — par exemple, pour les déplacer, les aligner ou changer leur apparence. 4D vous permet de sélectionner plusieurs objets en même temps. Vous pouvez réaliser une sélection multiple en utilisant l'une des solutions suivantes :

- Choisissez Tout sélectionner dans le menu Edition.
- Cliquez avec le bouton droit de la souris sur un objet et choisissez la commande Sélectionner objets de même type dans le menu contextuel.
- Maintenez la touche Maj enfoncee et cliquez l'un après l'autre sur tous les objets que vous souhaitez sélectionner.
- Cliquez hors du groupe d'objets que vous souhaitez sélectionner et dessinez un rectangle de sélection entourant ou traversant les objets à sélectionner. Tout objet inclus dans les limites du rectangle ou qui touche ces limites est sélectionné lorsque vous relâchez le bouton de la souris.
- Maintenez enfoncee la touche Alt (sous Windows) ou Option (sous Mac Os) et tracez un rectangle de sélection. Dans ce cas, seuls les objets entièrement inclus dans ce rectangle seront sélectionnés.

La fenêtre ci-dessous représente la sélection de deux objets à l'aide d'un rectangle de sélection :



Pour désélectionner un objet qui fait partie d'un groupe d'objets sélectionnés, maintenez la touche Majuscule enfoncee et cliquez sur cet objet. Les autres objets demeurent sélectionnés. Pour désélectionner tous les objets, cliquez hors des limites de ces objets.

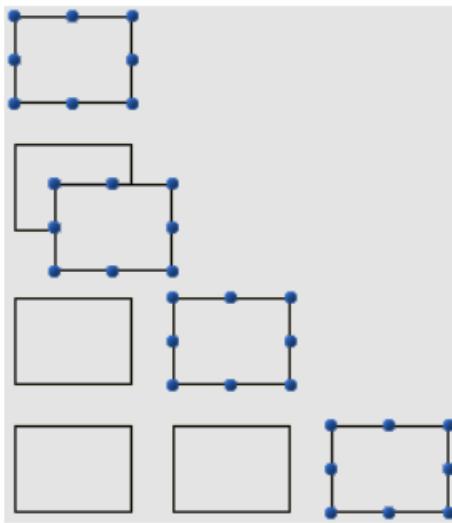
Duplicier des objets

Vous pouvez dupliquer tout objet de formulaire, y compris les objets actifs. Les copies d'objets actifs conservent toutes les propriétés de l'objet original comme le nom, le type, l'action automatique, le format d'affichage et la méthode objet.

Vous pouvez dupliquer directement un objet ou une sélection d'objets, ou utiliser la boîte de dialogue "Dupliquer plusieurs" pour paramétrer une duplication multiple d'objets. Cette boîte de dialogue vous permet de créer autant de dupliques d'un ou de plusieurs objets que vous voulez, en une seule opération.

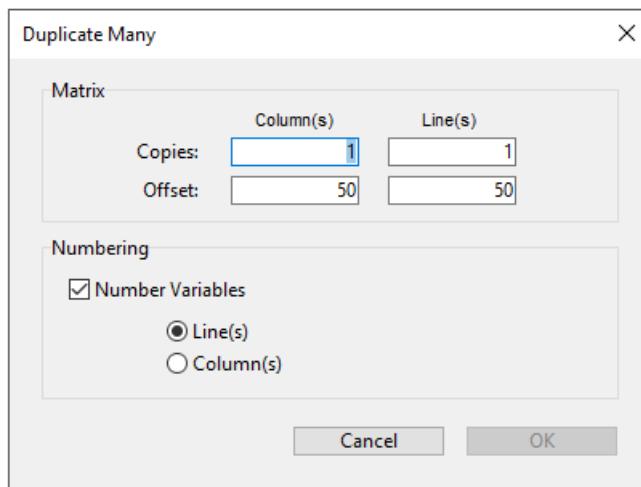
Pour dupliquer directement un ou plusieurs objet(s) :

1. Sélectionnez le ou les objet(s) que vous souhaitez dupliquer.
2. Choisissez la commande Dupliquer dans le menu Edition. 4D crée une copie de chaque objet sélectionné et place la copie juste à côté de l'original.
3. Déplacez la copie à l'emplacement souhaité. Si vous choisissez de nouveau la commande Dupliquer, 4D crée une autre copie pour chaque objet et la place exactement au même placement relatif par rapport à la première copie. Si vous devez répartir plusieurs copies d'un objet sur un axe, appliquez la procédure suivante. Dupliquez l'objet original, déplacez la copie à un autre emplacement sur le formulaire, puis dupliquez la copie. La deuxième copie adopte le même positionnement relatif par rapport à la première copie que celui qui existe entre la position de l'original et celle de la première copie. Les copies suivantes seront alors placées avec le même écart par rapport à leur original. Le schéma ci-dessous explique le fonctionnement du placement relatif des copies :



Dupliquer plusieurs

La boîte de dialogue “Dupliquer plusieurs” apparaît lorsque vous sélectionnez un ou plusieurs objet(s) puis choisissez la commande Dupliquer plusieurs... dans le menu Objets.



- Dans la zone supérieure, saisissez le nombre de colonnes et de lignes d'objets que vous souhaitez obtenir.

Par exemple, si vous voulez obtenir 3 colonnes et 2 lignes d'objets, saisissez 3 dans la zone Colonne(s) et 2 dans la zone Ligne(s). Si vous souhaitez ajouter horizontalement deux copies d'un objet, saisissez 3 dans la zone Colonnes (laissez la zone Ligne(s) à 1).

- Pour les lignes et les colonnes, définissez le décalage que vous souhaitez appliquer à chaque nouveau duplicata.

La valeur saisie doit être exprimée en points et sera appliquée relativement à l'origine de l'objet dupliqué.

Par exemple, si vous souhaitez laisser un intervalle vertical de 20 points entre chaque objet, et que la hauteur de l'objet source est de 50 points, saisissez 70 dans la zone “Intervalle” de Colonne.

- Si vous souhaitez créer une matrice de variables, cochez l'option Numéroter les variables et sélectionnez le sens dans lequel la numérotation des variables doit s'effectuer. Cette option n'est active que si l'objet sélectionné est une variable. Pour plus d'informations sur cette option, reportez-vous à la section Dupliquer sur matrice du *Manuel de développement*.

Déplacer des objets

Vous pouvez déplacer tout objet d'un formulaire, graphique ou actif, y compris les champs ou les objets créés à l'aide d'un modèle. Pour déplacer un objet, vous pouvez :

- Déplacer l'objet en le faisant glisser avec la souris,
- Déplacer l'objet pixel par pixel en utilisant les touches fléchées du clavier,
- Déplacer l'objet par paliers (de 20 pixels par défaut),

Lorsque vous commencez à déplacer un objet à l'aide de la souris, les poignées disparaissent. 4D affiche des marqueurs qui indiquent l'emplacement des limites de l'objet dans les règles, vous pouvez ainsi placer les objets avec précision. Prenez garde à ne pas cliquer sur les poignées, ce qui aurait pour effet de redimensionner l'objet. Appuyez sur la touche Majuscule pour effectuer un déplacement avec contrainte.

Si la [grille magnétique](#) est activée, le déplacement de l'objet s'effectue par paliers indiquant les emplacements remarquables.

Pour déplacer un objet pixel par pixel :

- Sélectionnez le ou les objet(s) que vous souhaitez déplacer. A chaque fois que vous appuyez sur une touche fléchée, la sélection est déplacée d'un pixel dans la direction de la flèche.

Pour déplacer l'objet par paliers :

- Sélectionnez le ou les objet(s) que vous souhaitez déplacer, appuyez sur la touche Majuscule et utilisez les touches fléchées du clavier pour déplacer l'objet par paliers. Par défaut, les paliers sont de 20 pixels. Vous pouvez modifier le pas dans la Page Formulaires des Préférences.

Grouper des objets

4D vous permet de grouper des objets de manière à ce que vous puissiez sélectionner, déplacer et modifier ce groupe comme un seul objet. Les objets qui sont groupés conservent leur position relative par rapport aux autres objets du groupe. Les objets groupés sont par exemple un champ et son libellé, un bouton invisible et son icône, etc.

Lorsque vous redimensionnez un groupe, tous les objets du groupe sont redimensionnés proportionnellement (hormis les zones de texte, qui sont redimensionnées par étape suivant leur taille de police de caractères).

Vous pouvez dégrouper un groupe d'objets à tout moment et les traiter de nouveau comme des objets indépendants.

Un objet actif qui a été groupé doit être dégroupé pour que vous puissiez accéder à ses propriétés ou à sa méthode. Il est toutefois possible de sélectionner un objet appartenant à un groupe sans devoir dégrouper l'ensemble : pour cela, effectuez Ctrl+clic (Windows) ou Commande+clic (Mac Os) sur l'objet (le groupe doit être sélectionné au préalable).

Grouper des objets n'a d'effet que dans l'éditeur de formulaires. Lors de l'exécution du formulaire, tous les objets groupés (hormis les boutons radio dans les bases binaires) se comportent comme s'ils étaient indépendants.

Il n'est pas possible de grouper des objets appartenant à des vues différentes et seuls les objets appartenant à la vue courante peuvent être groupés (cf. section [Utiliser les vues d'objet](#)).

Pour grouper les objets :

1. Sélectionnez les objets que vous souhaitez grouper.
2. Sélectionnez Grouper dans le menu Objets.

OR

Cliquez sur le bouton Grouper dans la barre d'outils de l'éditeur de formulaires :



4D matérialise les bordures du groupe avec des poignées. Les objets du groupe ne sont plus marqués séparément par des poignées. Désormais, lorsque vous modifiez le groupe d'objets, vous modifiez tous les objets qui le composent.

Pour dégrouper un groupe d'objets :

1. Sélectionnez le groupe que vous souhaitez dégrouper.
2. Choisissez Dégrouper dans le menu Objets.

OR

Sélectionnez la commande Dégrouper (menu du bouton Grouper) dans la barre d'outils de l'éditeur de formulaires.

Si la commande Dégrouper est désactivée, cela veut dire que l'objet sélectionné est déjà sous sa forme la plus simple.

4D rematérialise les bordures des objets qui constituaient le groupe avec des poignées.

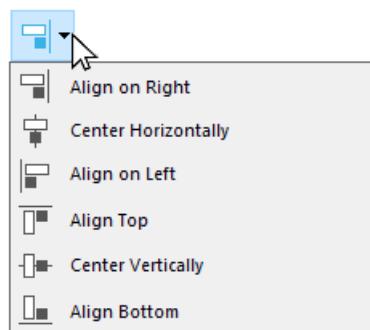
Aligner des objets

Vous pouvez aligner un ensemble d'objets entre eux ou à l'aide d'une grille magnétique.

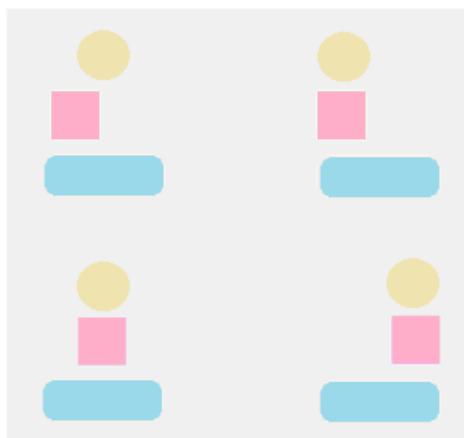
- Vous pouvez aligner entre eux des objets sur le haut, le bas, le côté, le centre horizontal ou le centre vertical. Vous pouvez aligner directement une sélection d'objets ou utiliser une boîte de dialogue vous permettant d'appliquer tout type d'alignement et de répartition aux objets sélectionnés. Cette boîte de dialogue vous permet en outre de sélectionner l'objet par rapport auquel vous voulez aligner les autres et de prévisualiser le résultat de vos paramétrages.
- Lorsque vous utilisez la grille magnétique, chaque objet peut être aligné manuellement avec les autres sur la base de positions "remarquables" représentées visuellement.

Utiliser les outils et les commandes d'alignement direct

Le sous-menu Aligner du menu Objets (ou du menu contextuel de l'éditeur) et les outils d'alignement de la barre d'outils vous permettent de rapidement aligner entre eux des objets sélectionnés.

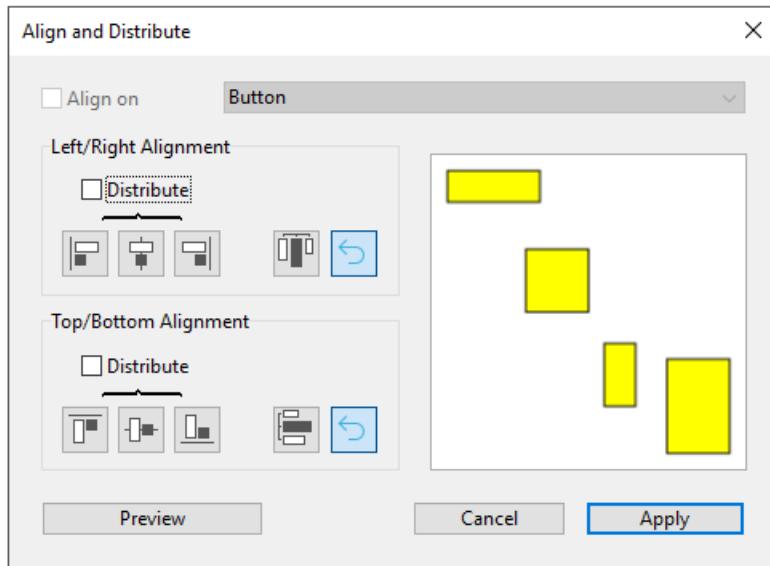


Lorsque 4D aligne des objets, il utilise l'objet le plus avancé dans la direction de l'alignement comme "ancre" sur laquelle tous les autres objets vont être alignés. This object is the "anchor." It uses the object that is the furthest in the alignment's direction as the anchor and aligns the other objects to that object. Par exemple, si vous alignez un groupe d'objets à droite, les objets seront alignés sur le côté droit de l'objet situé le plus à droite. Voici le résultat des alignements "aucun", "à gauche", "centré horizontalement" et "à droite" :



Utiliser la boîte de dialogue d'alignement

La boîte de dialogue d'alignement vous permet d'appliquer tout type d'alignement et/ou de répartition aux objets sélectionnés.



Pour afficher cette boîte de dialogue, vous devez sélectionner les objets que vous souhaitez aligner puis choisir la commande Alignement... dans le sous-menu Aligner du menu Objets ou du menu contextuel de l'éditeur.

- Cliquez sur l'icône d'alignement de votre choix dans les zones "Alignement droite/gauche" et/ou "Alignement haut/bas".

La zone d'exemple illustre le principe de l'opération sélectionnée.

- Pour effectuer un alignement standard des objets sélectionnés, cliquez sur le bouton Prévisualisation ou Appliquer.

Dans ce cas, 4D utilisera l'objet le plus avancé dans la direction de l'alignement comme "ancre" sur laquelle tous les autres objets vont être alignés. Par exemple, si vous alignez un groupe d'objets à droite, les objets seront alignés sur le côté droit de l'objet situé le plus à droite.

OU BIEN :

Pour aligner le groupe d'objets par rapport un objet particulier, cochez l'option Aligner sur puis sélectionnez dans la liste déroulante le nom de l'objet par rapport auquel aligner les autres. Dans ce cas, la position de l'objet de référence ne variera pas.

Vous pouvez prévisualiser le résultat réel de vos paramétrages en cliquant sur le bouton Prévisualisation. L'opération s'effectue dans l'éditeur de formulaires, mais la boîte de dialogue reste au premier plan. Vous pouvez alors Appliquer ou Annuler les modifications.

Cette boîte de dialogue combine l'alignement d'objets et leur répartition. Pour plus d'informations sur la répartition, reportez-vous au paragraphe [Répartir des objets](#).

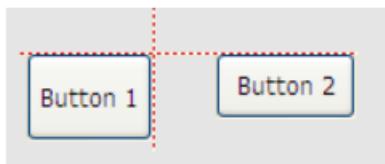
Utiliser l'alignement magnétique

L'éditeur de formulaires est doté d'une grille magnétique virtuelle qui peut vous aider à placer et à aligner des objets sur un formulaire. L'alignement magnétique des objets est basée sur la position relative des objets entre eux. Le magnétisme n'est utilisable que lorsqu'au moins deux objets sont présents dans le formulaire.

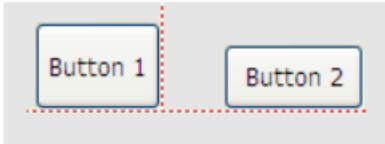
Le principe est le suivant : lorsque vous faites glisser un objet dans le formulaire, 4D indique des emplacements possibles pour cet objet sur la base d'alignements remarquables avec les autres objets du formulaire. Un alignement remarquable est établi à chaque fois que :

- Horizontalement, les extrémités ou les centres de deux objets coïncident,
- Verticalement, les extrémités de deux objets coïncident.

A ce moment, 4D place l'objet à l'emplacement et affiche un trait rouge indiquant l'alignement remarquable pris en compte :



En ce qui concerne la répartition des objets, 4D propose une distance basée sur les standards d'interface (20 points). Comme pour l'alignement magnétique, des traits rouges indiquent les distances remarquables au moment où elles sont atteintes.



Ce fonctionnement s'applique à tous les types d'objets des formulaires. Le magnétisme peut être activé ou désactivé à tout moment à l'aide de la commande Activer la grille magnétique du menu Formulaire ou du menu contextuel de l'éditeur. Il est également possible de définir l'activation par défaut de cette fonction dans la page Préférences >Formulaires (option Activer l'auto-alignement par défaut). Il est possible d'activer ou de désactiver manuellement la grille magnétique lorsqu'un objet est sélectionné en appuyant sur la touche Ctrl (Windows) ou Control (Mac Os).

Le magnétisme entraîne également l'observation de paliers lors du redimensionnement manuel des objets.

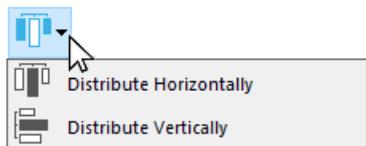
Distributing objects

Vous pouvez répartir des objets de manière à ce qu'ils soient disposés en respectant un espacement égal entre eux. Pour cela, vous pouvez utiliser des commandes directes de répartition ou passer par l'intermédiaire de la boîte de dialogue d'alignement et répartition pour effectuer des répartitions spécifiques ou combiner alignement et répartition. The latter allows you to align and distribute objects in one operation.

Lorsque la **grille magnétique** est activée, une aide visuelle est également fournie pour la répartition lors du déplacement manuel d'un objet.

Pour répartir directement une sélection d'objets (verticalement ou horizontalement) :

1. Sélectionnez les objets (au moins trois) que vous souhaitez répartir.
2. Dans la barre d'outils, cliquez sur l'outil de répartition qui correspond la répartition que vous souhaitez appliquer.



OR

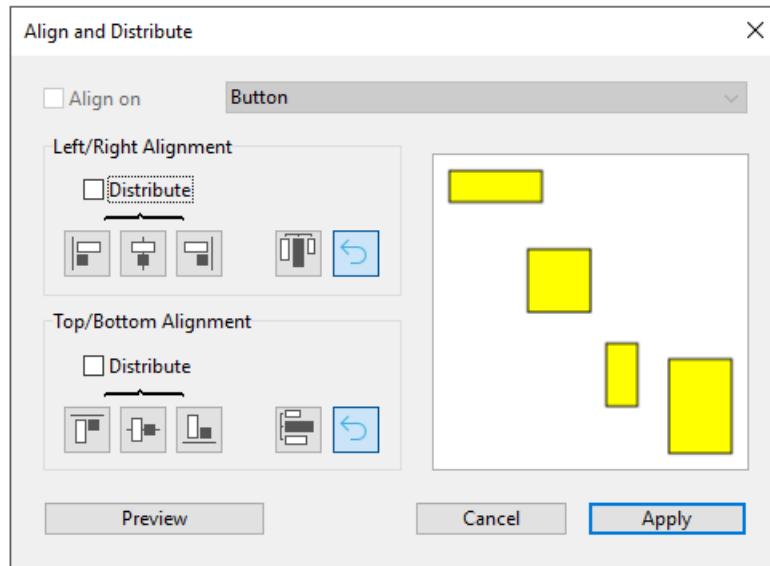
Choisissez la commande Répartir horizontalement ou Répartir verticalement dans le sous-menu Aligner du menu Objets ou du menu contextuel de l'éditeur.

4D répartit les objets par rapport à leurs centres, la plus grande distance entre deux objets contigus est utilisée comme distance de référence.

Pour répartir des objets à l'aide de la boîte de dialogue d'alignement et répartition :

1. Sélectionnez les objets que vous souhaitez répartir.
2. Choisissez la commande Alignement... dans le sous-menu Aligner du menu Objets ou du menu contextuel de l'éditeur.

La boîte de dialogue suivante apparaît :



3. Cliquez sur l'icône de répartition standard (horizontale ou verticale) de votre choix:



(icône de répartition horizontale standard)

La zone d'exemple illustre le principe de l'opération sélectionnée.

4. Pour effectuer une répartition standard, cliquez sur le bouton *Prévisualisation* ou *Appliquer*.

Dans ce cas, les objets seront répartis de manière à ce que leurs côtés soient équidistants (répartition standard).

OU BIEN :

Pour effectuer une répartition spécifique (par exemple répartir les objets de manière à ce que leurs côtés droits — et non plus leurs intervalles soient équidistants), cochez une option Répartir. This option acts like a switch. Lorsque l'option Répartir est cochée, les icônes situées au-dessous d'elle s'appliquent alors à la répartition :

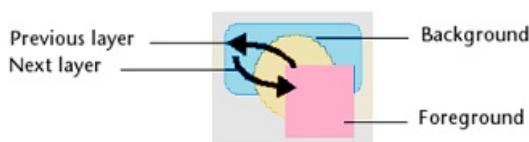
- Horizontalement, les icônes correspondent aux répartitions suivantes : équidistance des côtés gauches, des centres (hor.) et des côtés droits des objets sélectionnés.
- Verticalement, les icônes correspondent aux répartitions suivantes : équidistance des bords supérieurs, des centres (vert.) et des bords inférieurs des objets sélectionnés.

Vous pouvez prévisualiser le résultat réel de vos paramétrages en cliquant sur le bouton *Prévisualisation* : l'opération s'effectue dans l'éditeur de formulaires, mais la boîte de dialogue reste au premier plan. Vous pouvez alors *Appliquer* ou *Annuler* les modifications. Vous pouvez alors *Appliquer* ou *Annuler* les modifications.

Cette boîte de dialogue vous permet de combiner l'alignement d'objets et leur répartition. Pour plus d'informations sur l'alignement, reportez-vous au paragraphe [Aligner des objets](#).

Gérer les plans des objets

Il est parfois nécessaire de réorganiser certains objets qui occultent d'autres objets du formulaire. Par exemple, vous pouvez souhaiter voir apparaître un graphique derrière les champs dans un formulaire. 4D propose 4 commandes, Passer au dernier plan, Passer au premier plan, Plan suivant et Plan précédent, qui vous permettent d'organiser les plans des objets du formulaire. These layers also determine the default entry order (see [Modifying data entry order](#)). La fenêtre ci-dessous représente des objets organisés en couches :

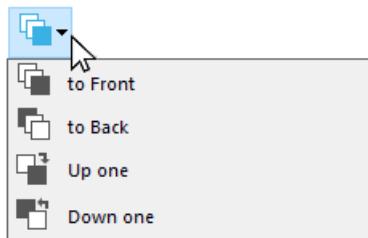


Pour modifier le plan d'un objet, sélectionnez-le et choisissez :

- Une des commandes Passer à l'avant-plan, Passer au dernier plan, Plan suivant et Plan précédent dans le menu

Objet,

- Une des commandes du sous-menu Plan> du menu contextuel de l'éditeur,
- Une des commandes associées au bouton de gestion des plans de la barre d'outils.



Lorsque plusieurs objets sont superposés, le raccourci Ctrl+clic / Commande+clic permet de sélectionner successivement chaque objet en descendant d'un plan à chaque clic.

Pour ordonner les différents plans, 4D va toujours de l'arrière-plan vers l'avant-plan. Par conséquent, le plan précédent fait reculer la sélection d'objets d'un plan vers l'arrière-plan du formulaire. Le plan suivant fait avancer la sélection d'objets d'un plan vers l'avant-plan du formulaire.

Ordre de saisie des données

L'ordre de saisie est l'ordre dans lequel les champs, les sous-formulaires et les autres objets actifs sont sélectionnés lorsque vous appuyez sur la touche Tabulation ou Retour chariot dans un formulaire. Il est possible de parcourir le formulaire dans le sens inverse de l'ordre de saisie en appuyant sur les touches Maj+Tabulation ou Maj+Retour chariot.

You can change the entry order at runtime using the `FORM SET ENTRY ORDER` and `FORM GET ENTRY ORDER` commands.

Every object that supports the `focusable` property is included in the data entry order by default.

Setting the entry order for a JSON form is done with the `entryOrder` property.

Si vous ne spécifiez pas d'ordre de saisie personnalisé, 4D utilise par défaut le plan des objets comme ordre de saisie, dans le sens "arrière-plan vers premier plan." Par défaut, l'ordre de saisie correspond donc à l'ordre de création des objets dans le formulaire.

Dans certains formulaires, il est nécessaire de définir un ordre de saisie personnalisé. Ci-dessous par exemple, des champs supplémentaires relatifs à l'adresse ont été ajoutés après la création du formulaire. The resulting standard entry order thus becomes illogical and forces the user to enter the information in an awkward manner:

A screenshot of a 'Employees' form in 4D. The form includes fields for Last Name, First Name, Full address, Telephone, Company, and Hire date. The 'Full address' field is currently selected. A series of arrows points from the 'Last Name' field to the 'First Name' field, then to the 'Full address' field, then to the 'Telephone' field, then to the 'Company' field, and finally to the 'Hire date' field. This visual cue indicates the current entry order, which may be counterintuitive given the physical layout of the form fields.

Il est dans ce cas nécessaire de personnaliser l'ordre de saisie afin de proposer une progression plus logique :

The screenshot shows the 'Employees' form with various fields and their corresponding input sequences. The sequence is visualized by arrows connecting the fields in the order they appear on the screen: Last Name, First Name, Full address, Telephone, Company, and Hire date.

Visualiser et modifier l'ordre de saisie

Vous pouvez visualiser l'ordre de saisie courant soit à l'aide des badges "Ordre de saisie", soit à l'aide du mode "Ordre de saisie". En revanche, vous ne pouvez modifier l'ordre de saisie qu'en mode "Ordre de saisie".

Ce paragraphe décrit la visualisation et la modification de l'ordre de saisie en mode "Ordre de saisie". Pour plus d'informations sur la visualisation de l'ordre de saisie à l'aide des badges, reportez-vous au paragraphe [Using shields](#).

Pour visualiser ou modifier l'ordre de saisie :

1. Sélectionnez Ordre de saisie dans le menu Formulaire ou cliquez sur le bouton dans la barre d'outils de la fenêtre :



Le pointeur prend la forme d'un pointeur d'ordre, et 4D dessine une ligne qui permet de visualiser la séquence de l'ordre de saisie courant.

Visualiser et modifier l'ordre de saisie sont les seules opérations que vous pouvez réaliser dans ce mode.

2. Pour changer l'ordre de saisie, placez le pointeur sur un objet, cliquez dessus et, tout en maintenant le bouton de la souris enfoncé, déplacez le pointeur vers l'objet qui doit le suivre dans l'ordre de saisie.

The screenshot shows the 'Employees' form with the same fields as before, but the input sequence has been modified. The sequence is now: Last Name → First Name → Full address → Telephone → Company → Hire date. This indicates that the user has moved the 'Telephone' field after 'Full address' and the 'Company' field after 'Telephone'.

4D ajuste l'ordre de saisie en conséquence.

3. Répétez l'étape 2 autant de fois que nécessaire pour obtenir le nouvel ordre de saisie.
4. Lorsque vous êtes satisfait de l'ordre de saisie, sélectionnez de nouveau la commande Ordre de saisie dans le menu Formulaire.

4D retourne dans le mode de fonctionnement normal de l'éditeur de formulaires.

Seul l'ordre de saisie de la page courante du formulaire est affiché. Si le formulaire contient des objets saisissables sur la page 0 ou provenant d'un formulaire hérité, l'ordre de saisie par défaut est le suivant : Objets de la page zéro du formulaire hérité > Objets de la page 1 du formulaire hérité > Objets de la page zéro du formulaire ouvert > Objets de la page courante du formulaire ouvert.

Utiliser un groupe de saisie

Lorsque vous changez l'ordre de saisie, vous pouvez sélectionner un groupe d'objets dans le formulaire afin que l'ordre de saisie s'applique aux objets du groupe. Ceci vous permet de définir facilement l'ordre de saisie pour les formulaires dans lesquels les champs sont organisés en groupes et colonnes.

Pour créer un groupe de saisie :

1. Sélectionnez Ordre de saisie dans le menu *Formulaire* ou cliquez sur le bouton dans la barre d'outils de la fenêtre.
2. Dessinez un rectangle de sélection autour des objets que vous souhaitez grouper pour la saisie.

Lorsque vous relâchez le bouton de la souris, les objets contenus ou touchés par le rectangle suivent l'ordre de saisie par défaut. L'ordre de saisie des autres objets est réorganisé en conséquence.

Exclude un objet de l'ordre de saisie

By default, all objects that support the focusable property are included in the entry order. To exclude an object from the entry order:

1. Select the Entry order mode, then
2. shift-click on the object
3. right-click on the object and select Remove from entry order option from the context menu

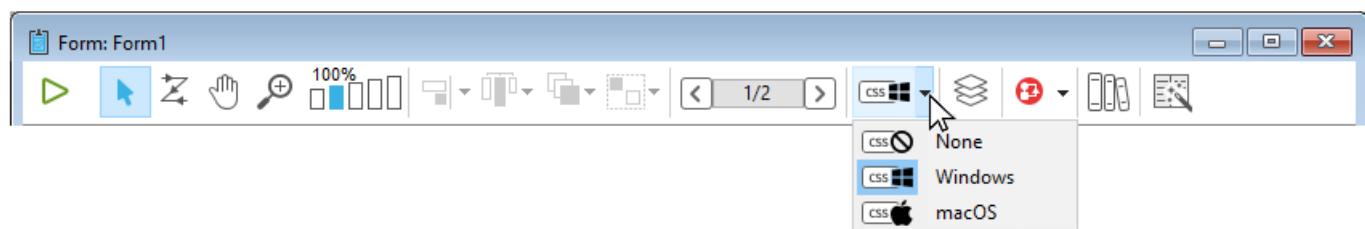
CSS Preview

The Form editor allows you to view your forms with or without applied CSS values.

When [style sheets](#) have been defined, forms (including inherited forms and subforms) are opened in the CSS Preview mode for your operating system by default.

Selecting CSS Preview Mode

The Form editor toolbar provides a CSS button for viewing styled objects:



Select one of the following preview modes from the menu:

Toolbar Icon	CSS Preview Mode	Description
	Aucun	No CSS values are applied in the form and no CSS values or icons displayed in the Property List.
	Sous Windows	CSS values for Windows platform are applied in the form. CSS values and icons displayed in the Property List.
	macOS	CSS values for macOS platform are applied in the form. CSS values and icons displayed in the Property List.

If a font size too large for an object is defined in a style sheet or JSON, the object will automatically be rendered to accommodate the font, however the size of the object will not be changed.

The CSS preview mode reflects the priority order applied to style sheets vs JSON attributes as defined in the [JSON vs Style Sheet](#) section.

Once a CSS preview mode is selected, objects are automatically displayed with the styles defined in a style sheet (if any).

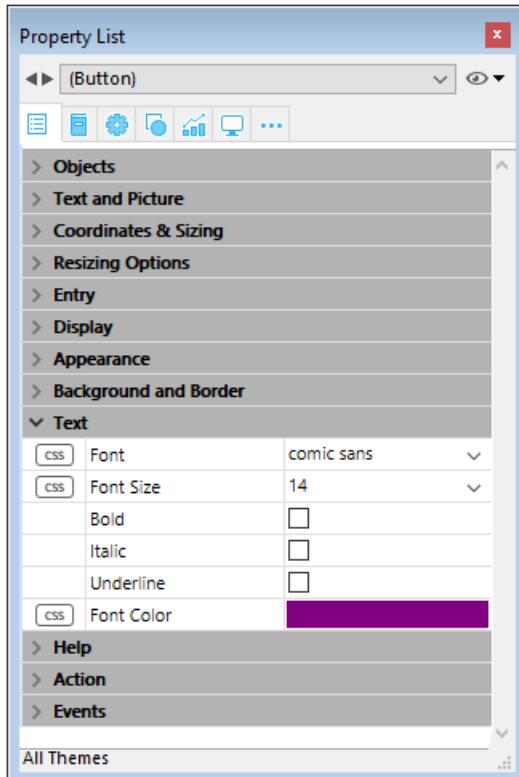
When copying or duplicating objects, only the CSS references (if any) and the JSON values are copied.

CSS support in the Property List

In CSS Preview mode, if the value of an attribute has been defined in a style sheet, the attribute's name will appear with a CSS icon displayed next to it in the Property List. For example, the attribute values defined in this style sheet:

```
.myButton {  
font-family: comic sans;  
font-size: 14;  
stroke: #800080;  
}
```

are displayed with a CSS icon in the Property List:



An attribute value defined in a style sheet can be overridden in the JSON form description (except if the CSS includes the `!important` declaration, see below). In this case, the Property List displays the JSON form value in **bold**. You can reset the value to its style sheet definition with the **Ctrl + click** (Windows) or **Command + click** (macOs) shortcuts.

If an attribute has been defined with the `!important` declaration for a group, an object within a group, or any object within a selection of multiple objects, that attribute value is locked and cannot be changed in the Property List.

Property List CSS Icons

Icône	Description
<code>css</code>	Indicates that an attribute value has been defined in a style sheet
<code>css !</code>	Indicates that an attribute value has been defined in a style sheet with the <code>!important</code> declaration
<code>...</code>	Displayed when an attribute value defined in a style sheet for at least one item in a group or a selection of multiple objects is different from the other objects

Création de list box

Vous pouvez créer rapidement de nouvelles list box de type sélection d'entités avec le générateur de list box. La nouvelle list box peut être utilisée immédiatement ou elle peut être modifiée via l'éditeur de formulaires.

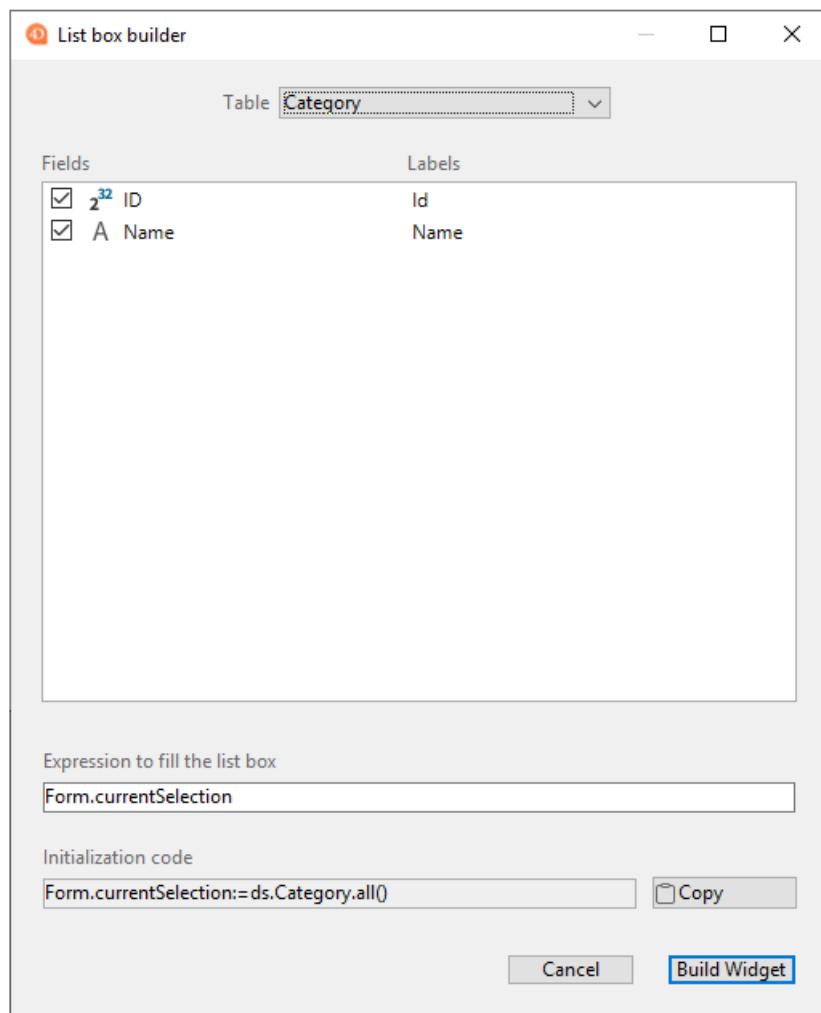
Le générateur de list box vous permet de créer et de remplir des list box de type sélection d'entités en quelques opérations simples.

Utilisation du générateur de list box

1. Dans la barre d'outils de l'éditeur de formulaire, cliquez sur l'icône du générateur de zone de liste :



Le générateur de list box s'affiche :



2. Sélectionnez une table dans la liste déroulante Table :



3. Sélectionnez les champs de la list box dans la zone Champs :

Fields		Labels
<input type="checkbox"/>	z ³² CategoryID	Category id
<input type="checkbox"/>	A Email	Email
<input checked="" type="checkbox"/>	A FirstName	First Name
<input type="checkbox"/>	A HomeAddress	Home Address
<input type="checkbox"/>	A HomeCity	Home City
<input type="checkbox"/>	A HomeCountry	Home Country
<input type="checkbox"/>	A HomePhone	Home Phone
<input type="checkbox"/>	A HomeState	Home State
<input type="checkbox"/>	A HomeZip	Home zip
<input type="checkbox"/>	z ³² ID	Id
<input type="checkbox"/>	A JobTitle	Job Title
<input checked="" type="checkbox"/>	A LastName	Last Name
<input type="checkbox"/>	A MiddleName	Middle Name
<input type="checkbox"/>	A MobilePhone	Mobile Phone
<input type="checkbox"/>	A Notes	Notes
<input checked="" type="checkbox"/>	A Organization	Organization

Par défaut, tous les champs sont sélectionnés. Vous pouvez sélectionner ou désélectionner les champs individuellement ou utiliser Ctrl+clic (Windows) ou Cmd+clic (macOS) pour les sélectionner ou les désélectionner tous à la fois.

Vous pouvez modifier l'ordre des champs via un glisser-déposer.

4. L'expression qui permet de remplir les lignes de la list box à partir de la sélection d'entité est préremplie :

Expression to fill the list box

Cette expression peut être modifiée si nécessaire.

5. En cliquant sur le bouton Copier, l'expression sera copiée pour charger tous les enregistrements en mémoire :

Initialization code

```
Form.currentSelection:=ds.Contact.all()
```

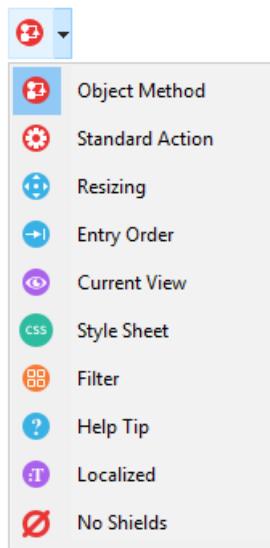
6. Cliquez sur le bouton Créer un widget pour créer la list box.

Build widget

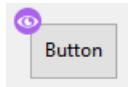
La list box finale :

Badges

L'éditeur de formulaires 4D permet d'utiliser des badges afin de faciliter la visualisation des propriétés des objets. Ils se trouvent dans la barre d'outils du formulaire :



Le principe de cette fonction est le suivant : chaque badge est associé à une propriété (par exemple `Vues`, signifiant que l'objet "est dans la vue courante"). Lorsque vous activez un badge, 4D affiche une petite icône (un badge) en haut à gauche de chaque objet du formulaire auquel s'applique la propriété.



Utilisation des badges

Pour activer un badge, cliquez sur l'icône *badge* jusqu'à ce que le badge souhaité soit sélectionné. Vous pouvez également cliquer sur la partie droite du bouton et sélectionner directement le type de badge à afficher dans le menu associé :

Pour ne pas afficher de badges, choisissez la ligne `Pas de badges` dans le menu de sélection.

Vous pouvez définir les badges à afficher par défaut dans la Préférences de l'application.

Description du badge

Voici la description de chaque type de badge :

Icône	Nom	Est affiché...
🔗	Méthode objet	Pour les objets auxquels une méthode objet est associée
⚙️	Action standard	Pour les objets auxquels une action standard est associée
🌐	Redimensionnement	Pour les objets disposant d'au moins une propriété de redimensionnement, indique la combinaison de propriétés courante
➡️	Ordre de saisie	Pour les objets saisissables, indique le numéro d'ordre de saisie
👁️	Vue courante	Pour tous les objets de la vue courante
css	Feuille de style	Pour les objets avec une ou plusieurs valeurs d'attribut remplacées par une feuille de style.
FilterWhere	Filtre	Pour les objets saisissables auxquels un filtre de saisie est associé
?	Infobulle	Pour les objets auxquels une infobulle (message d'aide) est associée
:T	Localisé	Pour les objets dont le libellé provient d'une référence (libellé débutant par ":"). La référence peut être de type ressource (STR#) ou XLIFF
∅	Pas de badge	Aucun badge n'apparaît

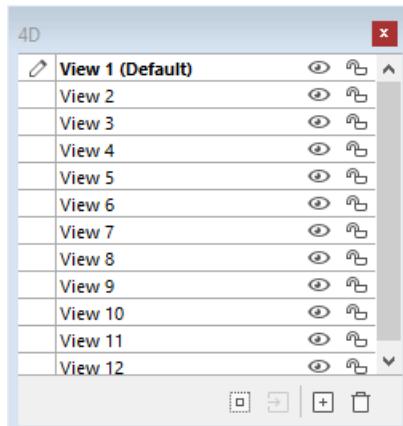
Vues

L'éditeur de formulaires 4D vous permet de créer des formulaires complexes en distribuant des objets de formulaire entre des vues séparées qui peuvent ensuite être masquées ou affichées selon les besoins.

Par exemple, vous pouvez répartir les objets par type (champs, variables, objets statiques, etc.). Tout type d'objet formulaire, y compris les sous-formulaires et les zones de plug-in, peut être inclus dans les vues.

Il n'y a pas de limite au nombre de vues par formulaire. Vous pouvez créer autant de vues différentes que nécessaire. De plus, chaque vue peut être affichée, masquée et/ou verrouillée.

Les vues sont gérées via la palette de vues.



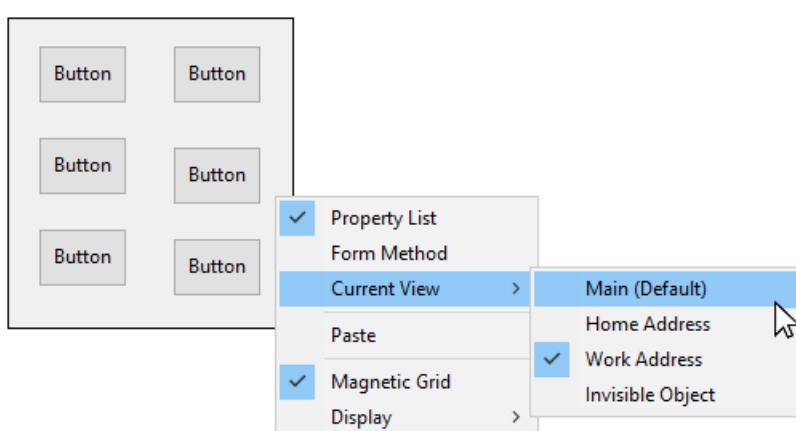
Accéder à la palette de vues

Il existe trois façons d'accéder à la palette de vues :

- Barre d'outils : cliquez sur l'icône de Vues dans la barre d'outils de l'éditeur de formulaires. (Cette icône apparaît en gris lorsqu'au moins un objet appartient à une vue autre que la vue par défaut.)

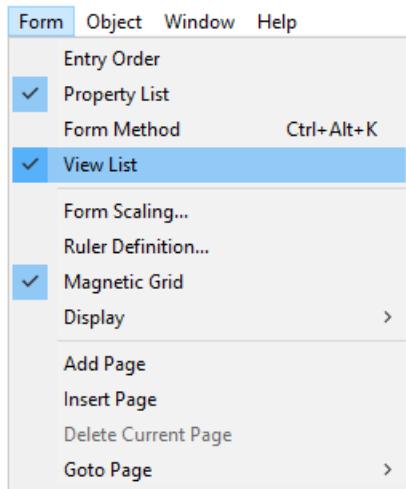
Vue par défaut uniquement	Avec des vues supplémentaires

- Menu contextuel (formulaire ou objet) : faites un clic droit n'importe où dans l'éditeur de formulaires ou sur un objet, puis sélectionnez Vue courante



La vue courante est indiquée par une coche (par exemple, "Adresse professionnelle" dans l'image ci-dessus)

- Menu Formulaire : cliquez sur le menu Formulaire et sélectionnez Afficher la liste



Avant de commencer

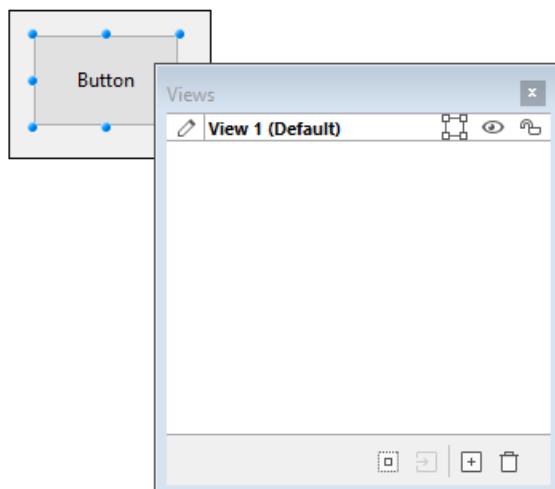
Voici quelques éléments importants à connaitre avant de commencer à travailler avec les vues :

- Contexte d'utilisation : les vues sont un outil purement graphique, utilisable uniquement dans l'éditeur de formulaires ; il n'est pas possible d'accéder aux vues par programmation ou en mode Application.
- Vues et pages : Les objets d'une même vue peuvent appartenir à des pages différentes d'un formulaire ; seuls les objets de la page courante (et de la page 0 si elle est visible) peuvent être affichés, quelle que soit la configuration des vues.
- Vues et plans : Les vues sont indépendantes des plans des objets, il n'y a pas de hiérarchie d'affichage entre les différentes vues.
- Vues et groupes : Seuls les objets appartenant à la vue courante peuvent être groupés.
- Vues courantes et par défaut : la vue par défaut est la première vue d'un formulaire et ne peut pas être supprimée; la vue courante est la vue en cours de modification et le nom est affiché en gras.

Gestion des vues

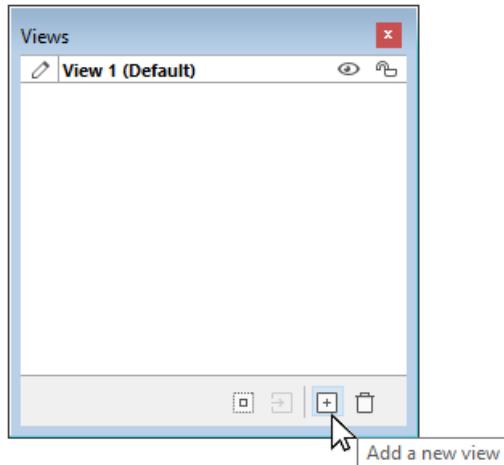
Créer des vues

Tout objet créé dans un formulaire est placé dans la première vue ("Vue 1") du formulaire. La première vue 1 est toujours la vue par défaut, indiquée par (par défaut) après le nom. Le nom de la vue peut être modifié (voir [Renommer les vues](#)), mais il demeure la vue par défaut.

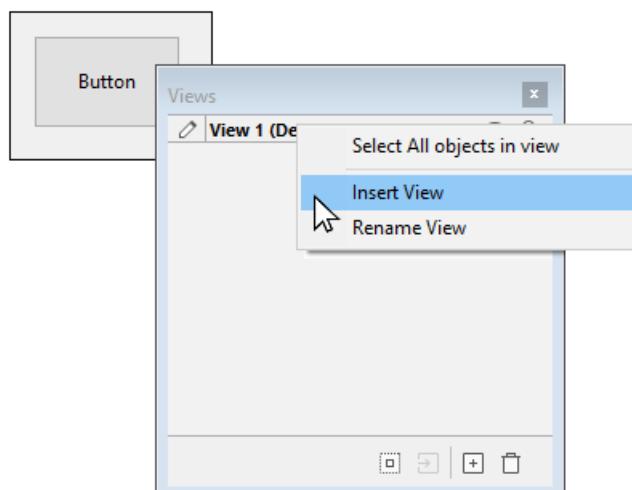


Il existe deux façons d'ajouter des vues supplémentaires :

- Cliquez sur le bouton Ajouter une nouvelle vue en bas de la palette Vue :



- Faites un clic droit sur une vue existante et sélectionnez Insérer une vue :



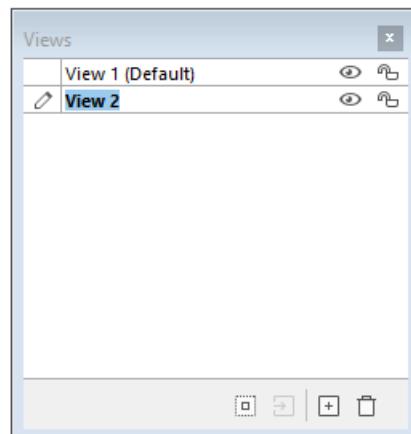
Il n'y a pas de limitation du nombre de vues.

Renommer des vues

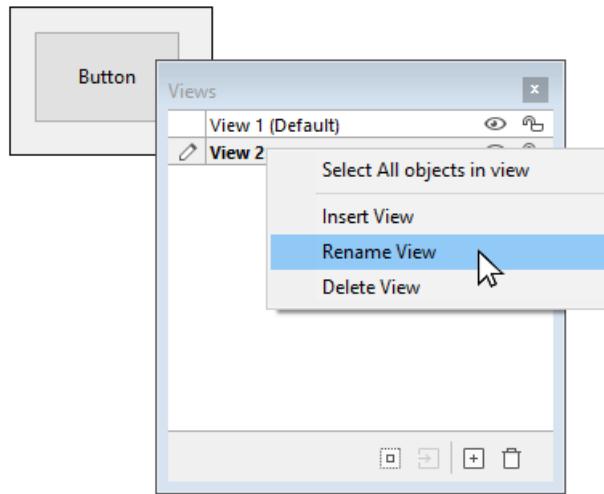
Par défaut, les vues sont nommées "Vue" + le numéro de vue, mais vous pouvez modifier ces noms pour améliorer la lisibilité et mieux répondre à vos besoins.

Pour renommer une vue, vous pouvez soit :

- Double-cliquer directement sur le nom de la vue (dans ce cas, la vue est sélectionnée). Le nom devient alors éditable :



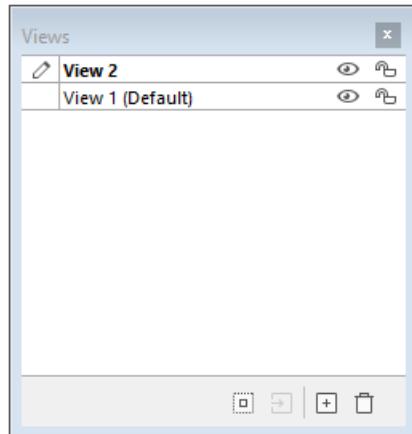
- Faire un clic droit sur le nom de la vue. Le nom devient alors éditable :



Réordonner les vues

Vous pouvez modifier l'ordre d'affichage des vues en les faisant glisser/déposer dans la palette des vues.

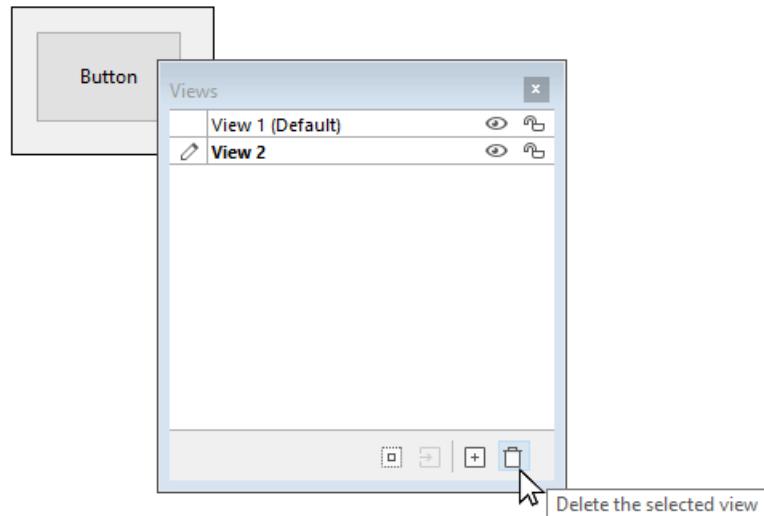
A noter que la vue par défaut ne change pas :



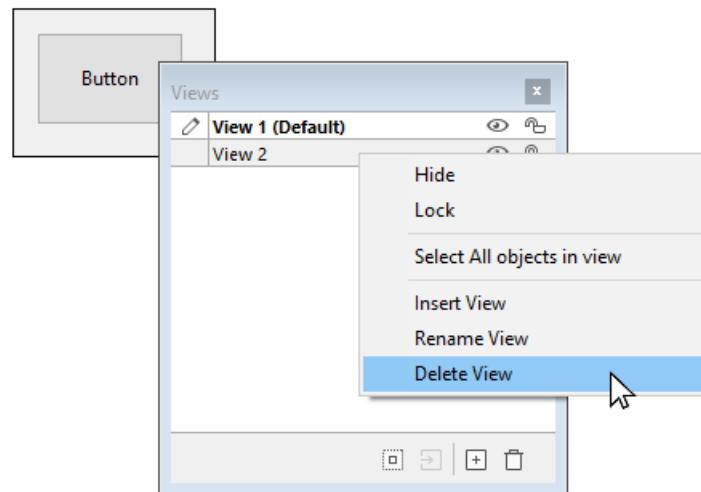
Supprimer des vues

Pour renommer une vue, vous pouvez soit :

- Cliquer sur le bouton Supprimer la vue sélectionnée en bas de la palette des vues :



- Faire un clic droit sur le nom de la vue et sélectionner Supprimer la vue :



Si une vue est supprimée, tous les objets qu'elle contient sont automatiquement déplacés vers la vue par défaut.

Utilisation des vues

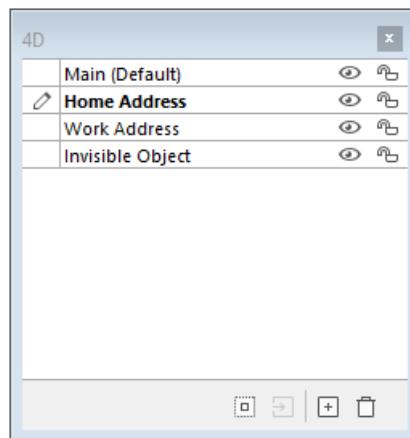
Une fois que les vues sont créées, vous pouvez utiliser la palette des vues pour :

- Ajouter un objet aux vues,
- Déplacer des objets d'une vue à une autre,
- Sélectionner tous les objets d'une même vue en un seul clic,
- Afficher ou masquer des objets pour chaque vue,
- Verrouiller les objets d'une vue.

Ajouter des objets aux vues

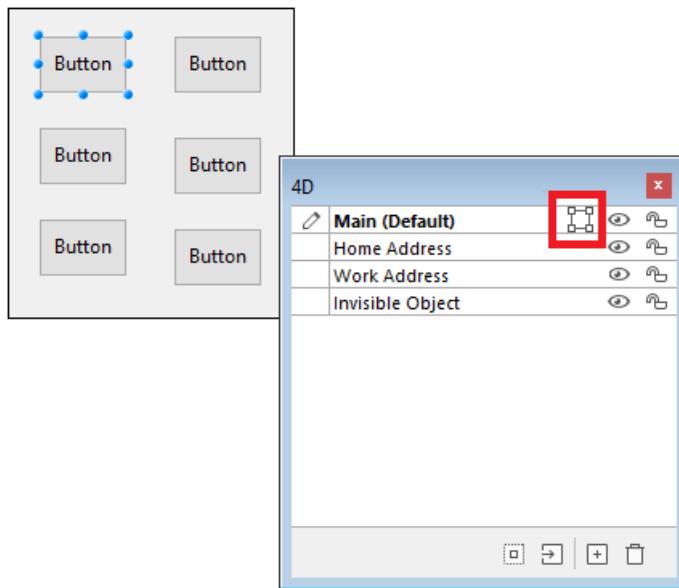
Un objet ne peut appartenir qu'à une seule vue.

Pour créer un objet dans une autre vue, sélectionnez simplement la vue dans la palette des vues (avant de créer l'objet) en cliquant sur son nom (une icône Modifier est affichée pour la **Vue courante** et le nom apparaît en gras) :



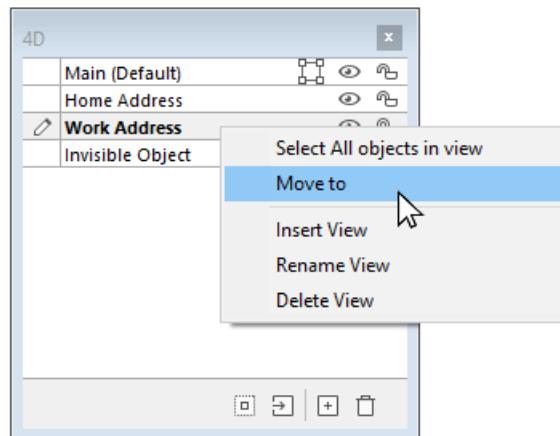
Déplacer des objets entre les vues

Il est également possible de déplacer un ou plusieurs objets d'une vue à une autre. Dans le formulaire, sélectionnez le ou les objets dont vous souhaitez modifier la vue. La liste des vues indique, à l'aide d'un symbole, la vue à laquelle appartient la sélection :



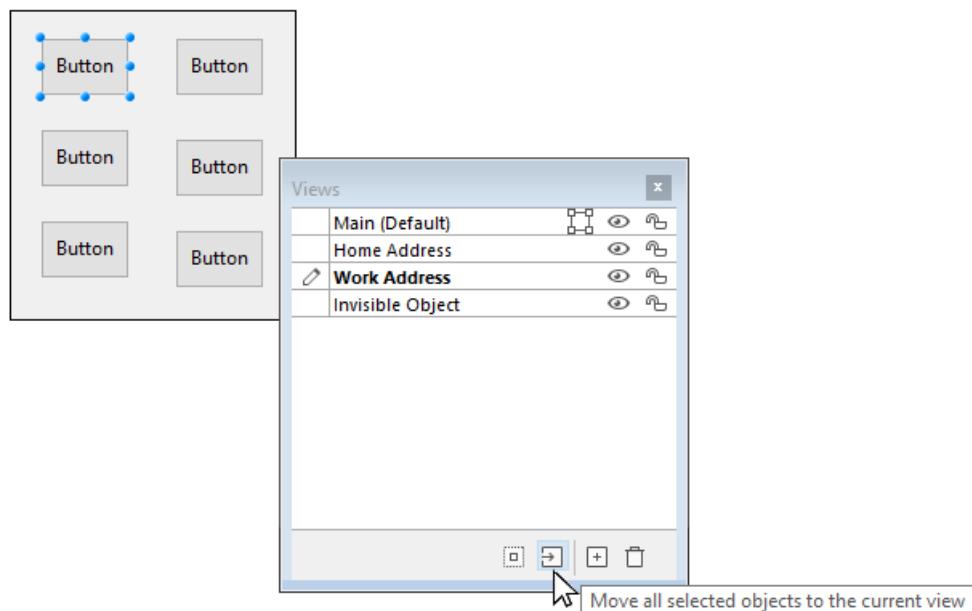
La sélection peut contenir plusieurs objets appartenant à différentes vues.

Sélectionnez simplement la vue de destination, faites un clic droit puis sélectionnez Déplacer vers :

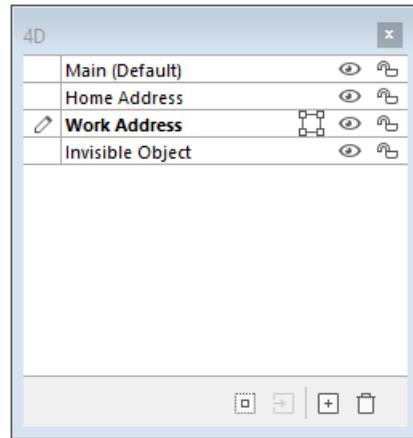


OR

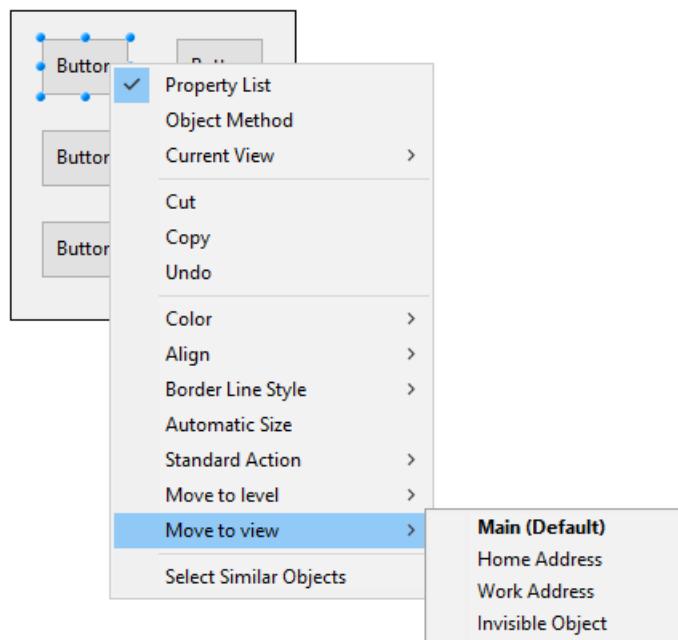
Sélectionnez la vue de destination de la sélection et cliquez sur le bouton Déplacer vers en bas de la palette des vues :



La sélection est ensuite placée dans la nouvelle vue :



Vous pouvez également déplacer un objet vers une autre vue via le menu contextuel de l'objet. Faites un clic droit sur l'objet, sélectionnez Déplacer vers la vue puis sélectionnez une vue dans la liste de vues disponibles :

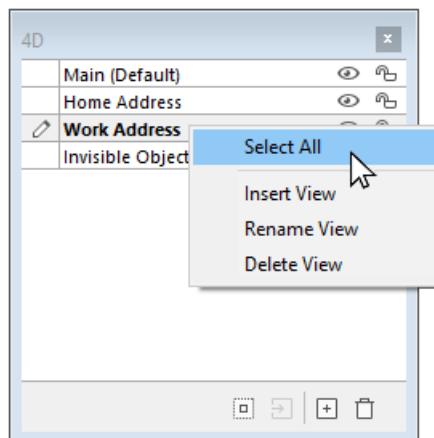


La **vue courante** est affichée en texte gras.

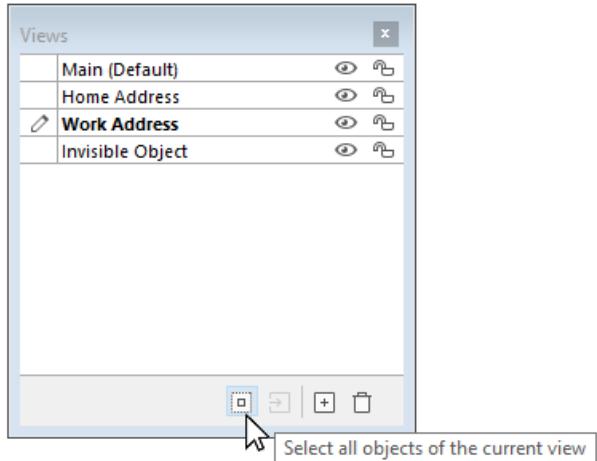
Sélectionner tous les objets d'une vue

Vous pouvez sélectionner dans la page courante du formulaire tous les objets appartenant à une même vue. Cette fonction est utile pour appliquer des modifications globales à un ensemble d'objets.

Pour cela, faites un clic droit sur la vue dans laquelle vous souhaitez sélectionner tous les objets et cliquez sur le bouton Tout sélect. dans vue:



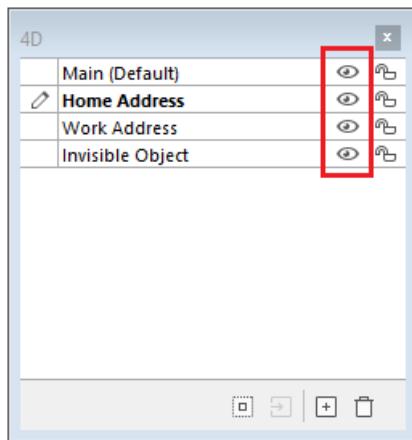
Vous pouvez également utiliser le bouton situé en dessous de la palette des vues :



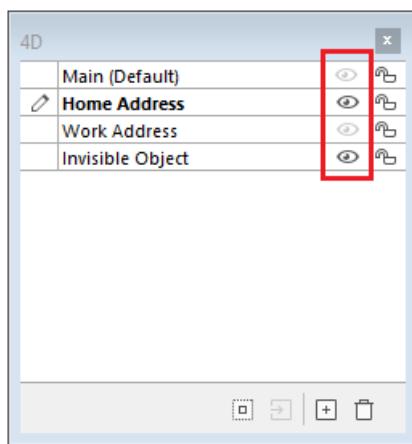
Afficher ou masquer les objets d'une vue

Vous pouvez à tout moment afficher ou masquer les objets d'une vue dans la page courante du formulaire. Cette fonction permet par exemple de se concentrer sur certains objets lors de la modification du formulaire.

Par défaut, toutes les vues sont affichées, comme l'indique l'icône en regard de chaque vue dans la palette des vues:



Pour masquer une vue, cliquez sur cette icône. Elle est alors grisée et les objets de la vue correspondante ne sont plus affichés dans le formulaire :



Il n'est pas possible de masquer la [vue courante](#).

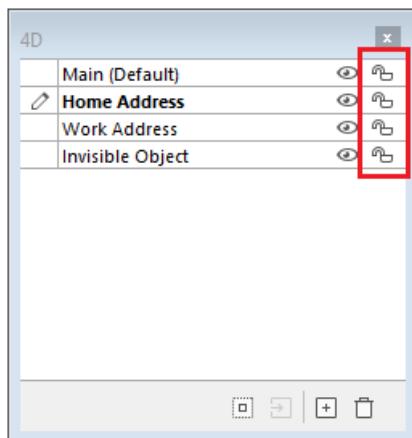
Pour afficher une vue masquée, il suffit de la sélectionner ou de cliquer de nouveau sur l'icône de visualisation.

Verrouiller les objets d'une vue

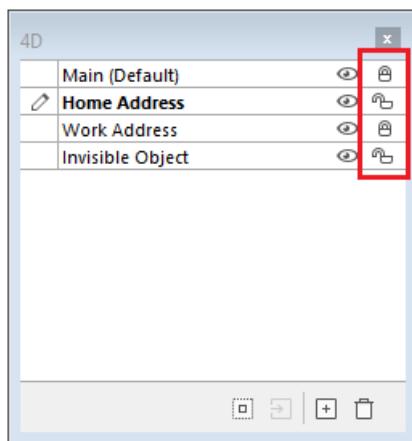
Vous pouvez verrouiller les objets d'une vue. Cela empêche leur sélection, leur modification ou leur suppression dans le formulaire. Une fois verrouillé, un objet ne peut pas être sélectionné par un clic, un rectangle de sélection ou la commande Sélectionner objets de même type du menu contextuel. Cette fonction est utile pour éviter les erreurs de

manipulation.

Par défaut, toutes les vues sont déverrouillées, comme l'indique l'icône en regard de chaque vue dans la palette des vues:



Pour verrouiller les objets d'une vue, cliquez sur cette icône. Le cadenas est alors refermé, ce qui indique que la vue est verrouillée :



Il n'est pas possible de verrouiller la [vue courante](#).

Pour déverrouiller une vue, il suffit de la sélectionner ou de cliquer à nouveau sur l'icône de verrouillage.

Zoom

Il est possible de zoomer dans le formulaire courant. Vous pouvez passer en mode "Zoom" soit en cliquant sur l'icône de loupe, soit en cliquant directement sur la barre correspondant à l'échelle désirée dans la barre d'outils de la fenêtre (les paliers d'affichage sont 50%, 100%, 200%, 400% et 800%) :



- Lorsque vous cliquez sur le bouton loupe, le curseur prend la forme d'une loupe. Pour réduire le pourcentage d'affichage d'un palier, appuyez sur la touche Majuscule et cliquez dans le formulaire.
- Lorsque vous cliquez sur une barre de pourcentage, l'affichage est immédiatement modifié.

En mode Zoom, toutes les fonctions de l'éditeur de formulaires restent disponibles(*) .

(*) Pour des raisons techniques, il n'est toutefois pas possible de sélectionner d'élément de list box (en-tête, colonne ou pied) lorsque l'éditeur de formulaires est en mode Zoom.

Form Editor Macros

L'éditeur de formulaires 4D prend en charge les macros. Une macro est un ensemble d'instructions permettant de réaliser une action ou une séquence d'actions. Lorsqu'elle est appelée, la macro exécutera ses instructions et, automatiquement, les actions.

Par exemple, si vous avez un rapport récurrent avec une mise en forme spécifique (par exemple, certains textes doivent apparaître en rouge et certains textes en vert), vous pouvez créer une macro pour définir automatiquement la couleur. Vous pouvez créer des macros pour l'éditeur de 4D Form qui peuvent :

- Créer et exécuter du code 4D
- Afficher les boîtes de dialogue
- Sélectionnez des objets de formulaire
- Ajouter / supprimer / modifier des formulaires, des objets de formulaire ainsi que leurs propriétés
- Modifier les fichiers de projet (mettre à jour, supprimer)

Le code des macros prend en charge les [class functions \(fonctions de classe\)](#) et les [propriétés d'objet de formulaire en JSON](#) pour vous permettre de définir n'importe quelle fonctionnalité personnalisée dans l'éditeur de formulaire.

Des macros peuvent être définies pour le projet hôte ou pour les composants du projet. Habituellement, vous créez une macro et l'installez dans les composants que vous utilisez pour le développement.

Lorsqu'elle est appelée, une macro remplace tous les comportements précédemment spécifiés.

Exemple pratique

Dans ce court exemple, vous verrez comment créer et appeler une macro qui ajoute un bouton d'alerte "Hello World!" dans le coin supérieur gauche de votre formulaire.

1. Dans un fichier `formMacros.json` dans le dossier `Sources` de votre projet, entrez le code suivant :

```
{  
  "macros": {  
    "Add Hello World button": {  
      "class": "AddButton"  
    }  
  }  
}
```

2. Créez une classe 4D nommée `AddButton`.
3. Dans la classe `AddButton`, écrivez la fonction suivante :

```

Function onInvoke($editor : Object)->$result : Object

var $btnHello : Object

// Créer un bouton "Hello"
$btnHello:=New object("type"; "button"; \
"text"; "Hello World!"; \
"method"; New object("source"; "ALERT(\"Hello World!\")"); \
"events"; New collection("onClick"); \
"width"; 120; \
"height"; 20; \
"top"; 0; \
"left"; 0)

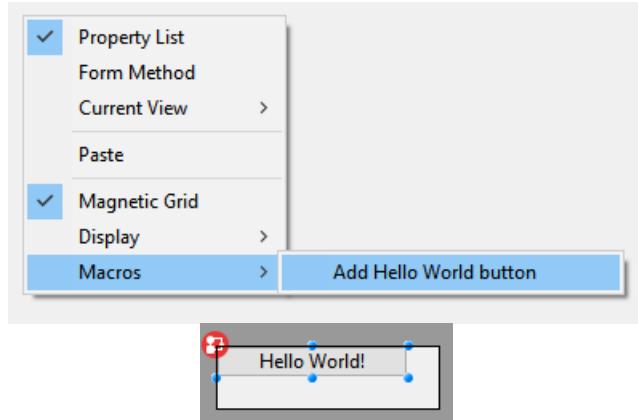
// Ajouter un bouton dans la page courante
$editor.editor.currentPage.objects.btnHello:=$btnHello

// Sélectionner le nouveau bouton dans l'éditeur de formulaire
$editor.editor.currentSelection.clear() //unselect elements
$editor.editor.currentSelection.push("btnHello")

// Notifier la modification à l'éditeur de formulaire 4D
$result:=New object("currentSelection"; $editor.editor.currentSelection; \
"currentPage"; $editor.editor.currentPage)

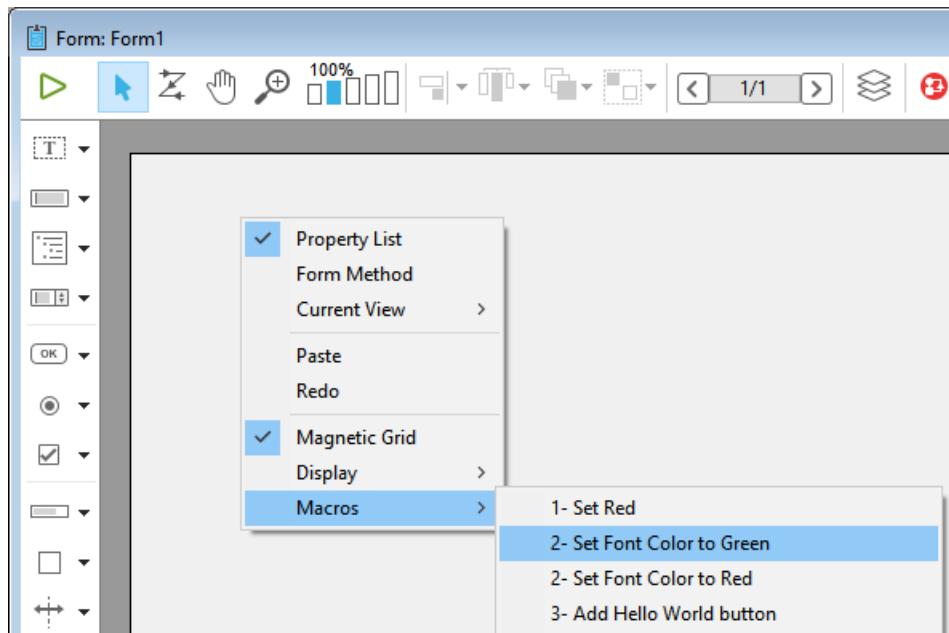
```

Vous pouvez alors appeler la macro :



Appeler des macros dans l'éditeur de formulaires

Lorsque des macros sont définies dans votre projet 4D, vous pouvez appeler une macro à l'aide du menu contextuel de l'éditeur de formulaires :



Ce menu est créé selon le(s) [fichier\(s\) de définition de macro](#) `formMacros.json`. Macro items are sorted in alphabetical order.

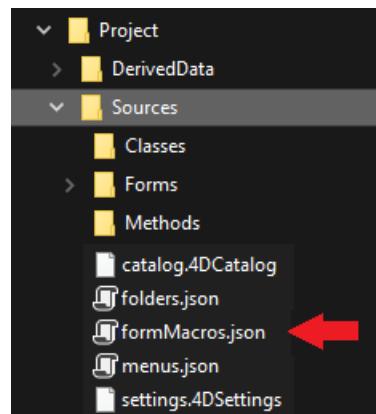
Ce menu peut être appelé dans une zone vide ou une sélection dans le formulaire. Les objets sélectionnés sont passés à `$editor.currentSelection` ou `$editor.target` dans la fonction `onInvoke` de la macro.

Une seule macro peut exécuter plusieurs opérations. Si elle est sélectionnée, la fonction `Annuler` de l'éditeur de formulaires peut être utilisée pour inverser les opérations de macro.

Emplacement du fichier de macro

Toutes les macros de l'éditeur de formulaire 4D sont définies dans un seul fichier JSON par projet ou composant : `FormMacros.json`.

Ce fichier doit se trouver dans le dossier Projet > Sources de l'hôte ou du composant :



Déclaration de macros

La structure du fichier `formMacros.json` est la suivante :

```
{
  "macros": {
    <macroName>: {
      "class": <className>,
      <customProperty> : <value>
    }
  }
}
```

Voici la description du contenu du fichier JSON :

Attribut			Type	Description
macros			object	liste des macros définis
	<macroName>		object	définition de la macro
		class	string	nom de classe de la macro
		<customProperty>	any	(optionnel) valeur personnalisée à récupérer dans le constructeur

Les propriétés personnalisées, lorsqu'elles sont utilisées, sont passées à la fonction [constructeur](#) de la macro.

Exemple

```
{
  "macros": {
    "Open Macros file": {
      "class": "OpenMacro"
    },
    "Align to Right on Target Object": {
      "class": "AlignOnTarget",
      "myParam": "right"
    },
    "Align to Left on Target Object": {
      "class": "AlignOnTarget",
      "myParam": "left"
    }
  }
}
```

Instancier des macros dans 4D

Chaque macro que vous souhaitez instancier dans votre projet ou composant doit être déclarée en tant que [classe 4D](#).

Le nom de la classe doit correspondre au nom défini à l'aide de l'attribut [class](#) du fichier `formMacros.json`.

Les macros sont instanciées au lancement de l'application. Par conséquent, si vous modifiez la structure de la classe de macro (ajouter une fonction, modifier un paramètre, etc.) ou le [constructeur](#), vous devrez redémarrer l'application pour appliquer les modifications.

Fonctions macro

Chaque classe de macro peut contenir un [Class constructor](#) (constructeur de classe) et deux fonctions : [onInvoke\(\)](#) et [onError\(\)](#).

Class constructor

Class constructor(\$macro : Object)

Paramètres	Type	Description
\$macro	Object	Objet de déclaration de macro (dans le fichier <code>formMacros.json</code>)

Les macros sont instanciées à l'aide d'une fonction `class constructor`, le cas échéant.

Le class constructor est appelé une fois lors de l'instanciation de classe, qui se produit au démarrage de l'application.

Les propriétés personnalisées ajoutées à la `déclaration macro` sont retournées dans le paramètre de la fonction class constructor.

Exemple

Dans le fichier `formMacros.json` :

```
{  
  "macros": {  
    "Align to Left on Target Object": {  
      "class": "AlignOnTarget",  
      "myParam": "left"  
    }  
  }  
}
```

Vous pouvez écrire :

```
// Class "AlignOnTarget"  
Class constructor($macro : Object)  
  This.myParameter:=$macro.myParam //gauche  
  ...
```

onInvoke()

onInvoke(\$editor : Object) -> \$result : Object

Paramètres	Type	Description
\$editor	Object	Objet Form Editor Macro Proxy contenant les propriétés du formulaire
\$result	Object	Objet Form Editor Macro Proxy retournant des propriétés modifiées par la macro (facultatif)

La fonction `onInvoke` est automatiquement exécutée à chaque fois que la macro est appelée.

Lorsque la fonction est appelée, elle reçoit dans la propriété `$editor.editor` une copie de tous les éléments du formulaire avec leurs valeurs courantes. Vous pouvez ensuite exécuter n'importe quelle opération sur ces propriétés.

Une fois les opérations terminées, si la macro entraîne la modification, l'ajout ou la suppression d'objets, vous pouvez transmettre les propriétés modifiées résultantes dans `$result`. Le processeur de macros analysera les propriétés retournées et appliquera les opérations nécessaires dans le formulaire. Évidemment, moins vous retournez de propriétés, moins le traitement prendra du temps.

Voici les propriétés retournées dans le paramètre `$editor` :

Propriété	Type	Description
\$editor.editor.form	Object	The entire form
\$editor.editor.file	File	File object of the form file
\$editor.editor.name	Chaine	Name of the form
\$editor.editor.table	number	Table number of the form, 0 for project form
\$editor.editor.currentPageNumber	number	The number of the current page
\$editor.editor.currentPage	Object	The current page, containing all the form objects and the entry order of the page
\$editor.editor.currentSelection	Collection	Collection of names of selected objects
\$editor.editor.formProperties	Object	Properties of the current form
\$editor.editor.target	string	Name of the object under the mouse when clicked on a macro

Here are the properties that you can pass in the `$result` object if you want the macro processor to execute a modification. All properties are optional:

Propriété	Type	Description
currentPage	Object	currentPage including objects modified by the macro, if any
currentSelection	Collection	currentSelection if modified by the macro
formProperties	Object	formProperties if modified by the macro
editor.groups	Object	group info, if groups are modified by the macro
editor.views	Object	view info, if views are modified by the macro
editor.activeView	Chaine	Active view name

Par exemple, si des objets de la page courante et des groupes ont été modifiés, vous pouvez écrire ce qui suit :

```
$result:=New object("currentPage"; $editor.editor.currentPage ; \
"editor"; New object("groups"; $editor.editor.form.editor.groups))
```

attribut `method`

When handling the `method` attribute of form objects, you can define the attribute value in two ways in macros:

- Using a [string containing the method file name/path](#).
- Using an object with the following structure:

Propriété	Type	Description
source	Chaine	Code de la méthode

4D créera un fichier en utilisant le nom de l'objet dans le dossier "objectMethods" avec le contenu de l'attribut `source`. Cette fonctionnalité n'est disponible que pour le code macro.

Propriété `$4dId` dans `currentPage.objects`

La propriété `$4dId` définit un ID unique pour chaque objet de la page courante. Cette clé est utilisée par le processeur de macros pour gérer les modifications dans `$result.currentPage` :

- si la clé `$4dId` est manquante à la fois dans le formulaire et dans un objet dans `$result`, l'objet est créé.
- si la clé `$4dId` existe dans le formulaire mais est manquante dans `$result`, l'objet est supprimé.

- si la clé `$4dId` existe à la fois dans le formulaire et dans un objet dans `$result` l'objet est modifié.

Exemple

You want to define a macro function that will apply the red color and italic font style to any selected object(s).

```
Function onInvoke($editor : Object)->$result : Object
    var $name : Text

    If ($editor.editor.currentSelection.length>0)
        // Définir le contour en rouge et le style en italique pour chaque objet sélectionné
        For each ($name; $editor.editor.currentSelection)
            $editor.editor.currentPage.objects[$name].stroke:="red"
            $editor.editor.currentPage.objects[$name].fontStyle:="italic"

        End for each

    Else
        ALERT("Please select a form object.")
    End if

    // Notifier la modification à 4D
    $result:=New object("currentPage"; $editor.editor.currentPage)
```

onError()

onError(\$editor : Object; \$resultMacro : Object ; \$error : Collection)

Paramètres		Type	Description
\$editor		Object	Object send to <code>onInvoke</code>
\$resultMacro		Object	Object returned by <code>onInvoke</code>
\$error		Collection	Error stack
	[]errorCode	Nombre	Code d'erreur
	[]message	Text	Description of the error
	[]componentSignature	Text	Internal component signature

La fonction `onError` est exécutée lorsque le processeur de macros rencontre une erreur.

Lors de l'exécution d'une macro, si 4D rencontre une erreur qui empêche l'annulation de la macro, il n'exécute pas la macro. C'est le cas par exemple si l'exécution d'une macro se traduirait par :

- supprimer ou modifier un script dont le fichier est en lecture seule.
- créer deux objets avec le même ID interne.

Exemple

In a macro class definition, you can write the following generic error code:

```
Function onError($editor : Object; $resultMacro : Object; $error : Collection)
    var $obj : Object
    var $txt : Text
    $txt:=""

    For each ($obj; $error)
        $txt:=$txt+$obj.message+"\n"
    End for each

    ALERT($txt)
```

Bibliothèques d'objets

Vous pouvez utiliser des bibliothèques d'objets dans vos formulaires. Une bibliothèque d'objets propose une collection d'objets préconfigurés pouvant être utilisés dans vos formulaires par simple copier-coller ou glisser-déposer.

4D propose deux types de bibliothèques d'objets :

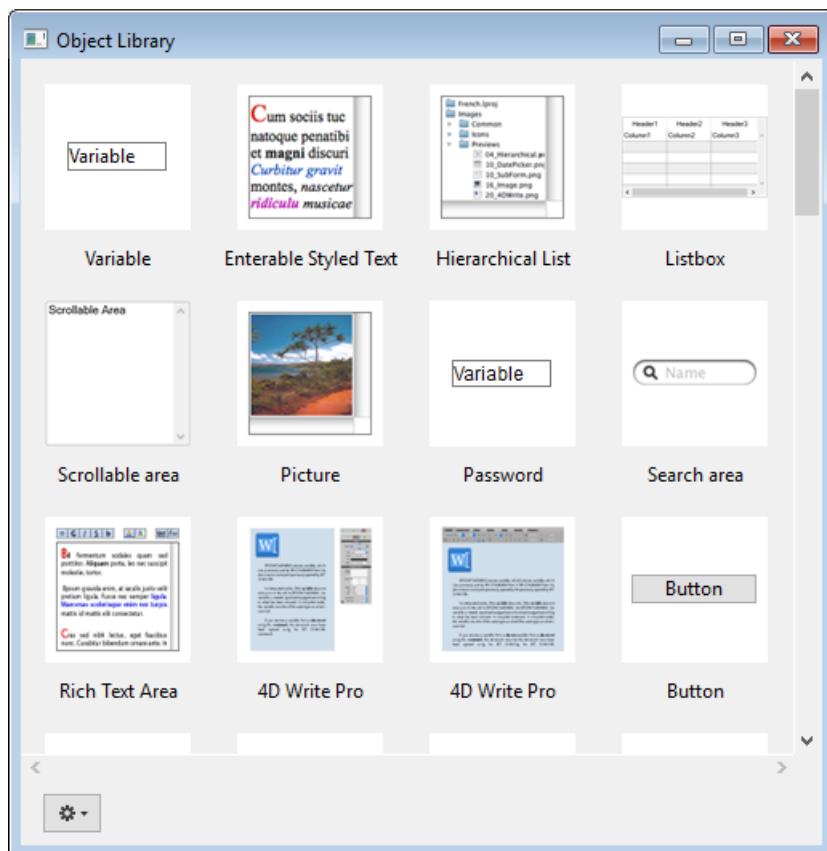
- une bibliothèque d'objets standard préconfigurée, standard, disponible dans tous vos projets.
- des bibliothèques d'objets personnalisées, que vous pouvez utiliser pour stocker vos objets formulaires favoris ou des formulaires projets complets.

Utilisation de la bibliothèque d'objets standard

La bibliothèque d'objets standard est disponible dans l'éditeur de formulaire : cliquez sur le dernier bouton de la barre d'outils :

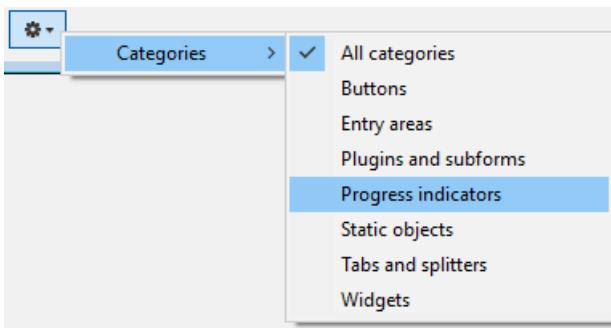


La bibliothèque est affichée dans une fenêtre séparée :



La fenêtre présente les caractéristiques principales suivantes :

- Zone d'aperçu avec des messages d'aide : la zone centrale affiche un aperçu de chaque objet. Vous pouvez survoler un objet pour obtenir des informations sur celui-ci dans un message d'aide.
- Vous pouvez filtrer le contenu de la fenêtre en utilisant le menu Catégories :



- Pour utiliser un objet de la bibliothèque dans votre formulaire, vous pouvez soit :
 - faire un clic droit sur un objet et sélectionnez Copier dans le menu contextuel
 - ou glisser-déposer l'objet de la bibliothèque. L'objet est ensuite ajouté au formulaire.

Cette bibliothèque est en lecture seule. Si vous souhaitez modifier des objets par défaut ou créer votre propre bibliothèque d'objets préconfigurés ou vos formulaires projets, vous devez créer une bibliothèque d'objets personnalisée (voir ci-dessous).

Tous les objets proposés dans la bibliothèque d'objets standard sont décrits dans [cette section sur doc.4d.com](#).

Créer et utiliser des bibliothèques d'objets personnalisées

4D permet de créer et d'utiliser des bibliothèques d'objets personnalisées. Une bibliothèque d'objets personnalisés est un projet 4D dans lequel vous pouvez stocker vos objets favoris (boutons, textes, images, etc.). Vous pouvez ensuite réutiliser ces objets sous différentes formes et dans différents projets.

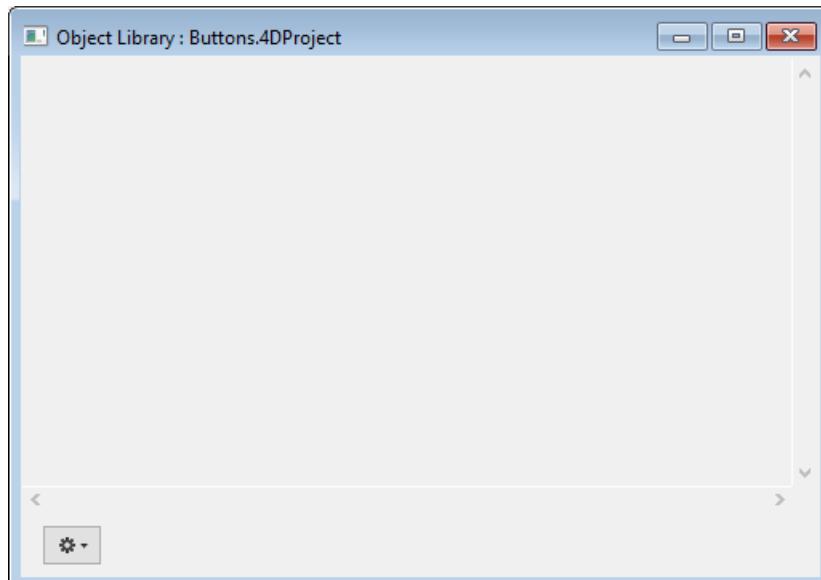
Les objets sont stockés avec toutes leurs propriétés, y compris leurs méthodes objet. Les bibliothèques sont constituées et utilisées par simple glisser-déposer ou copier-coller.

A l'aide des bibliothèques d'objets, vous pouvez constituer des fonds d'objets de formulaires regroupés par familles graphiques, par fonctionnalités, etc.

Créer une bibliothèque d'objets

Pour créer une bibliothèque d'objets, sélectionnez la commande Nouveau > Bibliothèque d'objets... dans le menu Fichier ou dans la barre d'outils de 4D. Une boîte de dialogue standard d'enregistrement de fichiers apparaît, vous permettant de choisir le nom et l'emplacement de la bibliothèque d'objets.

Lorsque vous validez la boîte de dialogue, 4D crée sur disque une nouvelle bibliothèque d'objets et affiche sa fenêtre (vide par défaut).



Vous pouvez créer autant de bibliothèques que vous voulez par projet. Une bibliothèque créée et construite sous macOS peut être utilisée sous Windows et inversement.

Ouvrir une bibliothèque d'objets

Une même bibliothèque d'objets ne peut être ouverte que par un seul projet à la fois. En revanche, il est possible d'ouvrir plusieurs bibliothèques différentes dans le même projet.

Pour ouvrir une bibliothèque d'objets personnalisée, sélectionnez la commande Ouvrir>Bibliothèque d'objets... dans le menu Fichier ou la barre d'outils de 4D. Une boîte de dialogue standard d'ouverture de fichiers apparaît, vous permettant de désigner la bibliothèque d'objets à ouvrir. Vous pouvez sélectionner les types de fichier suivants :

- .4dproject
- .4dz

Les bibliothèques d'objets personnalisées sont des projets 4D classiques. Seules les parties suivantes d'un projet sont exposées lorsqu'il est ouvert en tant que bibliothèque :

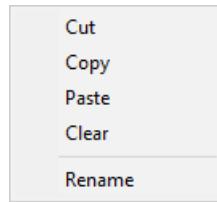
- formulaires projet
- form pages 1

Construire une bibliothèque d'objets

Les objets sont placés dans une bibliothèque d'objets par glisser-déposer ou couper/copier-coller. Ils peuvent provenir soit d'un formulaire soit d'une autre bibliothèque d'objets (y compris la [bibliothèque préconfigurée](#)). Aucun lien n'est conservé avec l'objet d'origine : si celui-ci est modifié, la modification ne sera pas reportée dans l'objet copié.

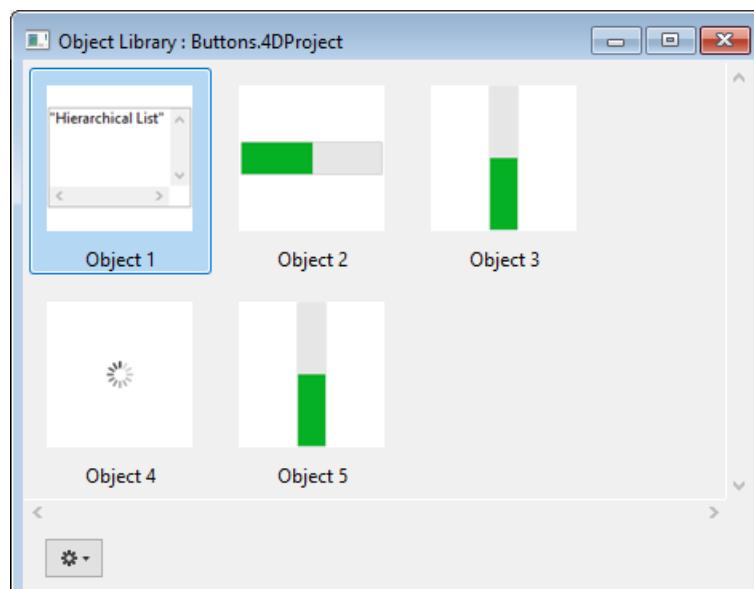
Pour que vous puissiez effectuer des glisser-déposer d'objets depuis les formulaires vers des bibliothèques, vous devez sélectionner l'option de déplacement Commencer un glisser-déposer dans les Préférences de 4D.

Les principales opérations sont accessibles via le menu contextuel ou le menu d'options de la fenêtre :



- **Couper ** ou Copier vers le conteneur de données
- Coller un objet à partir du conteneur de données
- Effacer - supprime l'objet de la bibliothèque
- Renommer - une boîte de dialogue apparaît pour vous permettre de renommer l'élément. A noter que les noms d'objets doivent être uniques dans une bibliothèque.

Vous pouvez placer dans la bibliothèque des objets individuels (y compris des sous-formulaires) ou des ensembles d'objets. Chaque objet ou ensemble d'objets est regroupé en un seul élément :



Une bibliothèque d'objets peut contenir jusqu'à 32 000 éléments.

Les objets sont copiés avec toutes leurs propriétés, graphiques et fonctionnelles, y compris leurs méthodes. Elles sont intégralement conservées lorsque l'élément est recopié dans un formulaire ou une autre bibliothèque.

Objets dépendants

Le copier-coller ou le glisser-déposer de certains objets dans la bibliothèque entraîne également la copie des objets dépendants. Par exemple, la copie d'un bouton entraînera obligatoirement la copie de la méthode objet qui lui est éventuellement attachée. Ces objets dépendants ne peuvent, quant à eux, être directement copiés ou glissés-déposés.

Voici la liste des objets dépendants qui seront collés dans la bibliothèque en même temps que l'objet principal qui les utilise (le cas échéant) :

- Enumérations
- Formats/Filtres
- Images
- Messages d'aide (liés à un champ)
- Méthodes objet

Feuilles de style (style sheets)

Une feuille de style regroupe une combinaison d'attributs d'objets formulaire — allant des attributs de texte à quasiment tous les attributs d'objet disponibles.

Outre l'harmonisation de l'interface de vos applications, l'usage de feuilles de style a trois avantages majeurs :

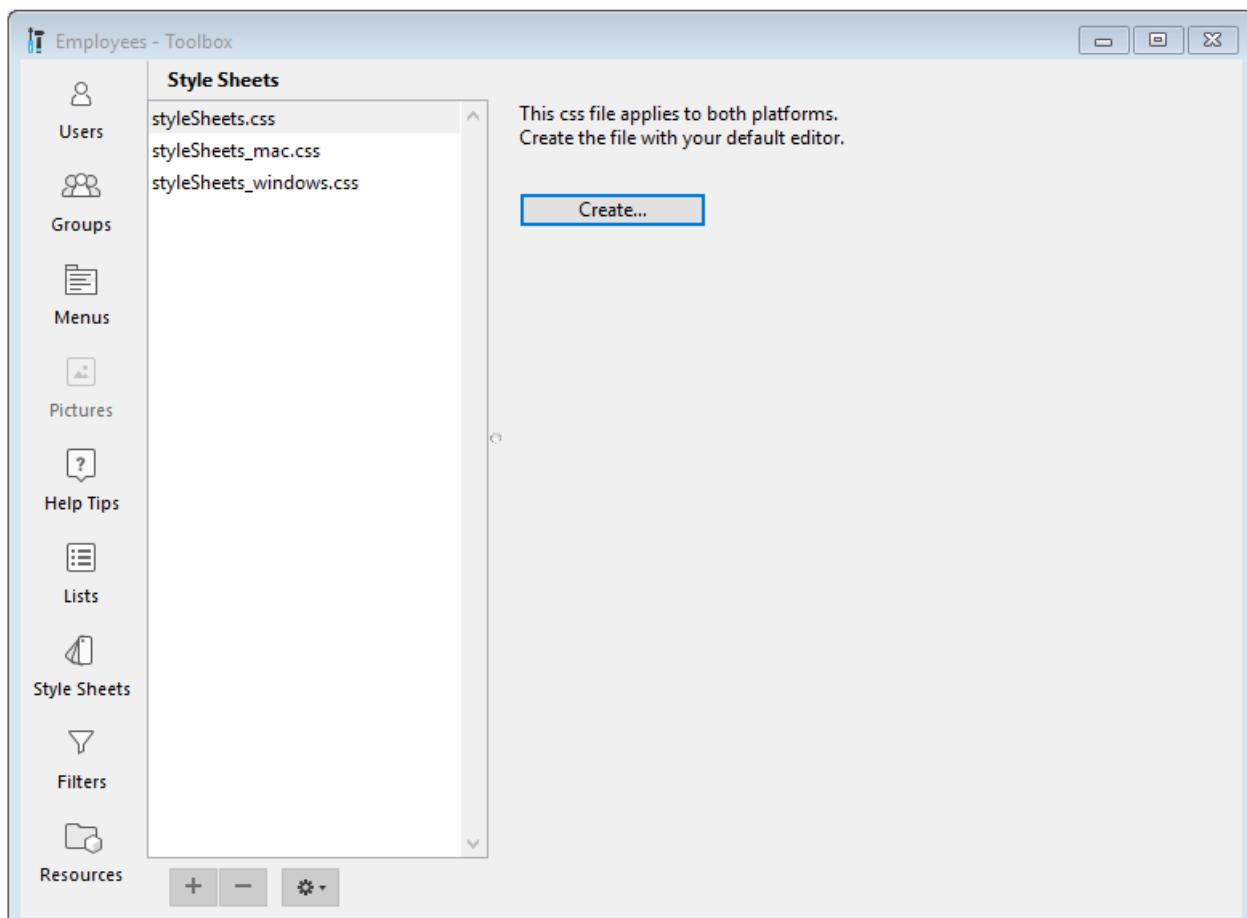
- Gain de temps en développement : pour chaque objet, vous définissez en une seule opération un ensemble de paramétrages.
- Facilité de maintenance : les feuilles de styles ont la propriété de modifier l'apparence de tous les objets qui les utilisent. Changer, par exemple, la taille de la police dans une feuille de style changera la taille de la police pour tous les objets qui utilisent cette feuille de style.
- Contrôle du développement multi-plate-forme : les feuilles de style peuvent s'appliquer aux deux plate-formes macOS et Windows, ou bien à l'une d'elles uniquement. Lorsqu'une feuille de style est appliquée, 4D utilise automatiquement la feuille de style appropriée.

Création ou modification d'une feuille de style

Vous créez des feuilles de styles à partir d'un éditeur de feuilles de styles de votre choix, en sauvegardant le fichier sous une extension ".css" dans le dossier "/SOURCES" du projet.

La Boîte à Outils fournit une page Feuilles de style sous forme de raccourci pour créer et modifier l'une des trois feuilles de style nommées en fonction de la plate-forme.

1. Ouvrez la page Styles en choisissant la **Boîte à outils > Styles ** dans le menu Développement ou en cliquant sur l'icône Boîte à outils dans la barre d'outils de l'éditeur de formulaire.



2. Choisissez le type de feuille de style que vous souhaitez créer et cliquez sur le bouton Crée ou Editer :

Create...

3. La feuille de style s'ouvrira dans votre éditeur de texte par défaut.

Fichiers feuilles de style

4D accepte trois fichiers feuilles de style spécifiques :

Feuille de style	Plate-forme
styleSheets.css	Feuille de style globale par défaut pour macOS et Windows
styleSheets_mac.css	Pour définir des styles d'attributs spécifiques de macOS uniquement
styleSheets_windows.css	Pour définir des styles d'attributs spécifiques pour Windows uniquement

Ces fichiers sont stockés dans le dossier "/SOURCES" du projet. Ils sont également accessibles directement via le [CSS Preview](#) dans la barre d'outils de l'éditeur de formulaires.

Architecture des feuilles de style

While adapted to meet the specific needs of 4D forms, style sheets for application projects generally follow CSS2 syntax and grammar.

Chaque règle de style d'une feuille de style contient deux parties :

- un *sélecteur* - Un sélecteur définit où appliquer le style. 4D prend en charge les sélecteurs "object type", "object name", "class", "all objects", et "attribute value".
- une *déclaration* - La déclaration définit le style à appliquer. Plusieurs lignes de déclaration peuvent être regroupées pour former un bloc de déclaration. Chaque ligne d'un bloc de déclaration CSS doit se terminer par un point-virgule et l'intégralité du bloc doit être entourée d'accolades.

Sélecteurs de feuilles de style

Type d'objet

Le type d'objet définit le type d'objet à styler et correspond au sélecteur d'élément CSS.

Spécifiez le type d'objet, puis entre accolades, déclarez le(s) style(s) à appliquer.

Le type d'objet correspond à la propriété JSON `type` des objets de formulaire.

Dans l'exemple suivant, tous les objets du type *bouton* afficheront du texte dans la police Helvetica Neue, d'une taille de 20 pixels :

```
button {  
    font-family: Helvetica Neue;  
    font-size: 20px;  
}
```

Pour appliquer le même style à plusieurs types d'objets, spécifiez les types d'objets séparés par une "," puis, entre accolades, déclarez le(s) style(s) à appliquer :

```
text, input {  
    text-align: left;  
    stroke: grey;  
}
```

Nom d'objet

Le nom de l'objet correspond au sélecteur d'ID CSS et définit un objet spécifique à styler, puisque que ce nom est

unique dans le formulaire.

Désignez l'objet avec le caractère "#" avant le nom de l'objet, puis entre accolades, déclarez le(s) style(s) à appliquer.

Dans l'exemple suivant, le texte de l'objet portant le nom "okButton" sera affiché dans la police Helvetica Neue, avec une taille de 20 pixels :

```
#okButton {  
    font-family: Helvetica Neue;  
    font-size: 20px;  
}
```

Class

Class correspond au sélecteur class CSS et définit un objet le style de tous les objets formulaires de l'attribut `class`.

Vous pouvez spécifier les classes à utiliser avec un caractère "." suivi du nom de la classe et, entre accolades, déclarez le(s) style(s) à appliquer.

Dans l'exemple suivant, le texte de tous les objets de la classe `okButtons` sera affiché dans la police Helvetica Neue, avec une taille de 20 pixels, alignée au centre :

```
.okButtons {  
    font-family: Helvetica Neue;  
    font-size: 20px;  
    text-align: center;  
}
```

Pour indiquer qu'un style doit être appliqué uniquement à des objets de type différent, spécifiez le type suivi de ":" et du nom de la classe, puis déclarez entre accolades le(s) style(s) à appliquer.

```
text.center {  
    text-align: center;  
    stroke: red;  
}
```

Dans la description du formulaire 4D, vous associez un nom de classe à un objet à l'aide de l'attribut `class`. Cet attribut contient un ou plusieurs noms de "class", séparés par un espace :

```
class: "okButtons important"
```

Tous les objets

Le caractère "*" correspond au sélecteur universel CSS et indique que le style qui suit sera appliqué à tous les objets du formulaire.

Indiquez qu'un style doit s'appliquer à tous les objets de formulaire avec le caractère "*", puis, entre accolades, déclarez le(s) style(s) à appliquer.

Dans l'exemple suivant, tous les objets auront un fond gris :

```
* {  
    fill: gray;  
}
```

Attributs spécifiques

Les styles correspondent aux sélecteurs d'attributs et peuvent s'appliquer à tous les objets du formulaire avec un attribut spécifique.

Spécifiez l'attribut entre parenthèses, puis entre accolades, déclarez le(s) style(s) à appliquer.

Syntaxes prises en charge

Syntaxe	Description
[attribute]	les objets ayant un <code>attribute</code>
[attribute="valeur"]	les objets dont la valeur de l' <code>attribute</code> correspond à la "valeur" indiquée
[attribute~="valeur"]	les objets dont la valeur de <code>attribute</code> correspond à la "valeur" présente dans une liste de mots séparés par des espaces
[attribute = "valeur"]	les objets dont <code>attribute</code> contient une valeur qui commence par celle de "valeur"

Exemples

Tous les objets ayant l'attribut `borderStyle` auront des lignes violettes :

```
[borderStyle]
{
    stroke: purple;
}
```

Tous les objets de type texte ayant un attribut `text` dont la valeur est "Hello" auront des lettres bleues :

```
text[text=Hello]
{
    stroke: blue;
}
```

Tous les objets ayant un attribut `text` dont la valeur est "Hello" auront des traits bleus :

```
[text~="Hello"]
{
    stroke: blue;
}
```

Tous les objets de type texte ayant un attribut `text` dont la valeur commence par "Hello" auront des lettres jaunes :

```
text[text|=Hello]
{
    stroke: yellow;
}
```

Déclarations de feuilles de style

Media Queries

Media queries are used to apply color schemes to an application.

A media query is composed of a media feature and a value (e.g., `<media feature>:<value>`).

Available media features:

- `prefers-color-scheme`

Available media feature expressions:

- light
For using a light scheme
- dark
For using a dark scheme

Color schemes are only supported on macOS.

Exemple

This CSS defines a color combination for text and text background in the light scheme (default) and another combination when the dark scheme is selected:

```
@media (prefers-color-scheme: light) {  
  .textScheme {  
    fill: LightGrey;  
    stroke: Black;  
  }  
}  
  
@media (prefers-color-scheme: dark) {  
  .textScheme {  
    fill: DarkSlateGray;  
    stroke: LightGrey;  
  }  
}
```

Object Attributes

La majorité des attributs d'objet formulaire peuvent être définis dans une feuille de style, à l'exception des attributs suivants : - `method` - `type` - `class` - `event` - `choiceList`, `excludedList`, `labels`, `list`, `requiredList` (list type)

Les attributs d'objet formulaire peuvent être déclarés avec leur [nom JSON](#) en tant qu'attributs CSS (à l'exclusion des types d'objet, méthodes, événements et listes).

Mappage d'attributs

Les attributs répertoriés ci-dessous peuvent accepter le nom 4D ou le nom CSS.

4D	CSS
borderStyle	border-style
border-style	background-color
fontFamily	font-family
fontSize	font-size
fontStyle	font-style
fontWeight	font-weight
stroke	color
textAlign	text-align
textDecoration	text-decoration
verticalAlign	vertical-align

4D-specific values (e.g., `sunken`) are not supported when using CSS attribute names.

Valeurs d'attributs spécifiques

- Pour les attributs `icon`, `picture`, et `customBackgroundPicture` qui prennent en charge un chemin vers une image, la syntaxe est la suivante :

```
icon: url("/RESOURCES/Images/Buttons/edit.png"); /* chemin absolu */
icon: url("edit.png"); /* chemin relatif vers le fichier du formulaire */
```

- Pour `fill`, `stroke`, `alternateFill`, `horizontalLineStroke` et `verticalLineStroke`, trois syntaxes sont prises en charge :
 - Nom la couleur CSS : `fill: red;`
 - Valeur hexadécimale : `fill: #FF0000;`
 - fonction `rgb()` : `fill:rgb(255,0,0)`
- Si une chaîne utilise des caractères interdits en CSS, vous pouvez l'entourer de guillemets simples ou doubles. Par exemple :
 - une référence xliff : `tooltip: ":xiff:CommonMenuFile";`
 - un datasource avec l'expression de champ : `dataSource: "[Table_1:1]ID:1";`

Ordre de priorité

Les projets 4D hiérarchisent les définitions de style en conflit, d'abord par la définition du formulaire, puis par les feuilles de style.

JSON vs Feuille de style

Si un attribut est défini dans la description du formulaire JSON et dans une feuille de style, 4D utilisera la valeur du fichier JSON.

Pour remplacer ce comportement, la valeur du style doit être suivie d'une déclaration `! Important`.

Exemple 1 :

Description du formulaire JSON	Feuille de style	4D affiche
<code>"text": "Button",</code>	<code>text: Edit;</code>	<code>"Button"</code>

Exemple 2 :

Description du formulaire JSON	Feuille de style	4D affiche
"text": "Button",	text: Edit !important;	"Edit"

Feuilles de style multiples

A l'exécution, 4D hiérarchise automatiquement les feuilles de style dans l'ordre suivant :

1. Le formulaire 4D chargera d'abord le fichier CSS par défaut `/SOURCES/styleSheets.css`.
2. Il chargera ensuite le fichier CSS pour la plate-forme courante `/SOURCES/styleSheets_mac.css` ou `/SOURCES/styleSheets_windows.css`.
3. S'il existe, il chargera alors un fichier CSS spécifique défini dans le formulaire JSON :

- o un fichier pour les deux plates-formes :

```
"css": "<path>"
```

- o ou une liste de fichiers pour les deux plates-formes :

```
"css": [  
    "<path1>",  
    "<path2>"  
,
```

- o ou une liste de fichiers par plate-forme :

```
"css": [  
    {"path": "<path>", "media": "mac"},  
    {"path": "<path>", "media": "windows"},  
,
```

Les chemins de fichiers peuvent être relatifs ou absous. * Les chemins relatifs sont résolus par rapport au fichier de description de formulaire JSON. * Pour des raisons de sécurité, seuls les chemins de système de fichiers (filesystem) sont acceptés pour les chemins absous. (ex : "/RESOURCES", "/DATA")

Voir aussi

Voir la présentation vidéo [CSS for 4D Forms](#).

Images

4D inclut une prise en charge spécifique des images utilisées dans vos formulaires.

Formats natifs pris en charge

4D intègre une gestion native des images. Cela signifie que les images sont affichées et stockées dans leur format d'origine, sans interprétation dans 4D. Les spécificités des différents formats (ombrages, zones transparentes...) sont conservées en cas de copier-coller et affichées sans altération. Ce support natif est valable pour toutes les images stockées dans les formulaires 4D : [images statiques](#) collées en mode Développement, images collées dans des [objets de saisie](#) à l'exécution, etc.

Les formats d'image les plus courants sont pris en charge par les deux plates-formes : jpeg, gif, png, tiff, bmp, etc. Sous macOS, le format pdf est également disponible pour l'encodage et le décodage.

La liste complète des formats pris en charge varie en fonction du système d'exploitation et des codecs personnalisés installés sur les postes. Pour savoir quels sont codecs disponibles, vous devez utiliser la commande `PICTURE CODEC LIST` (voir aussi la description du [type de données image](#)).

Format d'image non disponible

Une icône spécifique est affichée pour les images stockées dans un format non disponible sur le poste. L'extension du format manquant est inscrite en bas de l'icône :



L'icône est automatiquement utilisée partout où l'image doit être affichée :

FirstName :	LastName :	Photo :
Elizabeth	Smith	
Gerry	Mc Namara	
Henry	Portier	

Cette icône indique que l'image ne peut être ni affichée ni manipulée localement -- mais elle peut être stockée sans altération pour être affichée sur une autre machine. C'est le cas, par exemple, pour les images PDF sous Windows ou les images au format PICT.

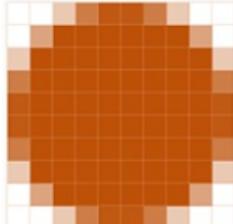
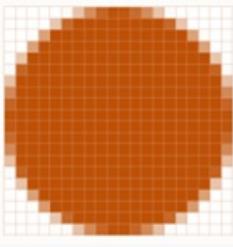
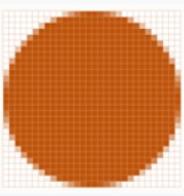
Images de haute résolution

4D prend en charge des images haute résolution sur les plateformes macOS et Windows. Les images haute résolution peuvent être définies par le facteur d'échelle ou le dpi.

Facteur d'échelle

Les écrans haute résolution ont une densité de pixels plus élevée que les écrans standard traditionnels. Pour que les images s'affichent correctement sur des écrans haute résolution, le nombre de pixels de l'image doit être multiplié par le *facteur d'échelle* (c'est-à-dire deux fois plus grand, trois fois plus grand, etc.).

Lorsque vous utilisez des images haute résolution, vous pouvez spécifier le facteur d'échelle en ajoutant "@nx" dans le nom de l'image (où n désigne le facteur d'échelle). Dans le tableau ci-dessous, vous constaterez que le facteur d'échelle est indiqué dans les noms des images haute résolution, *circle@2x.png* et *circle@3x.png*.

Type d'affichage	Facteur d'échelle	Exemple	
Résolution standard	densité de pixel 1:1.	1x 	<i>circle.png</i>
Haute résolution	Densité de pixel augmentée d'un facteur de 2 ou 3.	2x 	3x  <i>circle@2x.png</i> <i>circle@3x.png</i>

Les images haute résolution avec la convention @nx peuvent être utilisées dans les objets suivants :

- [Images statiques](#)
- [Boutons/radio/cases à cocher](#)
- [Boutons image/Pop-up image](#)
- [Onglets](#)
- [En-têtes de list box](#)
- [Icônes de menu](#)

4D priorise automatiquement les images avec la résolution la plus élevée. 4D priorise automatiquement les images avec la résolution la plus élevée. Même si une commande ou une propriété spécifie *circle.png*, *circle@3x.png* sera utilisé (le cas échéant).

A noter que cette résolution se produit uniquement pour l'affichage des images à l'écran, aucune hiérarchisation automatique n'est effectuée lors de l'impression.

DPI

Si 4D donne automatiquement la priorité à la résolution la plus élevée, il existe cependant des différences de comportement en fonction de la résolution de l'écran et de l'image (*) et du format de l'image :

Opération	Comportement
Déposer ou Coller	<p>Si l'image est de :</p> <ul style="list-style-type: none"> • 72dpi ou 96dpi - L'image est "Center" formatée et l'objet contenant l'image contient le même nombre de pixels. • Autre dpi - L'image est formatée "Mise à l'échelle" et l'objet contenant l'image est égal à (nombre de pixels de l'image * dpi de l'écran) / (dpi de l'image) • Pas de dpi - L'image est formatée "Mise à l'échelle".
Taille automatique (menu contextuel de l'éditeur de formulaires)	<p>Si le format d'affichage de l'image est :</p> <ul style="list-style-type: none"> • Scaled - L'objet contenant l'image est redimensionné en fonction de (nombre de pixels de l'image * dpi écran) / (dpi de l'image) • Non mis à l'échelle - L'objet contenant l'image a le même nombre de pixels que l'image.

(*) Généralement, macOS = 72 dpi, Windows = 96 dpi

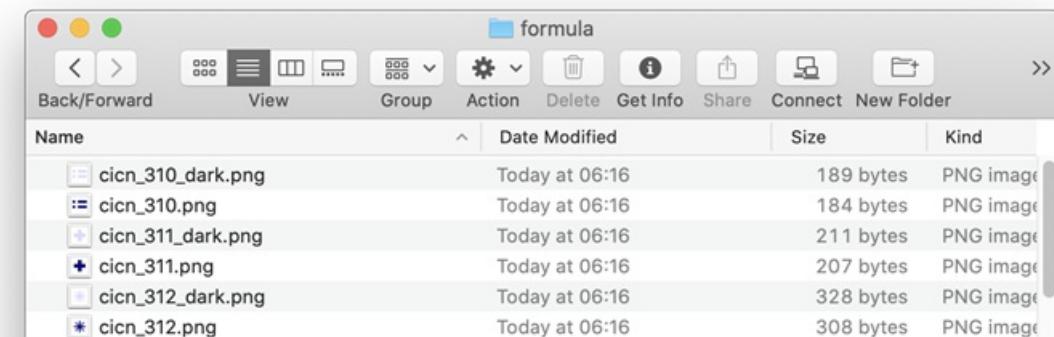
Images en mode sombre (macOS uniquement)

Vous pouvez définir des images et des icônes spécifiques qui seront utilisées à la place des images standard lorsque [les formulaires utilisent le mode sombre](#).

Une image en mode sombre est définie comme suit :

- l'image du mode sombre porte le même nom que la version standard (mode clair) avec le suffixe " _sombre "
- l'image en mode sombre est stockée à côté de la version standard.

Au moment de l'exécution, 4D charge automatiquement l'image en mode clair ou sombre, en fonction du [mode de couleurs du formulaire courant](#).



Coordonnées de la souris dans une image

4D vous permet de récupérer les coordonnées locales de la souris dans un [objet de saisie](#) associé à une [expression d'image](#), en cas de clic ou de survol, même si un défilement ou un zoom a été appliqué à l'image. Ce mécanisme, proche de celui d'une image map, peut être utilisé par exemple pour gérer les barres de bouton défilables ou bien l'interface de logiciels de cartographie.

Les coordonnées sont retournées dans les [Variables système](#) `MouseX` et `MouseY`. Les coordonnées sont exprimées en pixels par rapport à l'angle supérieur gauche de l'image (0,0). Lorsque la souris se trouve en dehors du système de coordonnées de l'image, la valeur -1 est retournée dans `MouseX` et `MouseY`.

Vous pouvez lire la valeur des variables des événements formulaire [On Clicked](#), [On Double Clicked](#), [On Mouse up](#), [On Mouse Enter](#), ou [On Mouse Move](#).

Liste de propriétés JSON

Cette page fournit une liste complète de toutes les propriétés du formulaire, triées par leur nom JSON. Cliquez sur un nom de propriété pour accéder à sa description détaillée.

Dans le chapitre "Propriétés du formulaire", les propriétés sont triées en fonction de leurs noms et thèmes dans la liste des propriétés.

a - c - d - e - f - h - i - m - p - r - s - w

Propriété	Description	Valeurs possibles
a		
bottomMargin	Valeur de la marge verticale (en pixels)	minimum : 0
c		
colorScheme	Mode de couleurs du formulaire	"dark", "light"
d		
destination	Type de formulaire	"detailScreen", "listScreen", "detailPrinter", "listPrinter"
e		
entryOrder	L'ordre dans lequel les objets actifs sont sélectionnés lorsque l'onglet ou la touche retour chariot est utilisé(e) dans un formulaire de saisie	Collection de noms d'objets 4D Form
events	Liste de tous les événements sélectionnés pour l'objet ou le formulaire	Collection de noms d'événements, ex : ["onClick", "onDataChange"...].
f		
formSizeAnchor	Nom de l'objet dont la position détermine la taille du formulaire. (longueur minimale : 1)	Nom d'un objet 4D
h		
height	Hauteur du formulaire	minimum : 0
i		
inheritedForm	Désigne le formulaire à hériter	Nom (chaîne) de la table ou du formulaire projet OU d'un chemin POSIX (chaîne) vers un fichier .json décrivant le formulaire OU un objet décrivant le formulaire
inheritedFormTable	Désigne la table qu'un formulaire hérité utilisera	Un nom ou un numéro de table
m		
markerBody	Position du marqueur de détail	minimum : 0
markerBreak	Position(s) du marqueur de rupture	minimum : 0
markerFooter	Position du marqueur de pied	minimum : 0
markerHeader	Position(s) du marqueur d'en-tête	integer minimum: 0; integer array minimum: 0
registerParams	Enregistre les paramètres du formulaire	true, false

<code>memorizeGeometry</code>	Enregistre les paramètres du formulaire lorsque la fenêtre du formulaire est fermée	true, false Valeurs possibles
<code>menuBar</code>	Barre de menu à associer au formulaire	Nom d'une barre de menus valide
<code>method</code>	Le nom d'une méthode projet.	Le nom d'une méthode projet existante
<code>p</code>		
<code>pages</code>	Collection de pages (chaque page est un objet)	Objets page
<code>pageFormat</code>	object	Propriétés d'impression disponibles
<code>r</code>		
<code>rightMargin</code>	Valeur de la marge horizontale (en pixels)	minimum : 0
<code>s</code>		
<code>shared</code>	Indique si un formulaire peut être utilisé comme sous-formulaire	true, false
<code>w</code>		
<code>width</code>	Largeur du formulaire	minimum : 0
<code>windowMaxHeight</code>	La plus grande hauteur autorisée de la fenêtre de formulaire	minimum : 0
<code>windowMaxWidth</code>	La plus grande largeur autorisée de la fenêtre de formulaire	minimum : 0
<code>windowMinHeight</code>	La plus petite hauteur autorisée de la fenêtre de formulaire	minimum : 0
<code>windowMinWidth</code>	La plus petite largeur autorisée de la fenêtre de formulaire	minimum : 0
<code>windowSizingX</code>	Dimensionnement vertical de la fenêtre de formulaire	"fixed", "variable"
<code>windowSizingY</code>	Dimensionnement horizontal de la fenêtre de formulaire	"fixed", "variable"
<code>windowTitle</code>	Désigne le titre d'une fenêtre de formulaire	Un nom pour la fenêtre de formulaire

Action

Méthode

Référence d'une méthode associée au formulaire. Vous pouvez utiliser une méthode formulaire pour gérer les données et les objets, mais il est généralement plus simple et plus efficace d'utiliser une méthode objet dans ces cas de figure. Voir [Méthodes spécialisées](#).

Vous n'appeliez pas de méthode formulaire - 4D l'appelle automatiquement lorsqu'un événement implique le formulaire auquel la méthode est associée.

Plusieurs types de références de méthode sont pris en charge :

- un chemin de fichier de méthode projet standard, c'est-à-dire qui utilise le modèle suivant :

`method.4dm`

Ce type de référence indique que le fichier de méthode se trouve à l'emplacement par défaut ("sources/{TableForms/numTable} | {Forms}/formName/"). Dans ce cas, 4D gère automatiquement la méthode formulaire lorsque des opérations sont exécutées sur le formulaire (renommage, duplication, copier/coller, etc.)

- a project method name: name of an existing project method without file extension, i.e.: `myMethod` In this case, 4D does not provide automatic support for form operations.
- a custom method file path including the .4dm extension, e.g.:
`MyMethods/myFormMethod.4dm` You can also use a filesystem:
`/RESOURCES/Forms/FormMethod.4dm` In this case, 4D does not provide automatic support for object operations.

Grammaire JSON

Nom	Type de données	Valeurs possibles
method	Texte	Chemin standard ou personnalisé de la méthode formulaire ou nom de la méthode projet

Propriétés des formulaires

Color Scheme

Color scheme property is only applied on macOS.

This property defines the color scheme for the form. This property defines the color scheme for the form. This can be changed for the form to one of the following two options:

- dark - light text on a dark background
- light - dark text on a light background > A defined color scheme can not be overridden by a CSS.

Le nombre de caractères pour un nom de fenêtre est limité à 31.

Grammaire JSON

Nom	Type de données	Valeurs possibles
colorScheme	string	"dark", "light"

Pages

Chaque formulaire est composé d'au moins deux pages :

- une page 0 (page de fond)
- une page 1 (page principale)

Pour plus d'informations, veuillez consulter le thème [Pages formulaire](#).

Grammaire JSON

Nom	Type de données	Valeurs possibles
pages	collection	Collection de pages (chaque page est un objet, la page 0 est le premier élément)

Form Name

This property is the name of the form itself and is used to refer to the form by name using the 4D language. The form name must comply with the [rules specified for identifiers](#) in 4D.

Grammaire JSON

The form name is defined by the name of the folder that contains the form.4Dform file. See [project architecture](#) for more information.

Form Type

The form type, *i.e.* its destination, defines the features that will be available to the form. For example, [markers](#) can only

be set for list (output) table forms.

Each table in a database generally has at least two table forms. One for listing records on-screen and the other for displaying one record at a time (used for data entry and modifications):

- Output form - the *output form* or *list form* displays a list of records, with a single line per record. The results of queries are shown in an output form and users can double-click a line to display the input form for that record.

ID :	name :
1	Friends
3	Work
4	Personal
5	Family

- Input form - used for data entry. It displays a single record per screen and typically has buttons for saving and canceling modifications to the record and for navigating from record to record (i.e., First Record, Last Record, Previous Record, Next Record).

Supported types depend on the form category:

Form Type	JSON grammar	Description	Supported with
Formulaire détaillé	detailScreen	A display form for data entry and modification	Project forms - Table forms
Detail Form for Printing	detailPrinter	A printed report with one page per record, such as an invoice	Project forms - Table forms
List Form	listScreen	A form for listing records on the screen	Table forms
List Form for Printing	listPrinter	A printed report that lists records	Table forms
Aucun	<i>no destination</i>	A form with no specific feature	Project forms - Table forms

Grammaire JSON

Nom	Type de données	Valeurs possibles
destination	string	"detailScreen", "listScreen", "detailPrinter", "listPrinter"

Inherited Form Name

This property designates the [form to inherit](#) in the current form.

To inherit from a table form, set the table in the [Inherited Form Table](#) property.

To remove inheritance, select <None> in the Property List (or " " in JSON).

Grammaire JSON

Nom	Type de données	Valeurs possibles
inheritedForm	string	Name of table or project form OR a POSIX path to a .json file describing the form OR an object describing the form

Inherited Form Table

This property specifies the database table from which to [inherit a form](#) in the current form.

Set to <None> in the Property List (or " " in JSON) to inherit from a project form.

Grammaire JSON

Nom	Type de données	Valeurs possibles
inheritedFormTable	string or number	table name or table number

Publié en tant que sous-formulaire

Pour qu'un formulaire de composant soit sélectionné comme [sous-formulaire](#) dans une base de données hôte, il doit avoir été explicitement partagé. Lorsque cette propriété est sélectionnée, le formulaire sera publié dans le projet hôte.

Seuls les projets formulaire peuvent être indiqués comme étant des sous-formulaires publiés.

Grammaire JSON

Nom	Type de données	Valeurs possibles
shared	boolean	true, false

Mémoriser géométrie

Lorsque cette option est cochée, si la fenêtre est ouverte via la commande `Creer fenetre formulaire` avec le paramètre `*`, plusieurs paramètres du formulaire seront automatiquement mémorisés par 4D au moment de la fermeture de la fenêtre, quelle que soit la manière dont ils ont été modifiés au cours de la session :

- la page courante,
- la position, la taille et la visibilité de chaque objet du formulaire (y compris la taille et la visibilité des colonnes de list box).

Cette option ne prend pas en compte les objets générés via la commande `OBJECT DUPLICATE`. Pour que l'utilisateur retrouve son environnement lors de l'utilisation de cette commande, le développeur doit répéter la séquence de création, définition et positionnement des objets.

Lorsque cette option est cochée, l'option [Mémoriser valeur](#) est en outre disponible pour certains objets.

Grammaire JSON

Nom	Type de données	Valeurs possibles
memorizeGeometry	boolean	true, false

Voir aussi

[Mémoriser valeur](#)

Nom de la fenêtre

Le nom de la fenêtre est utilisé lorsque le formulaire est ouvert à l'aide des commandes `Open form window` et `Open window` dans l'environnement d'application. Le nom de la fenêtre apparaît dans la barre de titre de la fenêtre.

Vous pouvez utiliser des références dynamiques pour définir les noms de fenêtre des formulaires, c'est-à-dire :

- Une référence XLIFF standard stockée dans le dossier Resources.
- Une étiquette de table ou de champ : la syntaxe à appliquer est `<?[TableName]FieldNum>` ou `<?[TableName]FieldName>`. * Une variable ou un champ : La syntaxe à appliquer est \<VariableName> ou `<[TableName]FieldName>`. La valeur du champ ou de la variable sera affichée dans le nom de la fenêtre.

Le nombre de caractères pour un nom de fenêtre est limité à 31.

Grammaire JSON

Nom	Type de données	Valeurs possibles
windowTitle	string	Le nom de la fenêtre sous forme de texte brut ou de référence

Form Size

4D vous permet de définir la taille du formulaire et de la [fenêtre](#). Ces propriétés sont interdépendantes et l'interface de votre application résulte de leur interaction.

Les options de taille dépendent de la valeur de l'option Taille basée sur.

Taille basée sur

- Taille automatique : La taille du formulaire permettra d'afficher tous les objets, auxquels s'ajouteront les valeurs de marge (en pixels) saisies dans

**les champs **Marge hor. et Marge ver.

Vous pouvez choisir cette option lorsque vous souhaitez utiliser des objets actifs placés dans une zone hors écran (c'est-à-dire en dehors du rectangle de délimitation de la fenêtre) avec une fenêtre de taille automatique. Grâce à cette option, la présence de ces objets ne modifiera pas la taille de la fenêtre.

- Fixer taille : La taille du formulaire sera basée sur ce que vous entrez (en pixels) dans les champs [Largeur](#) et [Hauteur](#).
- <object name> : La taille du formulaire sera basée sur la position de l'objet de formulaire sélectionné. Par exemple, si vous choisissez un objet qui est placé dans la partie inférieure droite de la zone à afficher, la taille du formulaire sera constituée d'un rectangle dont le coin supérieur gauche sera l'origine du formulaire et le coin inférieur droit correspondra à celle de l'objet sélectionné, plus les éventuelles valeurs de marge.

Pour les formulaires de sortie, seul les champs Marge **hor. ** ou [Largeur](#) sont disponibles.

Grammaire JSON

Nom	Type de données	Valeurs possibles
formSizeAnchor	string	Nom de l'objet à utiliser pour définir la taille du formulaire

Hauteur

Hauteur du formulaire (en pixels) lorsque la [taille du formulaire](#) est définie sur Fixer taille.

Grammaire JSON

Nom	Type de données	Valeurs possibles
height	number	valeur entier long

Marge hor.

Valeur à ajouter (en pixels) à la marge droite du formulaire lorsque la [taille du formulaire](#) est définie sur Taille automatique ou <object name>

Cette valeur détermine également les marges droites des formulaires utilisés dans l'éditeur d'étiquettes.

Grammaire JSON

Nom	Type de données	Valeurs possibles
rightMargin	number	valeur entier long

Marge hor.

Valeur à ajouter (en pixels) à la marge inférieure du formulaire lorsque la [taille du formulaire](#) est définie sur Taille automatique ou <object name>.

Cette valeur détermine également les marges supérieures des formulaires utilisés dans l'éditeur d'étiquettes.

Grammaire JSON

Nom	Type de données	Valeurs possibles
bottomMargin	number	valeur entier long

Largeur

Largeur du formulaire (en pixels) lorsque la [taille du formulaire](#) est définie sur Fixer taille.

Grammaire JSON

Nom	Type de données	Valeurs possibles
width	number	valeur entier long

Markers

These properties let you specify the precise location of markers on the vertical ruler of a table form. Markers are mainly used in output forms. They control the information that is listed and set header, breaks, detail and footer areas of the form. Any object that placed in these areas is displayed or printed at the appropriate location.

Whenever any form is used for output, either for screen display or printing, the output marker lines take effect and the areas display or print at designated locations. The markers also take effect when a form is used as the List form in a subform area. However, they have no effect when a form is used for input.

Methods that are associated with objects in these areas are executed when the areas are printed or displayed as long as the appropriate events have been activated. For example, a object method placed in the Header area is executed when the `On Header` event takes place.

Form Break

Form Break areas are displayed once at the end of the list of records and are printed once after the records have been printed in a report.

The Break area is defined as the area between the Detail control line and the Break control line. There can be [several Break areas](#) in your report.

You can make Break areas smaller or larger. You can use a Break area to display information that is not part of the records (instructions, current date, current time, etc.), or to display a line or other graphic element that concludes the screen display. In a printed report, you can use a Break area for calculating and printing subtotals and other summary calculations.

Grammaire JSON

Nom	Type de données	Valeurs possibles
markerBreak	integer integer collection	Break marker position or collection of break marker positions in pixels. Minimum value: 0

Form Detail

The form Detail area is displayed on the screen and printed once for each record in a report. The Detail area is defined as the area between the Header control line and the Detail control line.

You can make the Detail area smaller or larger. Whatever you place in the Detail area is displayed or printed once for each record. Most often you place fields or variables in the Detail area so that the information in each record is displayed or printed, but you can place other elements in the Detail area as well.

Grammaire JSON

Nom	Type de données	Valeurs possibles
markerBody	entier	Detail marker position. Minimum : 0

Form Footer

The Form Footer area is displayed on screen under the list of records. It is always printed at the bottom of every page of a report. The Footer area is defined as the area between the Break control line and the Footer control line. You make the Footer area smaller or larger.

You can use the Footer area to print graphics, page numbers, the current date, or any text you want at the bottom of each page of a report. For output forms designed for use on screen, the Footer area typically contains buttons that give the user options such as doing a search or sort, printing records, or putting away the current report. Active objects are accepted.

Grammaire JSON

Nom	Type de données	Valeurs possibles
markerFooter	entier	minimum : 0

Form Header

The form Header area is displayed at the top of each screen and is printed at the top of each page of a report. The Header area is defined as the area above the Header control line. You can make the Header area smaller or larger. You can use the Header area for column names, for instructions, additional information, or even a graphic such as a company logo or a decorative pattern.

You can also place and use active objects in the Header area of output forms displayed as subforms, in the records display window or using the `DISPLAY SELECTION` and `MODIFY SELECTION` commands. The following active objects can be inserted:

- Buttons, picture buttons,
- Combo boxes, drop-down lists, picture pop-up menus,
- hierarchical lists, list boxes
- Radio buttons, check boxes, 3D check boxes,
- Progress indicators, rulers, steppers, spinners.

Standard actions such as `Add Subrecord`, `Cancel` (lists displayed using `DISPLAY SELECTION` and `MODIFY SELECTION`) or `Automatic splitter` can be assigned to the inserted buttons. The following events apply to the active objects you insert in the Header area: `On Load`, `On Clicked`, `On Header`, `On Printing Footer`, `On Double Clicked`, `On Drop`, `On Drag Over`, `On Unload`. Keep in mind that the form method is called with the `On Header` event after calling the object methods of the area.

The form can contain [additional header areas](#) to be associated with additional breaks. A level 1 Header is printed just before the records grouped by the first sorted field are printed.

Grammaire JSON

Nom	Type de données	Valeurs possibles
markerHeader	integer integer collection	Header marker position or collection of header marker positions in pixels. Minimum value: 0

Additional areas

You can create additional Break areas and Header areas for a report. These additional areas allow you to print subtotals and other calculations in a report and to display other information effectively.

Additional areas are defined when you use a collection of positions in the [Form Break](#) and [Form Header](#) properties.

In the 4D Form editor, you create additional control lines by holding down the Alt key while clicking the appropriate control marker.

A form always starts with a Header, Detail, Break level 0, and Footer areas.

Break at level 0 zero takes in all the records; it occurs after all the records are printed. Additional Break areas can be

added, i.e. a Break level 1, Break level 2, etc.

A Break level 1 occurs after the records grouped by the first sorted field are printed.

Label	Description	Prints after groups created by:
Form Break 1	Break at level 1	First sorted field
Form Break 2	Break at level 2	Second sorted field
Form Break 3	Break at level 3	Third sorted field

Additional Header areas are associated with Breaks. A level 1 Header is printed just before the records grouped by the first sorted field are printed.

Label	Description	Prints after groups created by:
Form Header 1	Header at level 1	First sorted field
Form Header 2	Header at level 2	Second sorted field
Form Header 3	Header at level 3	Third sorted field

If you use the `Subtotal` function to initiate Break processing, you should create a Break area for every level of Break that will be generated by the sort order, minus one. If you do not need anything printed in one of the Break areas, you can reduce its size to nothing by placing its marker on top of another control line. If you have more sort levels than Break areas, the last Break area will be repeated during printing.

Menu

Barre de menus associée

Lorsqu'une barre de menus est associée à un formulaire, elle est ajoutée à droite de la barre de menus courante lorsque le formulaire est affiché dans l'environnement d'Application.

La sélection d'une commande de menu entraîne l'envoi d'un événement `Sur menu sélectionné` à la méthode formulaire; vous pouvez ensuite utiliser la commande `Menu selected` pour tester le menu sélectionné.

Si la barre de menus du formulaire est identique à la barre de menus courante, elle n'est pas ajoutée.

La barre de menus du formulaire fonctionnera pour les formulaires d'entrée et de sortie.

Grammaire JSON

Nom	Type de données	Valeurs possibles
menuBar	string	Nom d'une barre de menu

Imprimer

Settings

Permet de définir des paramètres d'impression spécifiques pour le formulaire. Cette fonctionnalité est utile pour afficher les limites de pages d'impression dans l'éditeur de formulaire.

Compatibilité : Même si ces paramètres sont pris en compte lors de l'impression du formulaire en mode Application, il est déconseillé de s'appuyer sur cette fonctionnalité pour stocker les paramètres d'impression du formulaire, en raison des limitations liées à la plateforme et au pilote. It is highly recommended to use the 4D commands `Print settings to BLOB / BLOB to print settings` which are more powerful.

You can modify the following print settings:

- Paper format
- Paper orientation
- Page scaling

Available options depend on the system configuration.

Grammaire JSON

Nom	Type de données	Valeurs possibles
pageFormat	object	Available print properties: <code>paperName</code> , <code>paperWidth</code> , <code>paperHeight</code> , <code>orientation</code> , <code>scale</code>
paperName	string	"A4", "US Letter"...
paperWidth	string	Used if a paper named <code>paperName</code> was not found. Requires unit suffix: pt, in, mm, cm.
paperHeight	string	Used if a paper named <code>paperName</code> was not found. Requires unit suffix: pt, in, mm, cm.
orientation	string	"landscape" (default is "portrait")
scale	number	minimum : 0

Window Size

Hauteur fixe

Si vous cochez cette option, la hauteur de la fenêtre sera verrouillée et l'utilisateur ne pourra plus la redimensionner.

Si cette option n'est pas cochée, la largeur de la fenêtre du formulaire peut être modifiée. Dans ce cas, les propriétés [Hauteur mini](#) et [Hauteur maxi](#) peuvent être utilisées pour déterminer les limites de redimensionnement.

Grammaire JSON

Nom	Type de données	Valeurs possibles
windowSizingY	string	"fixed", "variable"

Largeur fixe

Si vous cochez cette option, la largeur de la fenêtre sera verrouillée et l'utilisateur ne pourra plus la redimensionner.

Si cette option n'est pas cochée, la largeur de la fenêtre du formulaire peut être modifiée. Dans ce cas, les propriétés [Largeur mini](#) et [Largeur maxi](#) peuvent être utilisées pour déterminer les limites de redimensionnement.

Grammaire JSON

Nom	Type de données	Valeurs possibles
windowSizingX	string	"fixed", "variable"

Hauteur maxi, Hauteur mini

Hauteur maximale et minimale (en pixels) d'une fenêtre de formulaire redimensionnable si l'option [Hauteur fixe](#) n'est pas définie.

Grammaire JSON

Nom	Type de données	Valeurs possibles
windowMinHeight	number	valeur entier long
windowMaxHeight	number	valeur entier long

Largeur maxi, Largeur mini

Largeur maximale et minimale (en pixels) d'une fenêtre de formulaire redimensionnable si l'option [Largeur fixe](#) n'est pas définie.

Grammaire JSON

Nom	Type de données	Valeurs possibles
windowMinWidth	number	valeur entier long
windowMaxWidth	number	valeur entier long

A propos des objets formulaires 4D

Vous créez et personnalisez les formulaires de votre application en manipulant les objets qu'ils contiennent. Vous pouvez ajouter des objets, repositionner des objets, définir des propriétés d'objets, appliquer des règles métier en spécifiant des contraintes de saisie de données ou écrire des méthodes objet qui s'exécutent automatiquement lorsque l'objet est utilisé.

Objets actifs et statiques

Les formulaires 4D prennent en charge un grand nombre d'objets actifs et statiques intégrés :

- Les objets actifs réalisent une tâche ou une fonction de l'interface. Les champs sont des objets actifs. Les autres objets actifs — objets saisissables (variables), combo box, listes déroulantes, boutons image, etc. — stockent des données temporairement en mémoire ou effectuent une tâche telle que l'ouverture d'une boîte de dialogue, l'impression d'un état ou le lancement d'un processus d'arrière-plan.
- Les objets statiques sont généralement utilisés pour le décor, les libellés ou encore l'interface graphique du formulaire. A la différence des objets actifs, les objets statiques ne sont pas associés à des variables. A noter qu'il est possible d'insérer des éléments dynamiques dans les objets statiques.

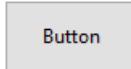
Gérer les objets de formulaire

Vous pouvez ajouter des objets dans un formulaire de nombreuses manières :

- [Form Editor](#): Drag an object from the Form Editor toolbar onto the form. Utilisez ensuite la liste des propriétés pour indiquer les propriétés de l'objet.
Pour plus d'informations, reportez-vous au chapitre [Construction de formulaires](#).
- Langage 4D : Commandes du thème [Objets \(Formulaires\)](#) telles que `OBJECT DUPLICATE` ou `OBJECT SET FONT STYLE` permettent de créer et de définir des objets de formulaire.
- Code JSON dans les formulaires dynamiques : Définissez les propriétés à l'aide du JSON. Utilisez la propriété `type` pour définir le type d'objet puis indiquez ses propriétés. See the [Dynamic Forms](#) page for information.
Example for a button object:

Bouton

Un bouton est un objet actif auquel une action peut être assignée (ex : une tâche de base de données ou une fonction d'interface) pour qu'elle soit réalisée lorsque l'utilisateur clique dessus.



Les boutons peuvent répondre à divers besoins qui dépendent du style et de l'action qui leur est affecté(e). Par exemple, les boutons peuvent amener l'utilisateur à faire des choix ou à compléter un questionnaire ou formulaire. En fonction de leurs propriétés, les bouton peuvent être destinés à être cliqués une fois seulement et à exécuter une commande, ou à être cliqués plusieurs fois pour obtenir le résultat escompté.

Gestion des boutons

Les actions assignées aux boutons peuvent provenir d'[actions standard](#) ou de méthodes objet personnalisées. Les actions typiques peuvent consister à laisser l'utilisateur accepter, annuler ou supprimer des enregistrements, à copier ou coller des données, à passer d'une page à l'autre dans un formulaire de plusieurs pages, à ouvrir, supprimer ou ajouter des enregistrements dans un sous-formulaire, à gérer les attributs de police dans les zones de texte , etc.

Les boutons avec des actions standard sont grisés le cas échéant lors de l'exécution du formulaire. Par exemple, si le premier enregistrement d'une table est affiché, un bouton avec l'action standard `firstRecord` apparaît grisé.

Si vous souhaitez qu'un bouton exécute une action qui n'est pas disponible en tant qu'action standard, laissez le champ d'action standard vide et écrivez une méthode objet pour spécifier l'action du bouton. Pour plus d'informations sur les méthodes d'objet et comment les créer et les associer, voir [Utilisation de méthodes objet](#). En règle générale, vous activez l'événement `Sur clic` et la méthode s'exécute uniquement lorsque vous cliquez sur le bouton. Vous pouvez associer une méthode à n'importe quel bouton.

La [variable](#) associée à un bouton est automatiquement définie sur 0 lorsque le formulaire est exécuté pour la première fois en mode Développement ou Application. Lorsque l'utilisateur clique sur un bouton, sa variable est définie sur 1.

Il est possible d'affecter à la fois une action standard et une méthode à un bouton. Dans ce cas, si le bouton n'est pas désactivé par l'action standard, la méthode est exécutée avant l'action standard.

Styles de bouton

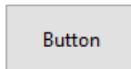
Les styles de bouton contrôlent l'apparence générale d'un bouton ainsi que ses propriétés. Il est possible d'appliquer différents styles prédéfinis aux boutons ou de leur associer des pop-up menus. Plusieurs variantes peuvent être obtenues en combinant ces propriétés/comportements.

À l'exception des [propriétés disponibles](#), de nombreux objets bouton sont *structurellement* identiques. La différence réside dans le traitement de leurs variables associées.

4D propose des boutons dans les styles prédéfinis suivants :

Classique

Le style de bouton Classique est un bouton système standard (c'est-à-dire un rectangle avec un libellé descriptif) qui exécute le code lorsqu'un utilisateur clique dessus.



Par défaut, le style Classique a un fond gris clair avec un libellé au centre. Lorsque le curseur survole le style de bouton Classique, la bordure et la l'arrière-plan changent de couleur. En plus de lancer l'exécution de code, le style de bouton Classique imite un bouton mécanique en changeant rapidement la couleur d'arrière-plan lorsque vous cliquez dessus.

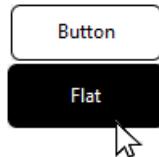
Exemple JSON :

```
"myButton": {  
    "type": "button", //définit le type d'objet  
    "style": "regular", //définit le style du bouton  
    "defaultButton": "true" //définit le bouton comme choix par défaut  
    "text": "OK", //texte à faire apparaître dans le bouton  
    "action": "Annuler", //Action à exécuter  
    "left": 60, //Position gauche dans le formulaire  
    "top": 160, //Position supérieure dans le formulaire  
    "width": 100, //largeur du bouton  
    "height": 20 //hauteur du bouton  
}
```

Seuls les styles Classique et A plat proposent la propriété [Bouton par défaut](#).

A plat

Le style de bouton A plat est un bouton système standard (c'est-à-dire un rectangle avec un libellé descriptif) qui exécute le code lorsqu'un utilisateur clique dessus.



Par défaut, le style A plat a un arrière-plan avec un libellé au centre, des bords arrondis et un design minimaliste. Le style graphique du bouton A plat est particulièrement utile pour les formulaires à imprimer.

Exemple JSON :

```
"myButton": {  
    "type": "button",  
    "style": "flat",  
    "defaultButton": "true"  
    "text": "OK",  
    "action": "Annuler",  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20  
}
```

Seuls les styles Classique et A plat proposent la propriété [Bouton par défaut](#).

Barre d'outils

Le style du bouton Barre d'outils est initialement destiné à être intégré dans une barre d'outils. Il inclut l'option à ajouter à un pop-up menu (indiqué par un triangle inversé) généralement utilisé pour afficher des choix de sélection supplémentaires pour l'utilisateur.

Par défaut, le style bouton Barre d'outils a un fond transparent avec un libellé au centre. En fonction du système d'exploitation, le design du bouton peut changer lorsque la souris le survole :

- *Sous Windows* - le contour du bouton apparaît lorsqu'il dispose de la propriété "Avec pop-up menu", et un triangle est affiché à droite et au centre du bouton.



- *Sous macOS* - le contour du bouton n'apparaît jamais. Lorsqu'il dispose de la propriété "Avec pop up menu", un triangle est affiché à droite et en bas du bouton.

Exemple JSON :

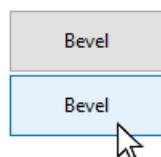
```
"myButton": {  
    "type": "button",  
    "style": "toolbar",  
    "text": "OK",  
    "popupPlacement": "separated",  
    "action": "Annuler",  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20  
}
```

Bevel

Le bouton barre d'outils combine l'apparence du style [Classique](#) (c'est-à-dire un rectangle avec un libellé descriptif) et la propriété de pop-up menu du style [Barre d'outils](#).

Par défaut, le style Bevel a un fond gris clair avec un libellé au centre. En fonction du système d'exploitation, le design du bouton peut changer lorsque la souris le survole :

- *Sous Windows* - le contour du bouton apparaît. Lorsqu'il dispose de la propriété "Avec pop up menu", un triangle est affiché à droite et au centre du bouton.



- *Sous macOS* - le contour du bouton n'apparaît jamais. Lorsqu'il dispose de la propriété "Avec pop up menu", un triangle est affiché à droite et en bas du bouton.

Exemple JSON :

```
"myButton": {  
    "type": "button",  
    "style": "bevel",  
    "text": "OK",  
    "popupPlacement": "linked",  
    "action": "Annuler",  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20  
}
```

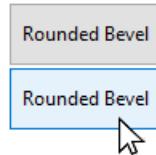
Bevel arrondi

Le style du bouton Bevel arrondi est presque identique au style [Bevel](#), à l'exception des coins du bouton qui peuvent, selon le système d'exploitation, être arrondis. Comme pour le style Bevel, le style Bevel arrondi combine l'apparence du

style [Classique](#) et du style [Barre outils](#).

Par défaut, le style Bevel arrondi a un fond gris clair avec un libellé au centre. En fonction du système d'exploitation, le design du bouton peut changer lorsque la souris le survole :

- *Sous Windows* - le bouton est identique au style Bevel. Lorsqu'il dispose de la propriété "Avec pop up menu", un triangle est affiché à droite et au centre du bouton.



- *Sous macOS* - les coins du bouton sont arrondis. Lorsqu'il dispose de la propriété "Avec pop up menu", un triangle est affiché à droite et en bas du bouton.

Exemple JSON :

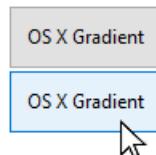
```
"myButton": {  
    "type": "button",  
    "style": "roundedBevel",  
    "text": "OK",  
    "popupPlacement": "none" /  
    "action": "Cancel",  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20  
}
```

OS X Gradient

Le style du bouton OS X Gradient est presque identique au style [Bevel](#). Comme pour le style Bevel, le style OS X Gradient combine l'apparence du style [Classique](#) et du style [Barre outils](#).

Par défaut, le style OS Gradient a un fond gris clair avec un libellé au centre. En fonction du système d'exploitation, le design du bouton peut changer lorsque la souris le survole :

- *Sous Windows* - le bouton est identique au style Bevel. Lorsqu'il dispose de la propriété "Avec pop up menu", un triangle est affiché à droite du bouton.



- *Sous macOS* - le bouton s'affiche comme un bouton à deux tons. Lorsqu'il dispose de la propriété "Avec pop up menu", un triangle est affiché à droite et en bas du bouton.

Exemple JSON :

```

"myButton": {
    "type": "button",
    "style": "gradientBevel",
    "text": "OK",
    "popupPlacement": "linked"
    "action": "Cancel",
    "left": 60,
    "top": 160,
    "width": 100,
    "height": 20
}

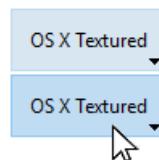
```

OS X Texture

The OS X Textured button style is nearly identical to the [Bevel](#) style but with a smaller size (maximum size is the size of a standard macOS system button). Comme pour le style Bevel, le style OS X Textured combine l'apparence du style [Classique](#) et du style [Barre outils](#).

Par défaut, le style OS X Textured apparaît comme :

- *Sous Windows* - un bouton système standard avec un fond gris clair et un libellé au centre. Il a la particularité d'être transparent dans Vista.



- *Sous macOS* - un bouton système standard affichant un changement de couleur du gris clair au gris foncé. Sa hauteur est prédéfinie : il n'est pas possible de l'agrandir ou de la réduire.

Exemple JSON :

```

"myButton": {
    "type": "button",
    "style": "texturedBevel",
    "text": "OK",
    "popupPlacement": "separated"
    "action": "Cancel",
    "left": 60,
    "top": 160,
    "width": 100,
    "height": 20
}

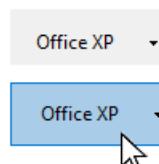
```

Office XP

Le style de bouton Office XP combine l'apparence du style [Classique](#) et du style [Barre outils](#).

Les couleurs (surbrillance et arrière-plan) d'un bouton au style Office XP sont basées sur les couleurs du système. En fonction du système d'exploitation, le design du bouton peut changer lorsque la souris le survole :

- *Sous Windows* - son arrière-plan n'apparaît que lorsque la souris le survole.



- *Sous macOS* - son arrière-plan est toujours affiché.

Exemple JSON :

```
"myButton": {  
    "type": "button",  
    "style": "office",  
    "text": "OK",  
    "popupPlacement": "none"  
    "action": "Cancel",  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20  
}
```

Aide

Le style du bouton Aide peut être utilisé pour afficher un bouton d'aide système standard. Par défaut, le style Aide s'affiche sous la forme d'un point d'interrogation dans un cercle.



Exemple JSON :

```
"myButton": {  
    "type": "button",  
    "style": "help",  
    "text": "OK",  
    "dropping": "custom",  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20  
}
```

Le style Aide ne prend pas en charge les propriétés basiques du [nombre d'états](#), du [chemin d'accès image](#) et de la [position Titre/Image](#).

Rond

Le style de bouton Rond apparaît comme un bouton système circulaire. Ce style de bouton est conçu pour macOS.



Sous Windows, il est identique au style «Aucun» (le cercle en arrière-plan n'est pas pris en compte).

Exemple JSON :

```

"myButton": {
    "type": "button",
    "style": "circular",
    "text": "OK",
    "dropping": "custom",
    "left": 60,
    "top": 160,
    "width": 100,
    "height": 20
}

```

Personnalisé

Le style de bouton Personnalisé accepte une image d'arrière-plan personnalisée et permet de gérer des paramètres supplémentaires tels que la marge et le décalage d'icône.



Exemple JSON :

```

"myButton": {
    "type": "button",
    "style": "custom",
    "text": "",
    "customBackgroundPicture": "/RESOURCES/bkgnd.png",
    "icon": "/RESOURCES/custom.png",
    "textPlacement": "center",
    "left": 60,
    "top": 160,
    "width": 100,
    "height": 20
}

```

Propriétés prises en charge

Tous les boutons partagent une même série de propriétés de base :

Gras - Style de la bordure - Bas - Style de bouton - CSS Class - Déposable - Focusable - Police - Couleur de la police - Taille - Haut - Message d'aide - Dim. horizontal - Italique - Gauche - Non représenté - Nombre d'états(1) - Object Name - Picture pathname(1) - Right - Shortcut - Standard action - Title - Title/Picture Position(1) - Top - Type - Underline - Variable or Expression - Vertical Sizing - Visibility - Width

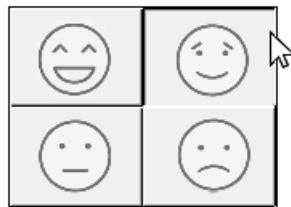
(1) Non pris en charge par le style Aide.

Des propriétés spécifiques supplémentaires sont disponibles, en fonction du [style de bouton](#) :

- [Chemin d'accès arrière-plan](#) - [Marge horizontale](#) - [Décalage icône](#) - [Marge verticale](#) (Personnalisé)
- [Bouton par défaut](#) (A plat, Classique)
- [Avec pop-up menu](#) (Barre outils, Bevel, Bevel arrondi, OS X Gradient, OS X Textured, Office XP, Rond, Personnalisé)

Grille de boutons

Une grille de boutons est un objet transparent placé sur une image. L'image doit correspondre à la forme d'un tableau. Lorsque l'utilisateur clique sur un graphique, ce dernier aura un aspect comprimé :

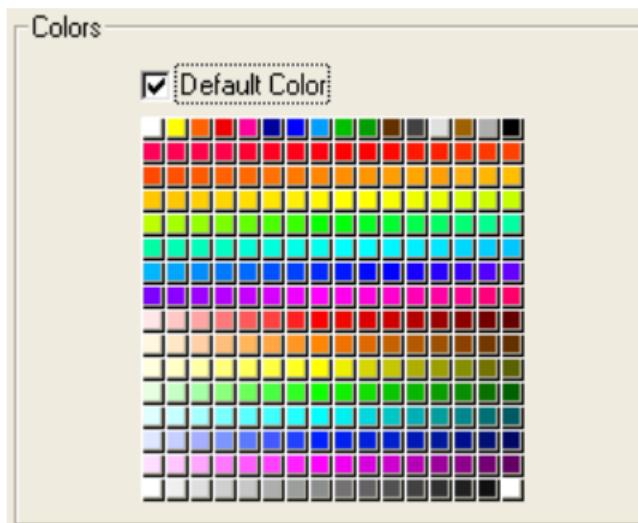


Vous pouvez utiliser une grille de boutons pour déterminer où l'utilisateur clique dans l'image. Votre méthode objet utilise alors l'événement `Sur clic` et gère les actions suivant l'emplacement du clic souris.

Créer une grille de boutons

Pour créer une grille de boutons, placez une image d'arrière-plan puis dessinez une grille de boutons par dessus. Spécifiez le nombre de [lignes](#) et de [colonnes](#).

4D utilise une grille de boutons pour les palettes de couleurs :



Utiliser une grille de boutons

Les boutons de la grille sont numérotés de gauche à droite et de haut en bas, à partir de l'angle supérieur gauche vers l'angle inférieur droit. Dans cet exemple, la grille est dotée de 16 rangées et 16 colonnes. Le bouton situé en haut à gauche est le bouton n° 1. Le dernier bouton de la deuxième rangée est le bouton n° 32. Si aucun élément n'est sélectionné, la valeur est de 0

Aller à page

Vous pouvez associer l'[action standard](#) `Aller à page` à un objet de type Grille de boutons. Lorsque cette action est activée, 4D affiche automatiquement la page du formulaire correspondant au numéro du bouton sélectionné dans la grille de boutons. Par exemple, si l'utilisateur clique sur le 10e bouton de la grille, 4D affichera la page 10 du formulaire courant (si elle existe).

Propriétés prises en charge

[Style de la bordure](#) - [Bas](#) - [Class](#) - [Colonnes](#) - [Hauteur](#) - [Message d'aide](#) - [Dim. horizontal](#) - [Gauche](#) - [Nom](#) - [Droite](#) -

Lignes - Action standard - Haut - Type - Variable ou expression - Dim. vertical - Largeur - Visibilité

Case à cocher

Une case à cocher est un type de bouton utilisée pour saisir ou afficher une donnée binaire (vrai-faux). Fondamentalement, elle est soit cochée, soit décochée, mais [un troisième état](#) peut être défini.



Les cases à cocher sont contrôlées par des méthodes ou des [actions standard](#). La méthode associée à une case à cocher est exécutée lorsqu'elle est cochée. Comme tous les boutons, une case à cocher est initialisée à la valeur zéro lorsque le formulaire est ouvert pour la première fois.

Une case à cocher affiche généralement du texte en face de la case. Ce texte est défini dans la zone [Titre](#) du thème "Objets" de la Liste des propriétés. Vous pouvez saisir dans cette zone un libellé sous forme de référence XLIFF (cf. [Annexe B : Architecture XLIFF](#)).

Utiliser une case à cocher

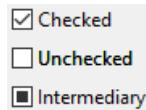
Une case à cocher peut être associée à une [variable ou expression](#) de type entier ou booléen.

- entier : si la case est cochée, la variable prend la valeur 1. Lorsqu'elle n'est pas cochée, elle porte la valeur 0. Si la case à cocher a un troisième état (voir ci-dessous), elle porte la valeur 2.
- booléen : si la case est cochée, la variable prend la valeur `Vrai`. Lorsqu'elle n'est pas cochée, elle prend la valeur `Faux`.

Une partie ou la totalité des cases à cocher contenues dans un formulaires peut être cochée ou non cochée. Plusieurs cases à cocher permettent à l'utilisateur de cocher plusieurs options.

Cases à cocher à trois états

Les objets case à cocher avec le [style de bouton Normal](#) et [Plat](#) acceptent un troisième état. Ce troisième état représente un statut intermédiaire, généralement utilisé pour l'affichage. Il permet par exemple d'indiquer qu'une propriété est présente parmi une sélection d'objets mais pas dans chaque objet de la sélection.



Pour qu'une case à cocher prenne en charge ce troisième état, vous devez lui attribuer la propriété [Trois états](#) dans la Liste des propriétés, thème "Affichage".

Cette propriété n'est disponible que pour les cases à cocher classiques et à plat associées à des [variables ou expressions](#) numériques — les cases à cocher de représentation des expressions booléennes sont exclues de ce principe (une expression booléenne ne pouvant pas se trouver dans un état intermédiaire).

La variable associée à la case à cocher retourne la valeur 2 lorsque celle-ci se trouve dans le troisième état.

En saisie, les cases à cocher à trois états affichent séquentiellement chaque état, dans l'ordre suivant : non coché / coché / intermédiaire / non coché, etc. L'état intermédiaire étant généralement inutile en saisie ; il vous suffit, dans le code, de "forcer" la valeur de la variable à 0 lorsqu'elle prend la valeur 2 afin de passer directement de l'état coché à l'état non coché.

Utiliser une action standard

Vous pouvez affecter une [action standard](#) à une case à cocher pour gérer les attributs des zones de texte. Par exemple, si vous sélectionnez l'action standard `fontBold`, à l'exécution la case à cocher permettra de gérer l'attribut "gras" du texte sélectionné dans la zone de texte courante.

Seules les actions qui peuvent représenter un statut vrai/faux (actions "à coche") sont prises en charge par cet objet :

Actions prises en charge	Conditions d'utilisation (le cas échéant)
avoidPageBreakInsideEnabled	Zones 4D Write Pro uniquement
fontItalic	
fontBold	
fontLinethrough	
fontSubscript	Zones 4D Write Pro uniquement
fontSuperscript	Zones 4D Write Pro uniquement
fontUnderline	
font/showDialog	Mac uniquement
htmlWYSIWIGEnabled	Zones 4D Write Pro uniquement
section/differentFirstPage	Zones 4D Write Pro uniquement
section/differentLeftRightPages	Zones 4D Write Pro uniquement
spell/autoCorrectionEnabled	
spell/autoDashSubstitutionsEnabled	Mac uniquement
spell/autoLanguageEnabled	Mac uniquement
spell/autoQuoteSubstitutionsEnabled	Mac uniquement
spell/autoSubstitutionsEnabled	
spell/enabled	
spell/grammarEnabled	Mac uniquement
spell/showDialog	Mac uniquement
spell/visibleSubstitutions	
visibleBackground	Zones 4D Write Pro uniquement
visibleFooters	Zones 4D Write Pro uniquement
visibleHeaders	Zones 4D Write Pro uniquement
visibleHiddenChars	Zones 4D Write Pro uniquement
visibleHorizontalRuler	Zones 4D Write Pro uniquement
visiblePageFrames	Zones 4D Write Pro uniquement
visibleReferences	
widowAndOrphanControlEnabled	Zones 4D Write Pro uniquement

Pour plus d'informations sur ces actions, veuillez vous reporter à la section [Actions standard](#).

Styles des boutons "Case à cocher"

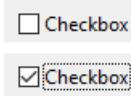
Les cases à cocher utilisent [les styles de bouton](#) pour contrôler l'apparence générale de la case à cocher ainsi que ses propriétés disponibles. Il est possible d'appliquer différents styles prédéfinis aux cases à cocher. Plusieurs variantes peuvent être obtenues en combinant ces propriétés/comportements.

À l'exception des [propriétés disponibles](#), de nombreux objets case à cocher sont *structurellement* identiques. La différence réside dans le traitement de leurs variables associées.

4D propose des cases à cocher avec les styles de bouton prédéfinis suivants :

Classique

Le style Classique du bouton case à cocher correspond à un système de case à cocher standard (i.e., un rectangle avec un titre descriptif) :

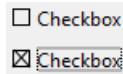


Exemple JSON :

```
"myCheckBox": {  
    "type": "checkbox",  
    "style": "regular",  
    "text": "Cancel",  
    "action": "Cancel",  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20  
    "dataSourceTypeHint": "boolean"  
}
```

A plat

Le style plat du bouton case à cocher a un design minimalist. Le graphisme du style A plat est particulièrement utile pour les formulaires à imprimer.



Exemple JSON :

```
"myCheckBox": {  
    "type": "checkbox",  
    "style": "flat",  
    "text": "Cancel",  
    "action": "cancel",  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20  
}
```

Bouton barre outils

Le style de bouton Barre d'outils est principalement destiné à l'intégration dans une barre d'outils.

Le style de bouton Barre d'outils a un arrière-plan transparent avec un titre. Il est généralement associé à une [image à 4 états](#).

Exemples avec les états coché / non coché / surligné :



Exemple JSON :

```
"myCheckBox": {  
    "type": "checkbox",  
    "style": "toolbar",  
    "text": "Checkbox",  
    "icon": "/RESOURCES/File.png",  
    "iconFrames": 4  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20  
}
```

Bevel

La case à cocher Bevel combine l'apparence du style de bouton [Classique](#) (c'est-à-dire un rectangle avec un libellé descriptif) et le comportement du style de bouton [Barre d'outils](#).

Le style de bouton Bevel possède un fond gris clair et un titre. Il est généralement associé à une [image à 4 états](#).

Exemples avec les états coché / non coché / surligné :



Exemple JSON :

```
"myCheckBox": {  
    "type": "checkbox",  
    "style": "bevel",  
    "text": "Checkbox",  
    "icon": "/RESOURCES/File.png",  
    "iconFrames": 4  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20  
}
```

Bevel arrondi

Le style de bouton de la case à cocher Bevel arrondi est presque identique au style de bouton [Bevel](#), à l'exception des coins du bouton qui peuvent, selon le système d'exploitation, être arrondis. Comme pour le style de bouton Bevel, le style de bouton Bevel arrondi combine l'apparence du style de bouton [Classique](#) et le comportement du style de bouton [Barre outils](#).

Le style de bouton Bevel arrondi possède un fond gris clair et un titre. Il est généralement associé à une [image à 4 états](#).

Exemple sous macOS :



Sous Windows, le style de bouton Bevel arrondi est identique au style de bouton [Bevel](#).

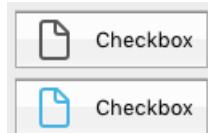
Exemple JSON :

```
"myCheckBox": {  
    "type": "checkbox",  
    "style": "roundedBevel",  
    "text": "Checkbox",  
    "icon": "/RESOURCES/File.png",  
    "iconFrames": 4  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20  
}
```

OS X Gradient

Le style du bouton OS X Gradient est presque identique au style du bouton [Bevel](#). As with the Bevel button style, the OS X Gradient button style combines the appearance of the [Regular](#) button style with the [Toolbar Button](#) button style's behavior.

The OS X Gradient button style has a light gray background with a title and may be displayed as a two-tone system button on macOS. Il est généralement associé à une [image à 4 états](#).



On Windows, this check box button style is identical to the [Bevel](#) button style.

Exemple JSON :

```

"myCheckBox": {
    "type": "checkbox",
    "style": "gradientBevel",
    "text": "Checkbox",
    "icon": "/RESOURCES/File.png",
    "iconFrames": 4
    "left": 60,
    "top": 160,
    "width": 100,
    "height": 20
}

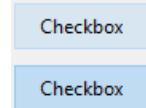
```

OS X Texture

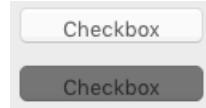
The OS X Textured button style is similar to the [Bevel](#) button style but with a smaller size (maximum size is the size of a standard macOS system button). As with the Bevel button style, the OS X Textured button style combines the appearance of the [Regular](#) button style with the [Toolbar Button](#) button style's behavior.

By default, the OS X Textured button style appears as:

- *Sous Windows* - un bouton système standard avec un fond bleu clair et un libellé au centre.



- *macOS* - a standard system button. Sa hauteur est prédéfinie : il n'est pas possible de l'agrandir ou de la réduire.



Exemple JSON :

```

"myCheckBox": {
    "type": "checkbox",
    "style": "texturedBevel",
    "text": "Checkbox",
    "left": 60,
    "top": 160,
    "width": 100,
    "height": 20
}

```

Office XP

The Office XP button style combines the appearance of the [Regular](#) button style with the [Toolbar Button](#) button style's behavior.

The colors (highlight and background) of a check box with the Office XP button style are based on the system colors. The appearance of the check box can be different when the cursor hovers over it, depending on the OS:

- *Sous Windows* - son arrière-plan n'apparaît que lorsque la souris le survole. Exemples avec les états coché / non coché / surligné :



- *Sous macOS* - son arrière-plan est toujours affiché. Exemples avec les états cochés / non cochés :



Exemple JSON :

```
"myCheckBox": {  
    "type": "checkbox",  
    "style": "office",  
    "text": "Checkbox",  
    "action": "fontBold",  
    "icon": "/RESOURCES/File.png",  
    "iconFrames": 4  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20  
}
```

Contracter/Déployer

This check box button style can be used to add a standard collapse/expand icon. These icons are used natively in hierarchical lists.

- *Windows* - the icon looks like a [+] or a [-]



- *Sous macOS* - il ressemble à un triangle pointant sur vers la droite ou vers le bas.



Exemple JSON :

```
"myCheckBox": {  
    "type": "checkbox",  
    "style": "disclosure",  
    "method": "mCollapse",  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20  
}
```

Bouton disclosure

In macOS and Windows, a check box with the "Disclosure" button style appears as a standard disclosure button, usually used to show/hide additional information. Lorsqu'il est utilisé comme bouton radio, le symbole du bouton pointe vers le bas avec la valeur 0 et vers le haut avec la valeur 1.

- *Sous Windows*



- *macOS*



Exemple JSON :

```
"myCheckBox": {  
    "type": "checkbox",  
    "style": "roundedDisclosure",  
    "method": "m_disclose",  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20  
}
```

Personnalisé

The Custom button style accepts a personalized background picture and allows managing specific properties:

- [Chemin d'accès arrière-plan](#)
- [Décalage icône](#)
- [Marge horizontale](#) and [Marge verticale](#)

Il est généralement associé à une [image à 4 états](#), qui peut être utilisée conjointement avec une image d'arrière-plan [à 4 états](#).

Exemple JSON :

```
"myCheckbox": {  
    "type": "checkbox",  
    "style": "custom",  
    "text": "OK",  
    "icon": "/RESOURCES/smiley.jpg",  
    "iconFrame": 4,  
    "customBackgroundPicture": "/RESOURCES/paper.jpg",  
    "iconOffset": 5, //décalage icône personnalisé au clic  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20,  
    "customBorderX": 20,  
    "customBorderY": 5  
}
```

Propriétés prises en charge

Toutes les cases à cocher partagent une même série de propriétés de base :

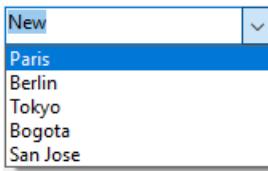
Gras - Bas - Style de bouton - Class - Saisissable - Type d'expression - Focusable - Police - Couleur - Taille - Hauteur - Message d'aide - Dimensionnement horizontal - Italique - Gauche - Nom - Droite - Enregistrer valeur - Raccourci - Action standard - Titre - Haut - Type - Souligné - Variable ou Expression - Dimensionnement vertical - Visibilité - Largeur

Des propriétés spécifiques supplémentaires sont disponibles, en fonction du [style de bouton](#) :

- [Chemin d'accès arrière-plan](#) - [Marge horizontale](#) - [Décalage icône](#) - [Marge verticale](#) (Personnalisé)
- [Trois états](#) (A plat, Classique)
- [Nombre d'états](#) - [Chemin d'accès image](#) - [Position Titre/Image](#) (Bouton barre outils, Bevel, Bevel arrondi, OS X Gradient, OS X Textured, Office XP, Personnalisé)

Combo Box

Une combo box est semblable à une [liste déroulante](#), hormis le fait que cet objet accepte la saisie de texte par l'utilisateur et qu'elle dispose d'options supplémentaires.



Fondamentalement, vous devez considérer l'objet combo box comme une zone saisissable qui utilise un tableau ou une liste de choix en tant que liste de valeurs par défaut.

Handling combo boxes

Use the [On Data Change](#) event to manage entries into the enterable area, as you would for any input form object.

You initialize a combo box in exactly the same way as a [drop-down list](#): using an object, an array, or a choice list.

Using an object

This feature is only available in 4D projects.

An [object](#) encapsulating a [collection](#) can be used as the data source of a combo box. The object must contain the following properties:

Propriété	Type	Description
<code>values</code>	Collection	Mandatory - Collection of scalar values. All values must be of the same type. Supported types: <ul style="list-style-type: none">• chaînes• nombres• dates• times If empty or not defined, the combo box is empty
<code>currentValue</code>	identique à Collection	Text entered by the user

If the object contains other properties, they are ignored.

When the user enters text into the combo box, the `currentValue` property of the object gets the entered text.

Utiliser un tableau

Please refer to Using an array in the [drop-down list page](#) for information about how to initialize the array.

When the user enters text into the combo box, the 0th element of the array gets the entered text.

Utiliser une énumération

If you want to use a combo box to manage the values of an input area (listed field or variable), 4D lets you reference the field or variable directly as the form object's data source. Cette possibilité facilite la gestion des champs/variables énumér(e)s.

Si vous utilisez une énumération hiérarchique, seul le premier niveau sera affiché et sélectionnable.

To associate a combo box with a field or variable, you can just enter the name of the field or variable directly in the [Variable or Expression](#) of the form object in the Property List.

When the form is executed, 4D automatically manages the combo box during input or display: when a user chooses a value, it is saved in the field; this field value is shown in the combo box when the form is displayed:

Please refer to Using a choice in the [drop-down list page](#) for more information.

Options

Combo box type objects accept two specific options:

- [Automatic insertion](#): enables automatically adding a value to the data source when a user enters a value that is not found in the list associated with the combo box.
- [Exclusion](#) (liste de valeurs exclues) : permet d'établir une liste dont les valeurs ne peuvent pas être saisies dans la combo box. Si une valeur exclue est saisie, elle n'est pas acceptée et un message d'erreur s'affiche.

La possibilité d'associer [une liste de valeurs obligatoires](#) n'est pas disponible pour les combo box. In an interface, if an object must propose a finite list of required values, then you must use a [drop-down list](#) object.

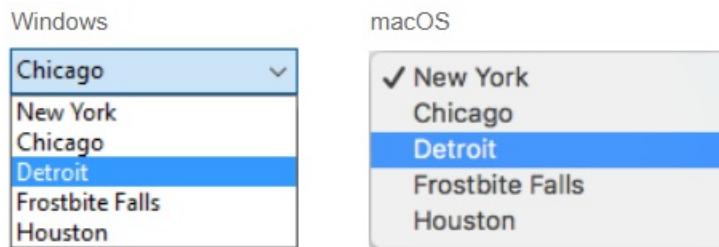
Propriétés prises en charge

[Alpha Format](#) - [Bold](#) - [Bottom](#) - [Choice List](#) - [Class](#) - [Date Format](#) - [Expression Type](#) - [Font](#) - [Font Color](#) - [Font Size](#) - [Height](#) - [Help Tip](#) - [Horizontal Sizing](#) - [Italic](#) - [Left](#) - [Object Name](#) - [Right](#) - [Time Format](#) - [Top](#) - [Type](#) - [Underline](#) - [Variable or Expression](#) - [Vertical Sizing](#) - [Visibility](#) - [Width](#)

Liste déroulante

Drop-down lists are form objects that allow the user to select from a list. You manage the items displayed in the drop-down list using an object, an array, a choice list, or a standard action.

Sous macOS, les listes déroulantes sont aussi parfois appelées "pop-up menu". Les deux noms font référence aux mêmes objets. Comme le montre l'exemple suivant, l'apparence de ces objets peut différer légèrement selon la plateforme :



Drop-down list types

You can create different types of drop-down lists with different features. To define a type, select the appropriate Expression Type and Data Type values in the Property list, or use their JSON equivalent.

Type	Features	Expression Type	Type de données	JSON definition
Object	Built upon a collection	Object	Numeric, Text, Date, or Time	<code>dataSourceTypeHint: object + numberFormat: <format> or textFormat: <format> or dateFormat: <format> or timeFormat: <format></code>
Tableau	Built upon an array	Tableau	Numeric, Text, Date, or Time	<code>dataSourceTypeHint: arrayNumber or arrayText or arrayDate or arrayTime</code>
Choice list saved as value	Built upon a choice list (standard)	List	Selected item value	<code>dataSourceTypeHint: text + saveAs: value</code>
Choice list saved as reference	Built upon a choice list. Item position is saved	List	Selected item reference	<code>dataSourceTypeHint: integer + saveAs: reference</code>
Hierarchical choice list	Can display hierarchical contents	List	List reference	<code>dataSourceTypeHint: integer</code>
Action standard	Automatically built by the action	<code>any</code>	<code>any</code> except List reference	<code>any definition + action: <action> (+ focusable: false for actions applying to other areas)</code>

Handling drop-down lists

Using an object

This feature is only available in 4D projects.

An [object](#) encapsulating a [collection](#) can be used as the data source of a drop-down list. The object must contain the following properties:

Propriété	Type	Description
<code>values</code>	Collection	Mandatory - Collection of scalar values. All values must be of the same type. Supported types: <ul style="list-style-type: none">• chaînes• nombres• dates• times If empty or not defined, the drop-down list is empty
<code>index</code>	number	Index of the currently selected item (value between 0 and <code>collection.length-1</code>). If you set -1, <code>currentValue</code> is displayed as a placeholder string
<code>currentValue</code>	identique à Collection	Currently selected item (used as placeholder value if set by code)

If the object contains other properties, they are ignored.

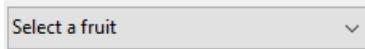
To initialize the object associated to the drop-down list, you can:

- Saisir une liste de valeurs par défaut dans les propriétés de l'objet<Static List>Pour cela, dans le thème [Sources de données](#) de la Liste des propriétés, sélectionnez "". The default values are loaded into an object automatically.
- Execute code that creates the object and its properties. For example, if "myList" is the [variable](#) associated to the drop-down list, you can write in the [On Load](#) form event:

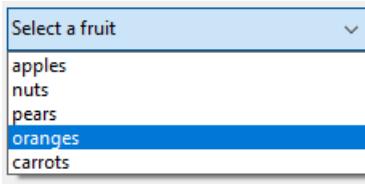
```
// Form.myDrop is the datasource of the form object

Form.myDrop:=New object
Form.myDrop.values:=New collection("apples"; "nuts"; "pears"; "oranges"; "carrots")
Form.myDrop.index:=-1 //currentValue is a placeholder
Form.myDrop.currentValue:="Select a fruit"
```

The drop-down list is displayed with the placeholder string:



After the user selects a value:



```
Form.myDrop.values // ["apples","nuts","pears","oranges","carrots"]
Form.myDrop.currentValue // "oranges"
Form.myDrop.index //3
```

Utiliser un tableau

Un [tableau](#) est une liste de valeurs gardées en mémoire qui sont référencées par le nom du tableau. A drop-down list can display an array as a list of values when you click on it.

To initialize the array associated to the drop-down list, you can:

- Saisir une liste de valeurs par défaut dans les propriétés de l'objet<Static List>Pour cela, dans le thème [Sources de](#)

données de la Liste des propriétés, sélectionnez "". Les valeurs par défaut sont automatiquement chargées dans un tableau. Vous pouvez faire référence à ce tableau par l'intermédiaire du nom de la variable associée à l'objet.

- Avant que l'objet ne soit affiché, exécutez une méthode qui affecte des valeurs au tableau. Par exemple :

```
ARRAY TEXT(aCities;6)
aCities{1}:= "Philadelphia"
aCities{2}:= "Pittsburg"
aCities{3}:= "Grand Blanc"
aCities{4}:= "Bad Axe"
aCities{5}:= "Frostbite Falls"
aCities{6}:= "Green Bay"
```

In this case, the name of the **variable** associated with the object in the form must be **aCities**. Ce code peut être placé dans la méthode formulaire et être exécuté lorsque l'événement formulaire **Sur chargement** se produit.

- Before the object is displayed, load the values of a list into the array using the **LIST TO ARRAY** command. Par exemple :

```
LIST TO ARRAY("Cities";aCities)
```

In this case also, the name of the **variable** associated with the object in the form must be **aCities**. Ce code peut être exécuté à la place de celui proposé plus haut.

Si vous voulez stocker dans un champ le choix de l'utilisateur, il est nécessaire d'écrire du code pour affecter les valeurs et de l'exécuter après la validation de l'enregistrement. Ce code pourrait être le suivant :

```
Case of
:(Form event=On Load)
  LIST TO ARRAY("Cities";aCities)
  If(Record number([People])<0) /Nouvel enregistrement
    aCities:=3 /affiche une valeur par défaut
  Else /enregistrement existant, on affiche une valeur stockée
    aCities:=Find in array(aCities;City)
  End if
:(Form event=On Clicked) /La sélection a été modifiée
  City:=aCities{aCities} /La nouvelle valeur est assignée au champ
:(Form event=On Validate)
  City:=aCities{aCities}
:(Form event=On Unload)
  CLEAR VARIABLE(aCities)
End case
```

You must select each event that you test for in your Case statement. Les tableaux contiennent toujours un nombre fini d'éléments. La liste des éléments est dynamique et peut être modifiée par programmation. Les éléments d'un tableau peuvent être modifiés et triés.

Utiliser une énumération

If you want to use a drop-down list to manage the values of an input area (listed field or variable), 4D lets you reference the field or variable directly as the drop-down list's **data source**. Cette possibilité facilite la gestion des champs/variables énuméré(e)s.

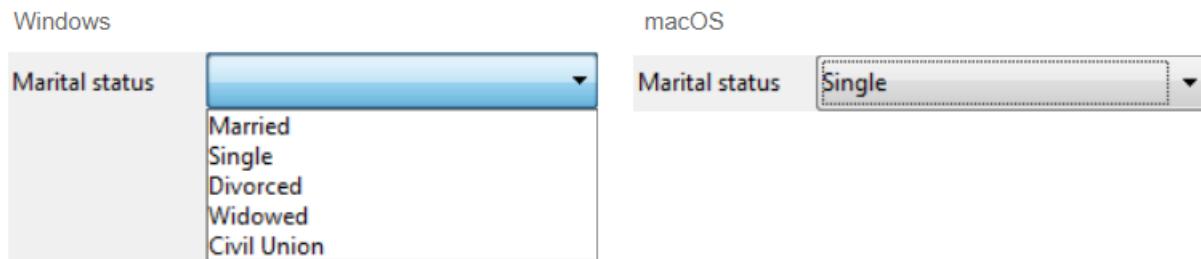
For example, in the case of a "Color" field that can only contain the values "White", "Blue", "Green" or "Red", it is possible to create a list containing these values and associate it with a drop-down list that references the 4D "Color" field. 4D se charge alors de gérer automatiquement la saisie et l'affichage de la valeur courante dans le formulaire.

Si vous utilisez une énumération hiérarchique, seul le premier niveau sera affiché et sélectionnable. Si vous utilisez une énumération hiérarchique, seul le premier niveau sera affiché et sélectionnable.

To associate a drop-down list with a field or variable, enter the name of the field or variable directly as the [Variable or Expression](#) field of the drop-down list in the Property List.

It is not possible to use this feature with an object or an array drop-down list. If you enter a field name in the "Variable or Expression" area, then you must use a choice list.

When the form is executed, 4D automatically manages the drop-down list during input or display: when a user chooses a value, it is saved in the field; this field value is shown in the drop-down list when the form is displayed:

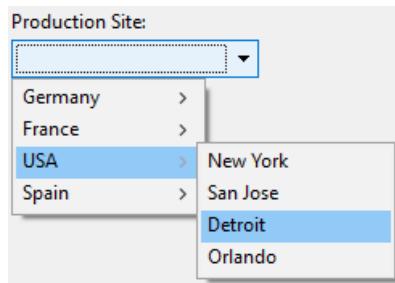


Selected item value or Selected item reference

When you have associated a drop-down list with a choice list and with a field or a variable, you can set the [Data Type](#) property to Selected item value or Selected item reference. Cette option permet d'optimiser la taille des données stockées.

Using a hierarchical choice list

A hierarchical drop-down list has a sublist associated with each item in the list. Here is an example of a hierarchical drop-down list:



In forms, hierarchical drop-down lists are limited to two levels.

You can assign the hierarchical choice list to the drop-down list object using the [Choice List](#) field of the Property List.

You manage hierarchical drop-down lists using the Hierarchical Lists commands of the 4D Language. All commands that support the `(*; "name")` syntax can be used with hierarchical drop-down lists, e.g. [List item parent](#).

Utiliser une action standard

You can build automatically a drop-down list using a [standard action](#). This feature is supported in the following contexts:

- Use of the `gotoPage` standard action. In this case, 4D will automatically display the [page of the form](#) that corresponds to the number of the item that is selected. For example, if the user selects the 3rd item, 4D will display the third page of the current form (if it exists). At runtime, by default the drop-down list displays the page numbers (1, 2...).
- Use of a standard action that displays a sublist of items, for example `backgroundColor`. This feature requires that:
 - a styled text area ([4D Write Pro area](#) or `input` with `multistyle` property) is present in the form as the standard action target.
 - the `focuable` property is not set to the drop-down list. At runtime the drop-down list will display an automatic list of values, e.g. background colors. You can override this automatic list by assigning in addition a choice list in

which each item has been assigned a custom standard action.

This feature cannot be used with a hierarchical drop-down list.

Propriétés prises en charge

[Alpha Format](#) - [Bold](#) - [Bottom](#) - [Button Style](#) - [Choice List](#) - [Class](#) - [Data Type \(expression type\)](#) - [Data Type \(list\)](#) - [Date Format](#) - [Expression Type](#) - [Focusable](#) - [Font](#) - [Font Color](#) - [Font Size](#) - [Height](#) - [Help Tip](#) - [Horizontal Sizing](#) - [Italic](#) - [Left](#) - [Not rendered](#) - [Object Name](#) - [Right](#) - [Standard action](#) - [Save value](#) - [Time Format](#) - [Top](#) - [Type](#) - [Underline](#) - [Variable or Expression](#) - [Vertical Sizing](#) - [Visibility](#) - [Width](#)

Zone de groupe

Une zone de groupe est un objet statique qui vous permet de rassembler visuellement plusieurs objets de formulaire :

The screenshot shows a form element with a title "Employee Info". Inside the group, there are three text input fields: "First name" with value "[Employee]firstName", "Last name" with value "[Employee]lastName", and "Salary" with value "[Employee]salary".

Le nom d'une zone de groupe est un texte statique ; vous pouvez utiliser une référence "localisable", comme pour toute étiquette 4D (voir [Utiliser des références dans les textes statiques](<https://doc.4d.com/4Dv17/4D/17.3/Utiliser-des-references-dans-les-textes-statiques.300-4639972.fr.html>) et la section *Architecture XLIFF* dans le manuel Développement de 4D.

Exemple JSON :

```
"myGroup": {  
    "type": "groupBox",  
    "title": "Employee Info"  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20  
}
```

Propriétés prises en charge

[Bas](#) - [CSS Class](#) - [Police](#) - [Couleur de la police](#) - [Taille](#) - [Hauteur](#) - [Alignement horizontal](#) - [Dim. horizontal](#) - [Italique](#) - [Gauche](#) - [Nom](#) - [Droite](#) - [Titre](#) - [Haut](#) - [Type](#) - [Souligné](#) - [Dim. vertical](#) - [Visibilité](#) - [Largeur](#)

Input

Les zones de saisie vous permettent d'ajouter des expressions saisissables ou non saisissables telles que des [champs](#) et des [variables](#) de base de données à vos formulaires. Les zone de saisie peuvent gérer des données basées sur des caractères (texte, dates, numériques, etc.) ou des images :



Les zones de saisie peuvent contenir [des expressions assignables ou non assignables](#).

De plus, les zones de saisie peuvent être [saisissables ou non saisissables](#). Une zone de saisie saisissable accepte des données. Vous pouvez définir des contrôles de saisie de données pour l'objet. Une zone de saisie non saisissable peut uniquement afficher des valeurs mais ne peut pas être modifiée par l'utilisateur.

Vous pouvez gérer les données avec des [méthodes](#) objet ou formulaire.

Exemple JSON :

```
"myText": {  
    "type": "input",      //définir le type d'objet  
    "spellcheck": true, //autoriser la correction orthographique  
    "left": 60,          //position gauche dans le formulaire  
    "top": 160,          //position haute dans le formulaire  
    "width": 100,        //largeur de l'objet  
    "height": 20         //hauteur de l'objet  
}
```

Propriétés prises en charge

Allow font/color picker - Alpha Format - Auto Spellcheck - Bold - Test when False/Text when True - Border Line Style - Bottom - Choice List - Class - Context Menu - Date Format - Default value - Draggable - Droppable - Enterable - Entry Filter - Excluded List - Expression type - Fill Color - Font - Font Color - Font Size - Height - Hide focus rectangle - Horizontal Alignment - Horizontal Scroll Bar - Horizontal Sizing - Italic - Left - Multiline - Multi-style - Number Format - Object Name - Orientation - Picture Format - Placeholder - Print Frame - Required List - Right - Selection always visible - Store with default style tags - Text when False/Text when True - Time Format - Top - Type - Underline - Variable or Expression - Vertical Scroll Bar - Vertical Sizing - Visibility - Width - Wordwrap

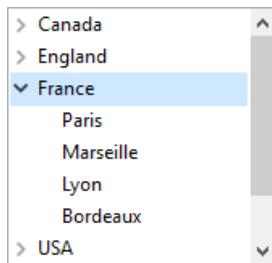
Alternatives

Vous pouvez également représenter des expressions de champ et de variable dans vos formulaires à l'aide d'objets alternatifs, plus particulièrement :

- Vous pouvez afficher et saisir des données à partir des champs de la base de données directement dans des colonnes [de type List box](#).
- Vous pouvez représenter un champ de liste ou une variable directement dans un formulaire à l'aide d'objets [Pop-up menus/Listes déroulantes](#) et [Combo box](#).
- Vous pouvez représenter une expression booléenne sous forme de [case à cocher](#) ou de [bouton radio](#).

Liste hiérarchique

Les listes hiérarchiques sont des objets de formulaire permettant d'afficher des données sous forme de listes comportant un ou plusieurs niveaux qu'il est possible de déployer ou de contracter.



Le cas échéant, l'icône déployer/contractée est automatiquement affichée à gauche de l'élément. Les listes hiérarchiques prennent en charge un nombre illimité de sous-niveaux.

Sources de données de liste hiérarchique

Le contenu d'un objet formulaire liste hiérarchique peut être initialisé de l'une des manières suivantes :

- Associer une [énumération](#) à l'objet. L'énumération doit avoir été définie dans l'éditeur de listes en mode Développement.
- Assignez directement une référence de liste hiérarchique à la [variable ou à l'expression](#) associée à l'objet formulaire.

Dans les deux cas, vous gérez une liste hiérarchique lors de l'exécution via sa référence `ListRef`, à l'aide des commandes de [liste hiérarchique](#) du langage 4D.

RefListe et nom d'objet

Une liste hiérarchique est à la fois un objet de langage existant en mémoire et un objet de formulaire.

L'objet de langage est référencé de manière unique par un identifiant interne, de type Entier long, désigné par `RefListe` dans ce manuel. Cet identifiant est retourné par les commandes permettant de créer des listes `New list`, `Copy list`, `Load list`, `BLOB to list`. Il n'existe qu'une seule instance en mémoire de l'objet de langage et toute modification effectuée sur cet objet est immédiatement répercutée dans tous les endroits où il est utilisé.

L'objet de formulaire n'est pas nécessairement unique : il peut exister plusieurs représentations d'une même liste hiérarchique dans un même formulaire ou dans des formulaires différents. Comme pour les autres objets de formulaire, vous désignez l'objet dans le langage via la syntaxe `(*;"NomListe", etc.)`.

Vous connectez l'"objet de langage" liste hiérarchique avec l'"objet de formulaire" liste hiérarchique par l'intermédiaire de la variable contenant la valeur de l'identifiant unique `RefListe`. Par exemple, si vous avez associé la [variable](#) `mylist` à l'objet de formulaire, vous écrivez :

```
mylist:=New list
```

Chaque représentation de liste dispose de caractéristiques propres et partage des caractéristiques communes avec l'ensemble des représentations. Les caractéristiques propres à chaque représentation de liste sont les suivantes :

- La sélection,
- L'état déployé/contracté des éléments,
- La position du curseur de défilement.

Les autres caractéristiques (police, style, filtre de saisie, couleur, contenu de la liste, icônes, etc.) sont communes à toutes les représentations et ne peuvent pas être modifiées séparément. Par conséquent, lorsque vous utilisez des commandes se basant sur la configuration déployé/contracté ou l'élément courant, par exemple `Count list items` (lorsque le paramètre `* final` n'est pas passé), il importe de pouvoir désigner sans ambiguïté la représentation à

utiliser.

Vous devez utiliser l'identifiant de type `RefListe` avec les commandes du langage lorsque vous souhaitez désigner la liste hiérarchique résidant en mémoire. Par ailleurs, si vous souhaitez désigner la représentation au niveau du formulaire d'un objet Liste hiérarchique, vous devez utiliser le nom de l'objet (type chaîne) dans la commande, via la syntaxe `(*;"NomListe"...)`.

Dans le cas des commandes définissant des propriétés, la syntaxe basée sur le nom d'objet ne signifie pas que seul l'objet de formulaire désigné sera modifié par la commande, mais que l'action de la commande sera basée sur l'état de cet objet. Les caractéristiques communes des listes hiérarchiques sont toujours modifiées dans toutes les représentations. Par exemple, si vous passez l'instruction :

```
SET LIST ITEM FONT(*;"mylist1";*;thefont)
```

... vous indiquez que vous souhaitez modifier la police d'un élément de la liste hiérarchique associée à l'objet de formulaire *mylist1*. La commande tiendra compte de l'élément courant de l'objet *mylist1* pour définir l'élément à modifier, mais cette modification sera reportée dans toutes les représentations de la liste dans tous les process.

Prise en compte du @

Comme pour les autres commandes de gestion des propriété d'objets, il est possible d'utiliser le caractère "@" dans le paramètre `NomListe`. En principe, cette syntaxe permet de désigner un ensemble d'objets dans le formulaire. Toutefois, dans le contexte des commandes de liste hiérarchique, ce principe n'est pas applicable dans tous les cas. Cette syntaxe aura deux effets différents en fonction du type de commande :

- Pour les commandes fixant des propriétés, cette syntaxe désigne tous les objets dont le nom correspond (fonctionnement standard). Par exemple, le paramètre "LH@" désigne tous les objets de type liste hiérarchique dont le nom débute par "LH"
 - `DELETE FROM LIST`
 - `INSERT IN LIST`
 - `SELECT LIST ITEMS BY POSITION`
 - `SET LIST ITEM`
 - `SET LIST ITEM FONT`
 - `SET LIST ITEM ICON`
 - `SET LIST ITEM PARAMETER`
 - `SET LIST ITEM PROPERTIES`
- Pour les commandes récupérant des propriétés, cette syntaxe désigne le premier objet dont le nom correspond. Ces commandes sont :
 - `Count list items`
 - `Find in list`
 - `GET LIST ITEM`
 - `Get list item font`
 - `GET LIST ITEM ICON`
 - `GET LIST ITEM PARAMETER`
 - `GET LIST ITEM PROPERTIES`
 - `List item parent`
 - `List item position`
 - `Selected list items`

Commandes génériques utilisables avec les listes hiérarchiques

Il est possible de modifier l'apparence d'une liste hiérarchique dans un formulaire à l'aide de plusieurs commandes 4D

génériques. Vous devez passer à ces commandes soit le nom d'objet de la liste hiérarchique (en utilisant le paramètre `*`), soit son nom de variable (contenant la valeur `RefListe`) :

- `OBJECT SET FONT`
- `OBJECT SET FONT STYLE`
- `OBJECT SET FONT SIZE`
- `OBJECT SET COLOR`
- `OBJECT SET FILTER`
- `OBJECT SET ENTERABLE`
- `OBJECT SET SCROLLBAR`
- `OBJECT SET SCROLL POSITION`
- `OBJECT SET RGB COLORS`

Rappel : A l'exception de la commande `OBJECT SET SCROLL POSITION`, ces commandes modifient toutes les représentations d'une même liste, même si vous désignez une liste via son nom d'objet.

Priorité des commandes de propriété

Certaines propriétés d'une liste hiérarchique (par exemple l'attribut saisissable ou la couleur) peuvent être définies de trois manières : via la Liste des propriétés en mode Développement, via une commande du thème "Propriétés des objets" ou via une commande du thème "Liste hiérarchique". Lorsque ces trois moyens sont utilisés pour définir les propriétés d'une liste, l'ordre de priorité suivant est appliqué :

1. Commandes du thème "Liste hiérarchique"
2. Commandes générique de propriété d'objet
3. Propriété formulaire

Ce principe est appliqué quel que soit l'ordre d'appel des commandes. Si une propriété d'élément est modifiée individuellement via une commande de liste hiérarchique, la commande de propriété d'objet équivalente sera sans effet sur cet élément même si elle est appelée ultérieurement. Par exemple, si vous modifiez la couleur d'un élément via la commande `SET LIST ITEM PROPERTIES`, la commande `OBJECT SET COLOR` n'aura aucun effet sur cet élément.

Gestion des éléments par position ou par référence

Vous pouvez généralement travailler de deux manières avec le contenu des listes hiérarchiques : par position ou par référence.

- Lorsque vous travaillez par position, 4D se base sur la position relative des éléments dans la liste affichée à l'écran pour les identifier. Le résultat sera différent selon que certains éléments hiérarchiques sont déployés ou non. A noter qu'en cas de multi-représentation, chaque objet de formulaire comporte sa propre configuration d'éléments contractés/déployés.
- Lorsque vous travaillez par référence, 4D se base sur le numéro unique `réfElément` des éléments de la liste. Chaque élément peut être ainsi désigné, quelle que soit sa position ou son affichage dans la liste hiérarchique.

Exploiter les numéros de référence des éléments (`réfElément`)

Chaque élément d'une liste hiérarchique dispose d'un numéro de référence (`réfElément`) de type Entier long. Cette valeur est destinée uniquement à votre propre usage : 4D ne fait que la maintenir.

Attention : Vous pouvez utiliser comme numéro de référence toute valeur de type entier long, sauf la valeur 0. En effet, pour la plupart des commandes de ce thème, la valeur 0 permet de désigner le dernier élément ajouté à la liste.

Voici quelques astuces quant à l'utilisation du numéro de référence unique :

1. Vous n'avez pas besoin d'identifier chaque élément de façon unique (niveau débutant).
 - Premier exemple : vous construisez par programmation un système d'onglets, par exemple, un carnet

d'adresses. Comme le système vous retournera le numéro de l'onglet sélectionné, vous n'aurez probablement pas besoin de davantage d'informations. Dans ce cas, ne vous préoccupez pas des numéros de référence des éléments : passez n'importe quelle valeur (hormis 0) dans le paramètre `réfElément`. Notez que pour un système de carnet d'adresses, vous pouvez prédéfinir une liste A, B,..., Z en mode Développement. Vous pouvez également la créer par programmation afin d'éliminer les lettres pour lesquelles il n'y a pas d'enregistrement.

- Deuxième exemple : en travaillant avec une base, vous construisez progressivement une liste de mots-clés. Vous pouvez sauvegarder la liste à la fin de chaque session, en utilisant les commandes `SAVE LIST` ou `LIST T0 BL0B`, et la recharger au début de chaque session, à l'aide des commandes `Load list` ou `BLOB to list`. Vous pouvez afficher cette liste dans une palette flottante ; lorsque l'utilisateur clique sur un mot-clé de la liste, l'élément choisi est inséré dans la zone saisissable sélectionnée du process de premier plan. En tout état de cause, l'important est que vous ne traitez que l'élément sélectionné (par clic ou glisser-déposer), car la commande `Selected list items` vous retourne la position de l'élément que vous devez traiter. En utilisant cette valeur de position, vous obtenez le libellé de l'élément grâce à la commande `GET LIST ITEM`. Ici aussi, vous n'avez pas besoin d'identifier de façon unique chaque élément ; vous pouvez passer n'importe quelle valeur (hormis 0) dans le paramètre `réfElément`.

2. Identifiez les éléments de la liste (niveau intermédiaire).

Utilisez le numéro de référence de l'élément pour stocker l'information nécessaire lorsque vous devez agir sur un élément ; ce point est détaillé dans l'exemple de la commande `APPEND T0 LIST`. Dans cet exemple, nous utilisons les numéros de référence des éléments pour stocker des numéros d'enregistrements. Cependant, nous devons pouvoir établir une distinction entre les éléments qui correspondent aux enregistrements [Départements] et ceux qui correspondent aux enregistrements [Employés].

3. Identifiez tous les éléments de la liste individuellement (niveau avancé).

Programmez une gestion élaborée des listes hiérarchiques dans lesquelles vous devez absolument pouvoir identifier chaque élément individuellement à tous les niveaux de la liste. Un moyen simple d'implémenter ce fonctionnement est de maintenir un compteur personnel. Supposons que vous créez une liste `hList` à l'aide de la commande `APPEND T0 LIST`. A ce stade, vous initialisez un compteur `vLhCounter` à 1. A chaque fois que vousappelez `APPEND T0 LIST` ou `INSERT IN LIST`, vous incrémentez ce compteur (`vLhCounter:=vLhCounter+1`), et vous passez le compteur comme numéro de référence de l'élément. L'astuce consiste à ne pas décrémenter le compteur lorsque vous détruissez des éléments — le compteur ne peut qu'augmenter. En procédant ainsi, vous gardez l'unicité des numéros de référence des éléments. Puisque ces nombres sont du type Entier long, vous pouvez ajouter ou insérer plus de deux milliards d'éléments dans une liste qui a été réinitialisée ... (cependant si vous travaillez avec un si grand nombre d'éléments, cela signifie généralement que vous devriez utiliser un tableau plutôt qu'une liste.)

Si vous exploitez les Opérateurs sur les bits, vous pouvez également utiliser les numéros de référence des éléments pour stocker des informations qui peuvent être logées dans un Entier long, c'est-à-dire 2 Entiers, des valeurs de 4 octets ou encore 32 Booléens.

Quand avez-vous besoin de numéros de référence uniques ?

Dans la plupart des cas, lorsque vous utilisez des listes hiérarchiques pour des besoins d'interface utilisateur, pour lesquels seul l'élément sélectionné (par un clic ou par glisser-déposer) est important, vous n'avez pas besoin d'utiliser les numéros de référence des éléments. Les commandes `Selected list items` et `GET LIST ITEM` vous fournissent toutes les informations nécessaires à la gestion de l'élément sélectionné. De plus, des commandes telles que `INSERT IN LIST` et `DELETE FROM LIST` vous permettent de manipuler la liste de manière "relative" à l'élément sélectionné.

En pratique, vous devez vous préoccuper des numéros de référence d'éléments lorsque vous voulez accéder directement par programmation à n'importe quel élément de la liste, et pas nécessairement à l'élément couramment sélectionné.

Élément modifiable

Vous pouvez choisir si les éléments de la liste hiérarchique peuvent être modifiés par l'utilisateur à l'aide du raccourci Alt + clic (Windows)/ Option + clic (macOS), ou en effectuant un clic long sur le texte de l'élément.

- Quelle que soit la source de données de la liste hiérarchique, vous pouvez contrôler l'ensemble de l'objet avec la propriété `Saisissable`.
- En outre, si vous remplissez la liste hiérarchique à l'aide d'une liste créée dans l'éditeur de listes, vous contrôlez si

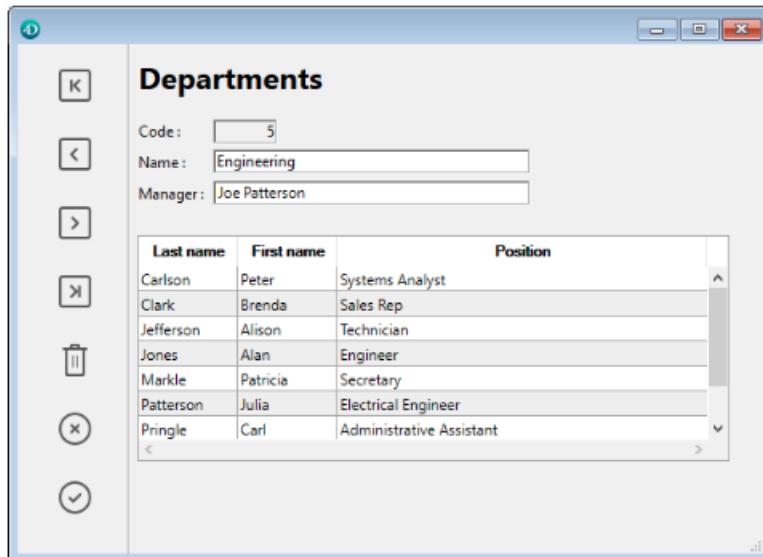
un élément d'une liste hiérarchique est modifiable à l'aide de l'option Élément modifiable dans l'éditeur de listes.
Pour plus d'informations, consultez [Définir les propriétés des énumérations](#) .

Propriétés prises en charge

[Bold](#) - [Border Line Style](#) - [Bottom](#) - [Choice List](#) - [Class](#) - [Draggable](#) - [Droppable](#) - [Enterable](#) - [Entry Filter](#) - [Fill Color](#) - [Focusable](#) - [Font](#) - [Font Color](#) - [Font Size](#) - [Height](#) - [Help Tip](#) - [Hide focus rectangle](#) - [Horizontal Scroll Bar](#) - [Horizontal Sizing](#) - [Italic](#) - [Left](#) - [Multi-selectable](#) - [Object Name](#) - [Right](#) - [Top](#) - [Type](#) - [Underline](#) - [Vertical Scroll Bar](#) - [Vertical Sizing](#) - [Variable or Expression](#) - [Visibility](#) - [Width](#)

List Box

Les list box sont des objets actifs complexes permettant d'afficher et de saisir des données sous forme de tableaux. Ils peuvent être liés au contenu de la base de données, comme les sélections d'entités ("entity selections") et les sections d'enregistrement, ou à tout contenu linguistique tel que les collections et les tableaux. Ils incluent des fonctionnalités avancées concernant la saisie de données, le tri des colonnes, la gestion des événements, l'apparence personnalisée, le déplacement des colonnes, etc.



Une list box contient une ou plusieurs colonnes dont le contenu est automatiquement synchronisé. Le nombre de colonnes est en principe illimité (il dépend des ressources de la machine).

Aperçu

Principes d'utilisation basiques

En exécution, les list box permettent d'afficher et de saisir des données sous forme de listes. Pour passer une cellule en mode édition ([si la saisie est autorisée pour la colonne associée](#)), il suffit de cliquer deux fois sur la valeur qu'elle contient :

Last name	First name
James	Henry
Jameson	Marc

Les utilisateurs peuvent saisir et afficher du texte sur plusieurs lignes au sein d'une cellule de list box. Pour ajouter un retour à la ligne, appuyez sur les touches Ctrl+Retour chariot sous Windows, ou appuyez sur les touches Option+Retour chariot sous macOS.

Les booléens et les images peuvent être affichés dans des cellules, ainsi que des dates, des heures ou des nombres. Il est possible de trier les valeurs de colonne en cliquant sur un en-tête ([tri standard](#)). Toutes les colonnes sont automatiquement synchronisées.

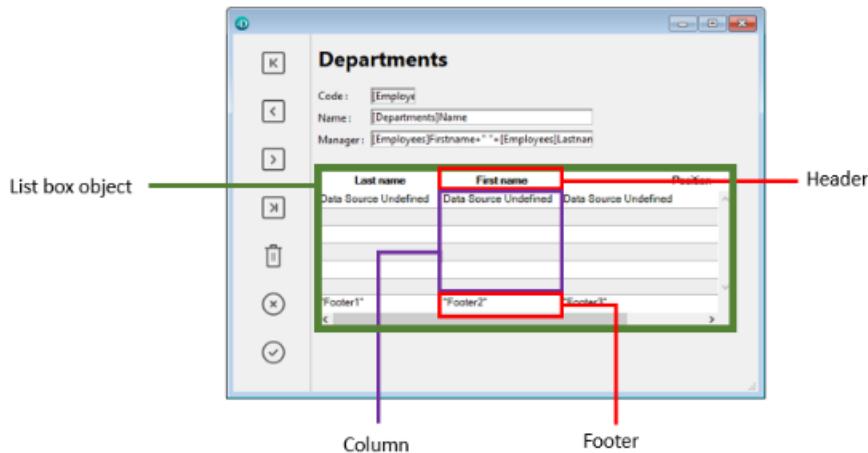
Il est également possible de redimensionner chaque colonne, et l'utilisateur peut modifier l'ordre des [colonnes](#) et des [lignes](#) en les déplaçant à l'aide de la souris, si cette action est autorisée. Notez que les list box peuvent être utilisées [en mode hiérarchique](#).

L'utilisateur peut sélectionner une ou plusieurs lignes à l'aide des raccourcis standard : Maj + clic pour une sélection adjacente et Ctrl + clic (Windows) ou Commande + clic (macOS) pour une sélection non adjacente.

Parties de list box

Une list box est composée de quatre parties distinctes :

- l'objet list box dans sa globalité,
- les colonnes,
- les en-têtes des colonnes, et
- les pieds des colonnes.



Chaque partie dispose de son propre nom d'objet et de propriétés spécifiques. Par exemple, le nombre de colonnes ou la couleur alternée de chaque ligne sont définies dans les propriétés de l'objet list box, la largeur de chaque colonne est définie dans les propriétés des colonnes et la police de l'en-tête est définie dans les propriétés des en-têtes.

Il est possible d'ajouter une méthode objet à l'objet list box et/ou à chaque colonne de la list box. Les méthodes objet sont appelées dans l'ordre suivant :

1. Méthode objet de chaque colonne
2. Méthode objet de la list box

La méthode objet de colonne obtient les événements qui se produisent dans son [en-tête](#) et son [pied](#).

Types de list box

Il existe différents types de list box avec leurs propres comportements et propriétés spécifiques. Le type de list box dépend de sa propriété [Data Source](#) :

- Arrays: each column is bound to a 4D array. Array-based list boxes can be displayed as [hierarchical list boxes](#).
- Selection (Current selection or Named selection): each column is bound to an expression (e.g. a field) which is evaluated for every record of the selection.
- Collection or Entity selection: each column is bound to an expression which is evaluated for every element of the collection or every entity of the entity selection.

It is not possible to combine different list box types in the same list box object. The data source is set when the list box is created. It is then no longer possible to modify it by programming.

Managing list boxes

You can completely configure a list box object through its properties, and you can also manage it dynamically through programming.

The 4D Language includes a dedicated "List Box" theme for list box commands, but commands from various other themes, such as "Object properties" commands or `EDIT ITEM`, `Displayed line number` commands can also be used. Refer to the [List Box Commands Summary](#) page of the *4D Language* reference for more information.

Objets de type List box

List box de type tableau

Dans une list box de type tableau, chaque colonne est associée à un tableau 4D à une dimension ; tous les types de tableaux peuvent être utilisés, à l'exception des tableaux de pointeurs. The number of rows is based on the number of array elements.

By default, 4D assigns the name "ColumnX" to each column. You can change it, as well as other column properties, in the [column properties](#). The display format for each column can also be defined using the [OBJECT SET FORMAT](#) command.

Array type list boxes can be displayed in [hierarchical mode](#), with specific mechanisms.

Avec les list box de type tableau, les valeurs des colonnes (saisie et affichage) sont gérées à l'aide des commandes du langage 4D. You can also associate a [choice list](#) with a column in order to control data entry. Les valeurs des colonnes sont gérées à l'aide des commandes de haut niveau du thème List box (telles que `LISTBOX INSERT ROWS` ou `LISTBOX INSERT COLUMN`) ainsi que des commandes de manipulation des tableaux. Par exemple, pour initialiser le contenu d'une colonne, vous pouvez utiliser l'instruction suivante :

```
ARRAY TEXT(varCol;size)
```

Vous pouvez également utiliser une énumération :

```
LIST TO ARRAY("ListName";varCol)
```

Attention : Lorsqu'un objet List box contient plusieurs colonnes de tailles différentes, seul le nombre d'éléments correspondant au plus petit tableau est affiché. Il est donc conseillé de veiller à ce que chaque tableau ait le même nombre d'éléments que les autres. A noter également que si une colonne de la list box est "vide" (c'est le cas lorsque le tableau associé n'a pas été correctement déclaré ou dimensionné via le langage), la list box n'affiche aucun contenu.

List box de type sélection

Dans ce type de list box, chaque colonne peut être associée à un champ (par exemple `[Employees]LastName`) ou à une expression. The expression can be based on one or more fields (for example, `[Employees]FirstName+ " "` `[Employees]LastName`) or it may simply be a formula (for example `String(Milliseconds)`). The expression can also be a project method, a variable or an array item. You can use the `LISTBOX SET COLUMN FORMULA` and `LISTBOX INSERT COLUMN FORMULA` commands to modify columns programmatically.

The contents of each row is then evaluated according to a selection of records: the current selection of a table or a named selection.

In the case of a list box based on the current selection of a table, any modification done from the database side is automatically reflected in the list box, and vice versa. The current selection is therefore always the same in both places.

List box collection ou entity selection

In this type of list box, each column must be associated to an expression. The contents of each row is then evaluated per collection element or per entity of the entity selection.

Each element of the collection or each entity is available as an object that can be accessed through the `This` keyword. A column expression can be a property path, a project method, a variable, or any formula, accessing each entity or collection element object through `This`, for example `This.<propertyPath>` (or `This.value` in case of a collection of scalar values). You can use the `LISTBOX SET COLUMN FORMULA` and `LISTBOX INSERT COLUMN FORMULA` commands to modify columns programmatically.

When the data source is an entity selection, any modifications made on the list box side are automatically saved in the database. On the other hand, modifications made on the database side are visible in the list box after touched entities have been reloaded.

When the data source is a collection, any modifications made in the list box values are reflected in the collection. On the other hand, if modifications are done on the collection using for example the various functions of the [Collection class](#), you will need to explicitly notify 4D by reassigning the collection variable to itself, so that the list box contents is refreshed. Par exemple :

```
myCol:=myCol.push("new value") //display new value in list box
```

Propriétés prises en charge

Les propriétés prises en charge dépendent du type de list box.

Propriété	List box tableau	Liste box sélection	List box collection ou entity selection
Couleur de fond alternée	X	X	X
Couleur de fond	X	X	X
Gras	X	X	X
Expression couleur de fond		X	X
Style de la bordure	X	X	X
Bas	X	X	X
Class	X	X	X
Collection ou entity selection		X	X
Redimensionnement colonnes auto	X	X	X
Élément courant			X
Position élément courant			X
Source de données	X	X	X
Nom formulaire détaillé		X	
Afficher en-têtes	X	X	X
Afficher pieds	X	X	X
Double-clic sur ligne		X	
Glissable	X	X	X
Déposable	X	X	X
Focusable	X	X	X
Police	X	X	X
Couleur de la police	X	X	X
Expression couleur police		X	X
Taille	X	X	X
Hauteur (list box)	X	X	X
Hauteur (en-têtes)	X	X	X
Hauteur (pieds)	X	X	X
Masquer lignes vides finales	X	X	X

Propriété Cacher rectangle de focus	X	List box tableau	Liste box sélection	List box collection ou entity selection
Cacher surlignage sélection	X	X	X	
List box hiérarchique	X			
Ensemble surlignage		X		
Alignement horizontal	X	X	X	
Couleur lignes horizontales	X	X	X	
Barre de défilement horizontale	X	X	X	
Dimensionnement horizontal	X	X	X	
Italique	X	X	X	
Gauche	X	X	X	
Table principale		X		
Meta info expression				X
Méthode	X	X	X	
Lignes déplaçables	X			
Sélection temporaire		X		
Nombre de colonnes	X	X	X	
Nombre de colonnes verrouillées	X	X	X	
Nombre de colonnes statiques	X	X	X	
Nom d'objet	X	X	X	
Droite	X	X	X	
Tableau couleurs de fond	X			
Tableau de contrôle des lignes	X			
Tableau couleurs de police	X			
Hauteur des lignes	X			
Tableau hauteurs des lignes	X			
Tableau de styles	X			
Eléments sélectionnés				X
Mode de sélection	X	X	X	
Saisie sur clic unique	X	X	X	
Triable	X	X	X	
Action standard	X			
Expression style		X		X
Haut	X	X	X	
Transparent	X	X	X	
Type	X	X	X	
Souligné	X	X	X	
Variable ou expression	X	X		
Alignement vertical	X	X	X	
Couleur lignes verticales	X	X	X	

Barre de défilement verticale	X	List box tableau	X	List box collection ou entity selection
Dimensionnement vertical	X	List box sélection	X	
Visibilité	X	X	X	
Largeur	X	X	X	

Les colonnes, en-têtes et pieds de list box prennent en charge des propriétés spécifiques.

Événements formulaire pris en charge

Événement formulaire	Propriétés supplémentaires retournées (voir Événement formulaire pour les propriétés principales)	Commentaires
Sur après modification	<ul style="list-style-type: none"> • column • columnName • row 	
Sur après frappe clavier	<ul style="list-style-type: none"> • column • columnName • row 	
Sur après tri	<ul style="list-style-type: none"> • column • columnName • headerName 	<i>Les formules composées ne peuvent pas être triées. (ex : This.firstName + This.lastName)</i>
Sur clic alternatif	<ul style="list-style-type: none"> • column • columnName • row 	<i>Listbox tableau uniquement</i>
Sur avant saisie	<ul style="list-style-type: none"> • column • columnName • row 	
Sur avant frappe clavier	<ul style="list-style-type: none"> • column • columnName • row 	
Sur début survol	<ul style="list-style-type: none"> • column • columnName • row 	
Sur clic	<ul style="list-style-type: none"> • column • columnName • row 	
Sur fermeture corps	<ul style="list-style-type: none"> • row 	<i>List box Sélection courante et Sélection temporaire uniquement</i>
Sur contracter	<ul style="list-style-type: none"> • column • columnName • row 	<i>List box hiérarchiques uniquement</i>
Sur déplacement colonne	<ul style="list-style-type: none"> • columnName • newPosition 	

Evénement	<ul style="list-style-type: none"> ● oldPosition <p>Propriétés supplémentaires retournées (voir Événement formulaire pour les propriétés principales)</p> <ul style="list-style-type: none"> ● columnName ● newSize ● oldSize 	Commentaires
Sur données modifiées	<ul style="list-style-type: none"> ● column ● columnName ● row 	
Sur action suppression	<ul style="list-style-type: none"> ● row 	
Sur affichage corps	<ul style="list-style-type: none"> ● isRowSelected ● row 	
Sur double clic	<ul style="list-style-type: none"> ● column ● columnName ● row 	
Sur glisser	<ul style="list-style-type: none"> ● area ● areaName ● column ● columnName ● row 	
Sur déposer	<ul style="list-style-type: none"> ● column ● columnName ● row 	
Sur déployer	<ul style="list-style-type: none"> ● column ● columnName ● row 	<i>List box hiérarchiques uniquement</i>
Sur clic pied	<ul style="list-style-type: none"> ● column ● columnName ● footerName 	<i>List box Tableau, Sélection courante et Sélection temporaire uniquement</i>
On Getting Focus	<ul style="list-style-type: none"> ● column ● columnName ● row 	<i>Propriétés supplémentaires retournées uniquement lors de la modification d'une cellule</i>
Sur clic entête	<ul style="list-style-type: none"> ● column ● columnName ● headerName 	
On Load		
On Losing Focus	<ul style="list-style-type: none"> ● column ● columnName ● row 	<i>Propriétés supplémentaires retournées uniquement lorsque la modification d'une cellule est achevée</i>
Sur début survol	<ul style="list-style-type: none"> ● area ● areaName ● column ● columnName 	

Evénement formulaire	Propriétés supplémentaires retournées (voir Événement formulaire pour les propriétés principales)	Commentaires
Sur fin survol		
Sur survol	<ul style="list-style-type: none"> • area • areaName • column • columnName • row 	
Sur ouverture corps	<ul style="list-style-type: none"> • row 	<i>List box Sélection courante et Sélection temporaire uniquement</i>
Sur déplacement ligne	<ul style="list-style-type: none"> • newPosition • oldPosition 	<i>Listbox tableau uniquement</i>
Sur nouvelle sélection		
Sur défilement	<ul style="list-style-type: none"> • horizontalScroll • verticalScroll 	
On Unload		

Propriétés supplémentaires

Les événements formulaire sur les list box ou colonnes de list box peuvent retourner les propriétés supplémentaires suivantes :

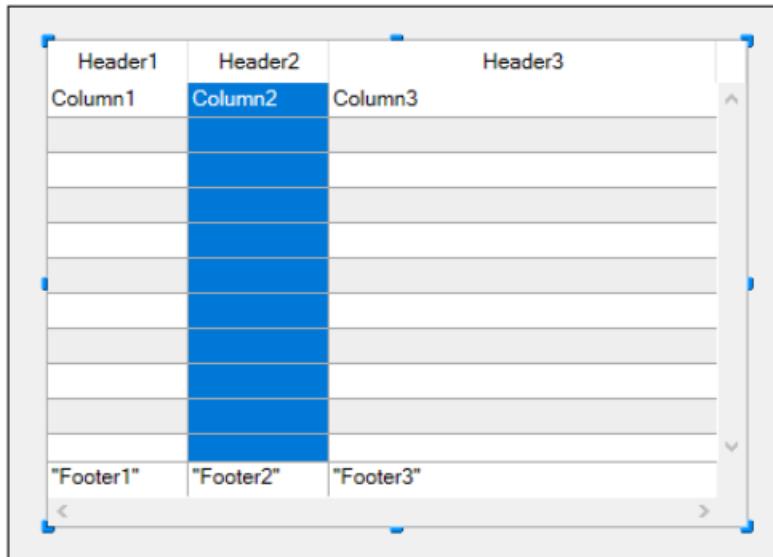
Propriété	Type	Description
area	Texte	List box object area ("header", "footer", "cell")
areaName	Texte	Name of the area
column	entier long	Column number
columnName	Texte	Name of the column
footerName	Texte	Name of the footer
headerName	Texte	Name of the header
horizontalScroll	entier long	Positive if scroll is towards the right, negative if towards the left
isRowSelected	boolean	True if row is selected, else False
newPosition	entier long	New position of the column or row
newSize	entier long	New size (in pixels) of the column or row
oldPosition	entier long	Previous position of the column or row
oldSize	entier long	Previous size (in pixels) of the column or row
row	entier long	Row number
verticalScroll	entier long	Positive if scroll is towards the bottom, negative if towards the top

If an event occurs on a "fake" column or row that doesn't exist, an empty string is typically returned.

Colonnes de list box

A list box is made of one or more column object(s) which have specific properties. You can select a list box column in

the Form editor by clicking on it when the list box object is selected:



You can set standard properties (text, background color, etc.) for each column of the list box; these properties take priority over those of the list box object properties.

You can define the [Expression type](#) for array list box columns (String, Text, Number, Date, Time, Picture, Boolean, or Object).

Propriétés spécifiques des list box

[Alpha Format](#) - [Alternate Background Color](#) - [Automatic Row Height](#) - [Background Color](#) - [Background Color Expression](#) - [Bold](#) - [Choice List](#) - [Class](#) - [Data Type \(selection and collection list box column\)](#) - [Date Format](#) - [Default Values](#) - [Display Type](#) - [Enterable](#) - [Entry Filter](#) - [Excluded List](#) - [Expression](#) - [Expression Type \(array list box column\)](#) - [Font](#) - [Font Color](#) - [Horizontal Alignment](#) - [Italic](#) - [Invisible](#) - [Maximum Width](#) - [Method](#) - [Minimum Width](#) - [Multi-style](#) - [Number Format](#) - [Object Name](#) - [Picture Format](#) - [Resizable](#) - [Required List](#) - [Row Background Color Array](#) - [Row Font Color Array](#) - [Row Style Array](#) - [Save as](#) - [Style Expression](#) - [Text when False/Text when True](#) - [Time Format](#) - [Truncate with ellipsis](#) - [Underline](#) - [Variable or Expression](#) - [Vertical Alignment](#) - [Width](#) - [Wordwrap](#)

Événements formulaire pris en charge

Evénement formulaire	Propriétés supplémentaires renvoyées (voir Événement formulaire pour les propriétés principales)	Commentaires
Sur après modification	<ul style="list-style-type: none">columncolumnNamerow	
Sur après frappe clavier	<ul style="list-style-type: none">columncolumnNamerow	
Sur après tri	<ul style="list-style-type: none">columncolumnNameheaderName	<i>Les formules composées ne peuvent pas être triées. (ex : This.firstName + This.lastName)</i>
Sur clic alternatif	<ul style="list-style-type: none">columncolumnNamerow	<i>Listbox tableau uniquement</i>
Sur avant saisie	<ul style="list-style-type: none">column	

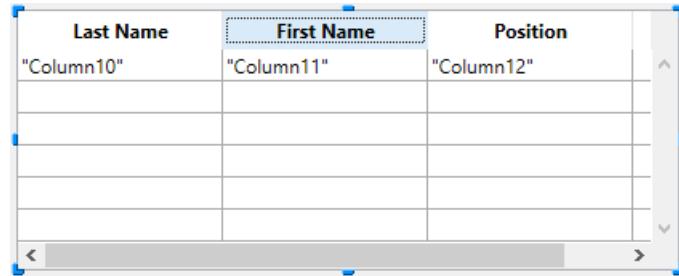
Evénement formulaire	<ul style="list-style-type: none"> ● columnName ● row <p>Evénement formulaire pour les propriétés principales)</p> <ul style="list-style-type: none"> ● column ● columnName ● row 	Commentaires
Sur avant frappe clavier		
Sur début survol	<ul style="list-style-type: none"> ● column ● columnName ● row 	
Sur clic	<ul style="list-style-type: none"> ● column ● columnName ● row 	
Sur déplacement colonne	<ul style="list-style-type: none"> ● columnName ● newPosition ● oldPosition 	
Sur redimensionnement colonne	<ul style="list-style-type: none"> ● column ● columnName ● newSize ● oldSize 	
Sur données modifiées	<ul style="list-style-type: none"> ● column ● columnName ● row 	
Sur double clic	<ul style="list-style-type: none"> ● column ● columnName ● row 	
Sur glisser	<ul style="list-style-type: none"> ● area ● areaName ● column ● columnName ● row 	
Sur déposer	<ul style="list-style-type: none"> ● column ● columnName ● row 	
Sur clic pied	<ul style="list-style-type: none"> ● column ● columnName ● footerName 	<i>List box Tableau, Sélection courante et Sélection temporaire uniquement</i>
On Getting Focus	<ul style="list-style-type: none"> ● column ● columnName ● row 	<i>Propriétés supplémentaires retournées uniquement lors de la modification d'une cellule</i>
Sur clic entête	<ul style="list-style-type: none"> ● column ● columnName ● headerName 	
On Load		

On Losing Focus Événement formulaire	<ul style="list-style-type: none"> • column Propriétés supplémentaires retournées (voir Événement formulaire pour les propriétés principales) 	<i>Propriétés supplémentaires retournées Commentaires uniquement lorsque la modification d'une cellule est achevée</i>
Sur déplacement ligne	<ul style="list-style-type: none"> • newPosition • oldPosition 	<i>Listbox tableau uniquement</i>
Sur défilement	<ul style="list-style-type: none"> • horizontalScroll • verticalScroll 	
On Unload		

En-têtes de list box

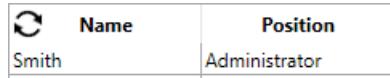
Pour pouvoir accéder aux propriétés des en-têtes d'une list box, vous devez avoir coché l'option [Afficher en-têtes](#) dans la Liste des propriétés de la list box.

Lorsque les en-têtes sont affichés, vous pouvez sélectionner un en-tête dans l'éditeur de formulaires en cliquant dessus lorsque l'objet List box est sélectionné :



Vous pouvez définir, pour chaque en-tête de colonne de List box, des propriétés standard de texte : dans ce cas, ces propriétés sont prioritaires par rapport à celles de la colonne ou de la list box.

Vous pouvez également accéder à des propriétés spécifiques aux en-têtes. Specifically, an icon can be displayed in the header next to or in place of the column title, for example when performing [customized sorts](#).



At runtime, events that occur in a header are generated in the [list box column object method](#).

When the `OBJECT SET VISIBLE` command is used with a header, it is applied to all headers, regardless of the individual element set by the command. For example, `OBJECT SET VISIBLE(*;"header3";False)` will hide all headers in the list box object to which *header3* belongs and not simply this header.

Propriétés spécifiques des en-têtes

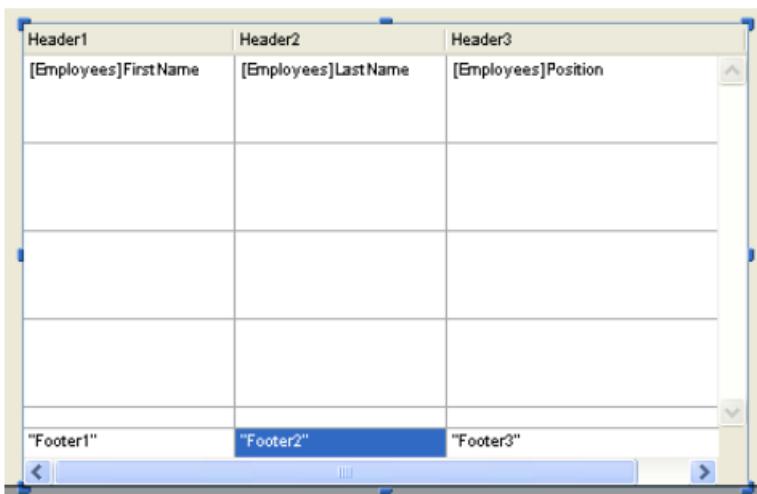
[Gras](#) - [Css Class](#) - [Police](#) - [Couleur de fond](#) - [Message d'aide](#) - [Alignement horizontal](#) - [Emplacement de l'icône](#) - [Italique](#) - [Nom](#) - [Chemin d'accès](#) - [Titre](#) - [Souligné](#) - [Variable ou expression](#) - [Alignement vertical](#) - [Largeur](#)

Pieds de List box

Pour pouvoir accéder aux propriétés des pieds d'une List box, vous devez avoir coché l'option [Afficher pieds](#) dans la Liste des propriétés de la List box.

Les List box peuvent contenir des zones de "pied de page" non saisissables, affichant des informations supplémentaires. Dans les données présentées sous forme de tableaux, les pieds sont généralement utilisés pour afficher des calculs, tels que des sommes ou des moyennes.

Lorsque les pieds sont affichés, vous pouvez sélectionner un pied de list box dans l'éditeur de formulaires en cliquant dessus lorsque l'objet List box est sélectionné :



Vous pouvez définir, pour chaque pied de colonne de List box, des propriétés standard de texte : dans ce cas, ces propriétés sont prioritaires par rapport à celles de la colonne ou de la list box. You can also access specific properties for footers. In particular, you can insert a [custom or automatic calculation](#).

At runtime, events that occur in a footer are generated in the [list box column object method](#).

When the `OBJECT SET VISIBLE` command is used with a footer, it is applied to all footers, regardless of the individual element set by the command. For example, `OBJECT SET VISIBLE(*;"footer3";False)` will hide all footers in the list box object to which `footer3` belongs and not simply this footer.

Footer Specific Properties

[Alpha Format](#) - [Background Color](#) - [Bold](#) - [Class](#) - [Date Format](#) - [Expression Type](#) - [Font](#) - [Font Color](#) - [Help Tip](#) - [Horizontal Alignment](#) - [Italic](#) - [Number Format](#) - [Object Name](#) - [Picture Format](#) - [Time Format](#) - [Truncate with ellipsis](#) - [Underline](#) - [Variable Calculation](#) - [Variable or Expression](#) - [Vertical Alignment](#) - [Width](#) - [Wordwrap](#)

Managing entry

For a list box cell to be enterable, both of the following conditions must be met:

- The cell's column must have been set as [Enterable](#) (otherwise, the cells of the column can never be enterable).
- In the [On Before Data Entry](#) event, \$0 does not return -1. When the cursor arrives in the cell, the [On Before Data Entry](#) event is generated in the column method. Si, dans le contexte de cet événement, \$0 est défini sur -1, la cellule est considérée comme non saisissable. Si l'événement a été généré après avoir appuyé sur Tab ou Maj+Tab, le focus va respectivement à la cellule suivante ou à la précédente. Si la valeur de \$0 n'est pas -1 (par défaut \$0 est 0), la cellule est saisissable et passe en mode d'édition.

Let's consider the example of a list box containing two arrays, one date and one text. The date array is not enterable but the text array is enterable if the date has not already past.

Header1	Header2
Variable Name: tDate	Variable Name: tText

Here is the method of the `arrText` column:

```

Case of
  : (FORM event.code=On Before Data Entry) // a cell gets the focus
    LISTBOX GET CELL POSITION(*;"lb";$col;$row)
  // identification of cell
    If(arrDate{$row}<Current date) // if date is earlier than today
      $0:=-1 // cell is NOT enterable
    Else
      // otherwise, cell is enterable
    End if
End case

```

The `On Before Data Entry` event is returned before `On Getting Focus`.

In order to preserve data consistency for selection type and entity selection type list boxes, any modified record/entity is automatically saved as soon as the cell is validated, i.e.:

- when the cell is deactivated (user presses tab, clicks, etc.)
- when the listbox is no longer focused,
- when the form is no longer focused.

The typical sequence of events generated during data entry or modification is as follows:

Action	Type(s) de Listbox	Séquence d'événements
A cell switches to edit mode (user action or a call to the <code>EDIT ITEM</code> command)	Tous	Sur avant saisie
	Tous	On Getting Focus
Its value is modified	Tous	Sur avant frappe clavier
	Tous	Sur après frappe clavier
	Tous	Sur après modification
A user validates and leaves the cell	List box de type sélection	Save
	Record selection list boxes	On saving an existing record trigger (if set)
	List box de type sélection	On Data Change(*)
	Entity selection list boxes	Entity is saved with automerge option, optimistic lock (see <code>entity.save()</code>). In case of successful save, the entity is refreshed with the last update done. If the save operation fails, an error is displayed
	Tous	On Losing Focus

(*) With entity selection list boxes, in the On Data Change event:

- the `Current item` object contains the value before modification.
- the `This` object contains the modified value.

Data entry in collection/entity selection type list boxes has a limitation when the expression evaluates to null. In this case, it is not possible to edit or remove the null value in the cell.

Managing selections

La gestion des sélections s'effectue différemment selon que la list box de type tableau, sélection d'enregistrements, ou collection/entity selection :

- List box de type sélection : les sélections sont gérées par l'intermédiaire d'un ensemble appelé par défaut `$ListboxSetN` (N débute à 0 et est incrémenté en fonction du nombre de list box dans le formulaire), que vous pouvez modifier si nécessaire. Cet ensemble est [défini dans les propriétés](#) de la list box. Il est maintenu automatiquement par 4D : si l'utilisateur sélectionne une ou plusieurs ligne(s) dans la list box, l'ensemble est immédiatement mis à jour. A l'inverse, il est possible d'utiliser les commandes du thème "Ensembles" afin de modifier par programmation la sélection dans la list box.
- List box de type collection/entity selection : les sélections sont gérées via des propriétés de list box dédiées :
 - [Elément courant](#) est un objet qui reçoit l'élément/l'entité sélectionné(e),
 - [Éléments sélectionnés](#) retourne la collection des éléments sélectionnés,
 - [Position élément courant](#) retourne la position de l'élément ou de l'entité sélectionné(e).
- List box de type tableau : la commande `LISTBOX SELECT ROW` permet de sélectionner par programmation une ou plusieurs lignes de list box. En outre, [la variable associée à l'objet List box](#) peut être utilisée pour lire, fixer ou stocker les sélections de lignes dans l'objet. Cette variable correspond à un tableau de booléens automatiquement créé et maintenu par 4D. La taille de ce tableau est déterminée par celle de la list box : il contient le même nombre d'éléments que le plus petit tableau associé aux colonnes. Chaque élément de ce tableau contient `Vrai` si la ligne correspondante est sélectionnée et `Faux` sinon. 4D met à jour le contenu de ce tableau en fonction des actions utilisateur. A l'inverse, vous pouvez modifier la valeur des éléments de ce tableau afin de modifier la sélection dans la list box. En revanche, vous ne pouvez ni insérer ni supprimer de ligne dans ce tableau ; il n'est pas possible non plus de le retyper. La commande `Count in array` est utile dans ce cas pour connaître le nombre de lignes sélectionnées. Par exemple, cette méthode permet d'inverser la sélection de la première ligne de la list box (type tableau) :

```
ARRAY BOOLEAN(tBLListBox;10)
// tBLListBox est le nom de la variable associée à la List box dans le formulaire
If(tBLListBox{1}=True)
    tBLListBox{1}:=False
Else
    tBLListBox{1}:=True
End if
```

The `OBJECT SET SCROLL POSITION` command scrolls the list box rows so that the first selected row or a specified row is displayed.

Customizing appearance of selected rows

When the [Hide selection highlight](#) option is selected, you need to make list box selections visible using available interface options. Since selections are still fully managed by 4D, this means:

- For array type list boxes, you must parse the Boolean array variable associated with the list box to determine which rows are selected or not.
- For selection type list boxes, you have to check whether the current record (row) belongs to the set specified in the [Highlight Set](#) property of the list box.

You can then define specific background colors, font colors and/or font styles by programming to customize the appearance of selected rows. This can be done using arrays or expressions, depending on the type of list box being displayed (see the following sections).

You can use the `lk_inherited` constant to mimic the current appearance of the list box (e.g., font color, background color, font style, etc.).

List box de type sélection

To determine which rows are selected, you have to check whether they are included in the set indicated in the [Highlight Set](#) property of the list box. You can then define the appearance of selected rows using one or more of the relevant [color or style expression property](#).

Keep in mind that expressions are automatically re-evaluated each time the:

- list box selection changes.
- list box gets or loses the focus.
- form window containing the list box becomes, or ceases to be, the frontmost window.

List box de type tableau

You have to parse the Boolean array [Variable or Expression](#) associated with the list box to determine whether rows are selected or not selected.

You can then define the appearance of selected rows using one or more of the relevant [color or style array property](#).

Note that list box arrays used for defining the appearance of selected rows must be recalculated during the [On Selection Change](#) form event; however, you can also modify these arrays based on the following additional form events:

- [On Getting Focus](#) (list box property)
- [On Losing Focus](#) (list box property)
- [On Activate](#) (form property)
- [On Deactivate](#) (form property) ...depending on whether and how you want to visually represent changes of focus in selections.

Exemple

You have chosen to hide the system highlight and want to display list box selections with a green background color, as shown here:

Category	ID	Reference	Value
Alpha	215	5A0DF64-EC5-955F7EA-BD284E4-8A	15,425
Bravo	196	D9E3484-547-AEECB8B-B1808FF-A6	4,592
Alpha	205	3295824-3A8-B48B870-2074C57-B9	-3,672
Charlie	197	B3800C4-C64-A6C95CB-ED27729-B5	16,212
Echo	214	835F344-8EE-B422E66-5C52074-01	-12,332
Alpha	200	46784B4-B20-AE2E51D-0159EA4-D0	1,283
Delta	213	11F5FD4-E48-9BD6E93-E8B9F82-59	13,236
Delta	203	3E80494-879-9F2CEC2-4008AA4-F5	-12,231
Charlie	202	015D694-9C8-91113AA-B8A51A1-52	27,100
Bravo	211	E998AC4-FAE-93BE025-E4CA634-E8	2,630
Charlie	207	B19F244-A30-A03B668-C407B43-D4	16,677
Delta	208	41B1DE4-D29-BC8E7BF-5062D92-B7	-14,759
Echo	199	7005654-722-926CDCE-D8E1BBB-83	23,952
Delta	198	0AD0734-0C7-BA8168E-A0AB67A-1A	-19,758
Alpha	210	6F46794-0D6-AF0E61A-D43231E-3E	24,342
Bravo	201	00A8334-B4B-8419285-8772CFB-B4	-3,657
Charlie	212	9EF2FD4-B1B-97138DE-B3750BA-FB	-4,850
Echo	209	FD05424-365-B8DB0C2-91098A8-80	2,941
Echo	204	7473724-2FA-82F49A5-010BDED-98	22,200
Bravo	206	3537D34-A9A-AE41C4E-34EC5B6-43	1,205

For an array type list box, you need to update the [Row Background Color Array](#) by programming. In the JSON form, you have defined the following Row Background Color Array for the list box:

```
"rowFillSource": "_ListboxBackground",
```

In the object method of the list box, you can write:

```

Case of
  : (FORM event.code=On Selection Change)
    $n:=Size of array(LB_Arrays)
    ARRAY LONGINT(_ListboxBackground;$n) // row background colors
    For($i;1;$n)
      If(LB_Arrays{$i}=True) // selected
        _ListboxBackground{$i}:=0x0080C080 // green background
      Else // not selected
        _ListboxBackground{$i}:=lk inherited
      End if
    End for
  End case

```

For a selection type list box, to produce the same effect you can use a method to update the [Background Color Expression](#) based on the set specified in the [Highlight Set](#) property.

For example, in the JSON form, you have defined the following Highlight Set and Background Color Expression for the list box:

```

"highlightSet": "$SampleSet",
"rowFillSource": "UI_SetColor",

```

You can write in the *UI_SetColor* method:

```

If(Is in set("$SampleSet"))
  $color:=0x0080C080 // green background
Else
  $color:=lk inherited
End if

$0:=$color

```

In hierarchical list boxes, break rows cannot be highlighted when the [Hide selection highlight](#) option is checked. Since it is not possible to have distinct colors for headers of the same level, there is no way to highlight a specific break row by programming.

Managing sorts

A sort in a list box can be standard or custom. When a column of a list box is sorted, all other columns are always synchronized automatically.

Standard sort

By default, a list box provides standard column sorts when the header is clicked. A standard sort is an alphanumeric sort of evaluated column values, alternately ascending/descending with each successive click.

You can enable or disable standard user sorts by disabling the [Sortable](#) property of the list box (enabled by default).

Standard sort support depends on the list box type:

List box type	Support of standard sort	Commentaires
Collection d'objets	Oui	<ul style="list-style-type: none"> "This.a" or "This.a.b" columns are sortable. The list box source property must be an assignable expression.
Collection of scalar values	Non	Use custom sort with <code>orderBy()</code> function
Entity selection	Oui	<ul style="list-style-type: none"> The list box source property must be an assignable expression. Supported: sorts on object attribute properties (e.g. "This.data.city" when "data" is an object attribute) Supported: sorts on related attributes (e.g. "This.company.name") Not supported: sorts on object attribute properties through related attributes (e.g. "This.company.data.city"). For this, you need to use custom sort with <code>orderByFormula()</code> function (see example below)
Current selection	Oui	Only simple expressions are sortable (e.g. <code>[Table_1]Field_2</code>)
Named selection	Non	
Tableaux	Oui	Columns bound to picture and pointer arrays are not sortable

Custom sort

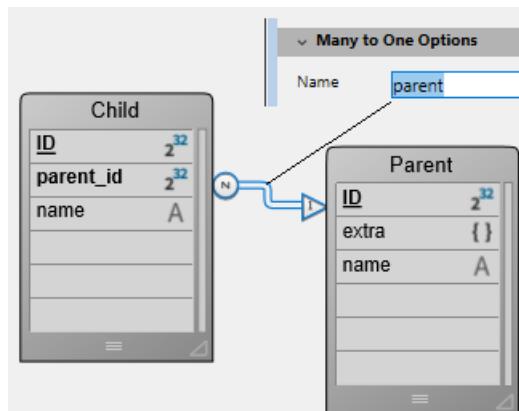
The developer can set up custom sorts, for example using the [LISTBOX SORT COLUMNS](#) command and/or combining the [On Header Click](#) and [On After Sort](#) form events and relevant 4D commands.

Custom sorts allow you to:

- carry out multi-level sorts on several columns, thanks to the [LISTBOX SORT COLUMNS](#) command,
- use functions such as `collection.orderByFormula()` or `entitySelection.orderByFormula()` to sort columns on complex criteria.

Exemple

You want to sort a list box using values of a property stored in a related object attribute. You have the following structure:



You design a list box of the entity selection type, bound to the `Form.child` expression. In the [On Load](#) form event, you execute `Form.child:=ds.Child.all()`.

You display two columns:

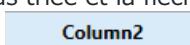
Child name	Parent's nickname
This.name	This.parent.extra.nickname

If you want to sort the list box using the values of the second column, you have to write:

```
If (Form event code=On Header Click)
    Form.child:=Form.child.orderByFormula("This.parent.extra.nickname"; dk ascending)
End if
```

Column header variable

The value of the [column header variable](#) allows you to manage additional information: the current sort of the column (read) and the display of the sort arrow.

- Si la variable est définie sur 0, la colonne n'est pas triée et la flèche de tri n'est pas affichée.

- Si la variable est définie sur 1, la colonne est triée par ordre croissant et la flèche de tri s'affiche.

- Si la variable est définie sur 2, la colonne est triée par ordre décroissant et la flèche de tri s'affiche.


Seules les [variables](#) déclarées ou dynamiques peuvent être utilisées comme variables d'en-tête de colonne. Les autres types d'[expressions](#) telles que `Form.sortValue` ne sont pas pris en charge.

Vous pouvez définir la valeur de la variable (par exemple, `Header2:=2`) afin de "forcer" l'affichage de la flèche de tri. The column sort itself is not modified in this case; it is up to the developer to handle it.

La commande [OBJECT SET FORMAT](#) permet une prise en charge spécifique des icônes dans les en-têtes de listbox, ce qui peut être utile lorsque vous souhaitez travailler avec une icône de tri personnalisée.

Managing row colors, styles, and display

There are several different ways to set background colors, font colors and font styles for list boxes:

- at the level of the [list box object properties](#),
- at the level of the [column properties](#),
- using [arrays or expressions properties](#) for the list box and/or for each column,
- at the level of the text of each cell (if [multi-style text](#)).

Priority & inheritance

Priority and inheritance principles are observed when the same property is set at more than one level.

Niveau de priorité	Setting location
high priority	Cell (if multi-style text)
	Column arrays/methods
	List box arrays/methods
	Column properties
	List box properties
low priority	Meta Info expression (for collection or entity selection list boxes)

For example, if you set a font style in the list box properties and another using a style array for the column, the latter one will be taken into account.

For each attribute (style, color and background color), an inheritance is implemented when the default value is used:

- for cell attributes: attribute values of rows
- for row attributes: attribute values of columns
- for column attributes: attribute values of the list box

This way, if you want an object to inherit the attribute value from a higher level, you can use pass the `lk inherited` constant (default value) to the definition command or directly in the element of the corresponding style/color array. For example, given an array list box containing a standard font style with alternating colors:

standard	standard	standard	standard	standard	standard
standard	standard	standard	standard	standard	standard
standard	standard	standard	standard	standard	standard
standard	standard	standard	standard	standard	standard
standard	standard	standard	standard	standard	standard
standard	standard	standard	standard	standard	standard

You perform the following modifications:

- change the background of row 2 to red using the [Row Background Color Array](#) property of the list box object,
- change the style of row 4 to italics using the [Row Style Array](#) property of the list box object,
- two elements in column 5 are changed to bold using the [Row Style Array](#) property of the column 5 object,
- the 2 elements for column 1 and 2 are changed to dark blue using the [Row Background Color Array](#) property for the column 1 and 2 objects:

standard	standard	standard	standard	standard	standard
standard	standard	standard	standard	standard	standard
standard	standard	standard	standard	standard	standard
standard	standard	standard	standard	standard	standard
standard	standard	standard	standard	standard	standard
standard	standard	standard	standard	standard	standard

To restore the original appearance of the list box, you can:

- pass the `lk inherited` constant in element 2 of the background color arrays for columns 1 and 2: then they inherit the red background color of the row.
- pass the `lk inherited` constant in elements 3 and 4 of the style array for column 5: then they inherit the standard style, except for element 4, which changes to italics as specified in the style array of the list box.
- pass the `lk inherited` constant in element 4 of the style array for the list box in order to remove the italics style.
- pass the `lk inherited` constant in element 2 of the background color array for the list box in order to restore the original alternating color of the list box.

Using arrays and expressions

Depending of the list box type, you can use different properties to customize row colors, styles and display:

Propriété	List box tableau	Liste box sélection	List box collection ou entity selection
Couleur de fond	Tableau couleurs de fond	Expression couleur de fond	Background Color Expression or Meta info expression
Font color	Tableau couleurs de police	Expression couleur police	Font Color Expression or Meta info expression

[Row Style Array](#) | [Style Expression](#) | [Style Expression or Meta info expression](#) | [Display](#) | [Row Control Array](#) | -|-|

Printing list boxes

Two printing modes are available: preview mode - which can be used to print a list box like a form object, and advanced mode - which lets you control the printing of the list box object itself within the form. Note that the "Printing" appearance is available for list box objects in the Form editor.

Preview mode

Printing a list box in preview mode consists of directly printing the list box and the form that contains it using the standard print commands or the Print menu command. The list box is printed as it is in the form. This mode does not allow precise control of the printing of the object; in particular, it does not allow you to print all the rows of a list box that contains more rows than it can display.

Mode avancé

In this mode, the printing of list boxes is carried out by programming, via the `Print object` command (project forms and table forms are supported). The `LISTBOX GET PRINT INFORMATION` command is used to control the printing of the object.

In this mode:

- The height of the list box object is automatically reduced when the number of rows to be printed is less than the original height of the object (there are no "blank" rows printed). On the other hand, the height does not automatically increase according to the contents of the object. The size of the object actually printed can be obtained via the `LISTBOX GET PRINT INFORMATION` command.
- The list box object is printed "as is", in other words, taking its current display parameters into account: visibility of headers and gridlines, hidden and displayed rows, etc. These parameters also include the first row to be printed: if you call the `OBJECT SET SCROLL POSITION` command before launching the printing, the first row printed in the list box will be the one designated by the command.
- An automatic mechanism facilitates the printing of list boxes that contain more rows than it is possible to display: successive calls to `Print object` can be used to print a new set of rows each time. The `LISTBOX GET PRINT INFORMATION` command can be used to check the status of the printing while it is underway.

List box hiérarchiques

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

Seules les [list box de type tableau](#) peuvent être hiérarchiques.

Les list box hiérarchiques constituent un mode de représentation particulier des données, mais ne modifient pas la structure de ces données (les tableaux). Les list box hiérarchiques sont gérées exactement de la même manière que les list box non hiérarchiques.

Définir une hiérarchie

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

- Configurer manuellement les éléments hiérarchiques via la liste des propriétés dans l'éditeur de formulaire (ou déditez le formulaire JSON).
- Générer visuellement la hiérarchie à l'aide du pop up menu de gestion des list box, dans l'éditeur de formulaires.
- Use the `LISTBOX SET HIERARCHY` and `LISTBOX GET HIERARCHY` commands, described in the *4D Language Reference manual*.

Propriétés de la List Box hiérarchique

This property specifies that the list box must be displayed in hierarchical form. In the JSON form, this feature is triggered [when the `dataSource` property value is an array](#), i.e. a collection.

Additional options (Variable 1...10) are available when the *Hierarchical List Box* option is selected, corresponding to each `dataSource` array to use as break column. Each time a value is entered in a field, a new row is added. Up to 10 variables can be specified. These variables set the hierarchical levels to be displayed in the first column.

The first variable always corresponds to the name of the variable for the first column of the list box (the two values are automatically bound). This first variable is always visible and enterable. For example: country. The second variable is

also always visible and enterable; it specifies the second hierarchical level. For example: regions. Beginning with the third field, each variable depends on the one preceding it. For example: counties, cities, and so on. A maximum of ten hierarchical levels can be specified. If you remove a value, the whole hierarchy moves up a level.

The last variable is never hierarchical even if several identical values exists at this level. For example, referring to the configuration illustrated above, imagine that arr1 contains the values A A A B B B, arr2 has the values 1 1 1 2 2 2 and arr3 the values X X Y Y Y Z. In this case, A, B, 1 and 2 could appear in collapsed form, but not X and Y:

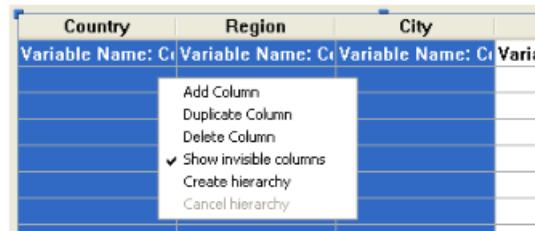
```
+ A  
+ 1  
X  
X  
Y  
+ B  
+ 2  
Y  
Y  
Z
```

This principle is not applied when only one variable is specified in the hierarchy: in this case, identical values may be grouped.

If you specify a hierarchy based on the first columns of an existing list box, you must then remove or hide these columns (except for the first), otherwise they will appear in duplicate in the list box. If you specify the hierarchy via the pop-up menu of the editor (see below), the unnecessary columns are automatically removed from the list box.

Créer une hiérarchie via le menu contextuel

When you select at least one column in addition to the first one in a list box object (of the array type) in the form editor, the Create hierarchy command is available in the context menu:



This command is a shortcut to define a hierarchy. Lorsque vous la choisissez, les actions suivantes sont effectuées :

- The Hierarchical list box option is checked for the object in the Property List.
- Les variables des colonnes sont utilisées pour définir la hiérarchie. Elles remplacent les variables éventuellement déjà définies.
- Les colonnes sélectionnées n'apparaissent plus dans la list box (à l'exception du titre de la première).

Exemple : soit une list box dont les premières colonnes contiennent Pays, Région, Ville et Population. When Country, Region and City are selected, if you choose Create hierarchy in the context menu, a three-level hierarchy is created in the first column, columns 2 and 3 are removed and the Population column becomes the second:

The screenshot shows a software interface with two main components: a configuration dialog on the right and a preview window on the left.

PropertyList Dialog (Right):

- Type:** List Box
- Object Name:** List Box
- Variable or Expression:** List Box
- Data Source:** Arrays

List Box Properties:

- Number of Columns: 2
- Number of Locked Columns: 0
- Number of Static Columns: 1
- Row Control Array:
- Selection Mode: Multiple

Headers:

- Display Headers:
- Height: 1 lines

Footers:

- Display Footers:
- Height: 1 lines

Hierarchy:

- Hierarchical List Box:
- Variable 1: Country
- Variable 2: Region
- Variable 3: City Selected
- Variable 4:

Other Sections:

- Coordinates & Sizing
- Resizing Options
- Entry
- Display
- Appearance
- Background and Border
- Text

All Themes

Preview Window (Left):

A grid view showing two columns: "Country" and "Population". The "Country" column has a header "Variable Name: Column1" and the "Population" column has a header "Variable Name: Column4". The grid contains several rows of data, with the first row being highlighted in blue.

Annuler une hiérarchie

When the first column is selected and already specified as hierarchical, you can use the Cancel hierarchy command. Lorsque vous choisissez cette commande, les actions suivantes sont effectuées :

- The Hierarchical list box option is deselected for the object,
 - Les niveaux hiérarchiques 2 à n sont supprimés et transformés en colonnes ajoutées dans la list box.

Principes de fonctionnement

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

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

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

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

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

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

```
> A B A C
```

Pour déployer ou contracter un "nœud" hiérarchique, cliquez dessus. If you Alt+click (Windows) or Option+click (macOS) on the node, all its sub-elements will be expanded or collapsed as well. Ces opérations peuvent également être effectuées par programmation à l'aide des commandes `LISTBOX EXPAND` et `LISTBOX COLLAPSE`.

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

Gestion des tris dans les list box hiérarchiques

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

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

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

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

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

1- Tri automatique de tous les niveaux hiérarchiques par ordre croissant

En cas de clics ultérieurs sur l'en-tête Population, seul le dernier niveau sera synchronisé

Clic sur l'en-tête : tri de la population (ordre croissant)

2 - Tri de la population par ordre croissant à l'intérieur du dernier niveau

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

Comme pour toutes les list box, vous pouvez [désactiver le mécanisme de tri standard](#) en désélectionnant la propriété "Trible" pour la list box et gérer le tri par programmation.

Gestion des sélections et des positions dans les list box hiérarchiques

Une list box hiérarchique affiche un nombre variable de lignes à l'écran en fonction de l'état déployé/contracté des nœuds hiérarchiques. Cela ne signifie pas pour autant que le nombre de lignes des tableaux varie. Seul l'affichage est modifié, pas les données. Il est important de comprendre ce principe car la gestion programmée des list box hiérarchiques se base toujours sur les données des tableaux, pas sur les données affichées. En particulier, les lignes de rupture ajoutées automatiquement ne sont pas prises en compte dans les tableaux d'options d'affichage (cf. ci-dessous).

le paragraphe Gestion des lignes de rupture).

Examinons par exemple les tableaux suivants :

France	Bretagne	Brest
France	Bretagne	Quimper
France	Bretagne	Rennes

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

France
Bretagne
Brest
Quimper
Rennes

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

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

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

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

```
->MyListbox{3}:=True
```

Représentation non hiérarchique :

France	Bretagne	Brest
France	Bretagne	Quimper
France	Bretagne	Rennes

Représentation hiérarchique :

France
Bretagne
Brest
Quimper
Rennes

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

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

Non-hierarchical representation:

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

Hierarchical representation:

France	
Brittany	
Brest	120000
Quimper	80000
Rennes	200000
Normandy	
Caen	75000

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

Lignes de rupture

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

France	
Bretagne	
Brest	120000
Quimper	80000
Rennes	200000
Normandie	
Caen	75000

... `LISTBOX GET CELL POSITION` returns (2;4). Pour sélectionner une ligne de rupture par programmation, vous devez utiliser la commande `LISTBOX SELECT BREAK` .

Les lignes de rupture ne sont pas prises en compte dans les tableaux internes permettant de gérer l'apparence graphique des list box (styles et couleurs). Il est toutefois possible de modifier ces caractéristiques pour les lignes de rupture via les commandes de gestion graphique des objets. Il suffit pour cela d'exécuter ces commandes appropriées sur les tableaux constituant la hiérarchie.

Soit par exemple la list box suivante (les noms des tableaux associés sont précisés entre parenthèses) :

Non-hierarchical representation:

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

Hierarchical representation:

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

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

```
OBJECT SET RGB COLORS(T1;0x0000FF;0xB0B0B0)
OBJECT SET FONT STYLE(T2;Bold)
```

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

Résultat :

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

Gestion optimisée du déployer/contracter

Vous pouvez optimiser l'affichage et la gestion des list box hiérarchiques en tirant parti des événements formulaire `On Expand` et `On Collapse`.

Une list box hiérarchique est construite à partir du contenu des tableaux qui la constituent, elle ne peut donc être affichée que lorsque tous les tableaux sont chargés en mémoire. Ce principe peut rendre difficile la génération de list box hiérarchiques de grande taille basées sur des tableaux générés à partir des données (via la commande `SELECTION TO ARRAY`), pour des raisons de rapidité d'affichage et d'utilisation de la mémoire.

L'emploi des événements formulaire `On Expand` et `On Collapse` permet de s'affranchir de ces contraintes : il est possible de n'afficher qu'une partie de la hiérarchie et d'effectuer le chargement et le déchargement des tableaux à la volée, en fonction des actions de l'utilisateur. Dans le contexte de ces événements, la commande `LISTBOX GET CELL POSITION` retourne la cellule sur laquelle l'utilisateur a cliqué afin de déployer ou de contracter une ligne.

Dans ce cas, le remplissage et le vidage des tableaux doivent être effectués par le code. Les principes à mettre en oeuvre sont :

- A l'affichage de la listbox, seul le premier tableau doit être rempli. Vous devez toutefois créer un second tableau avec des valeurs vides afin que la list box affiche les boutons déployer/contracter :

Artistes/Albums/Pistes	CDs	Pistes	Durées
Brasil			
Celtic			
Classical			
Jazz			
New Age			
Others			
Pop/Rock			
Soundtrack			
World			

- Lorsque l'utilisateur clique sur un bouton de déploiement, vous pouvez traiter l'événement `On Expand`. La commande `LISTBOX GET CELL POSITION` retourne la cellule concernée et vous permet de construire la hiérarchie adéquate : vous alimentez le premier tableau avec des valeurs répétées et le second avec les valeurs issues de la commande `SELECTION TO ARRAY`, et vous insérez dans la list box autant de lignes que nécessaire à l'aide de la commande `LISTBOX INSERT ROWS`.

Artistes/Albums/Pistes	CDs	Pistes	Durées
Brasil			
Celtic			
Classical			
Jazz			
New Age			
Others			
Jacqueline Maillan			
Pierre Dac			
Pierre Dac & Francis Blanche			
Pop/Rock			
Soundtrack			
World			

- Lorsque l'utilisateur clique sur un bouton de contraction, vous pouvez traiter l'événement `On Collapse`. La commande `LISTBOX GET CELL POSITION` retourne la cellule concernée : vous supprimez de la list box autant de lignes que nécessaire à l'aide de la commande `LISTBOX DELETE ROWS`.

Object arrays in columns

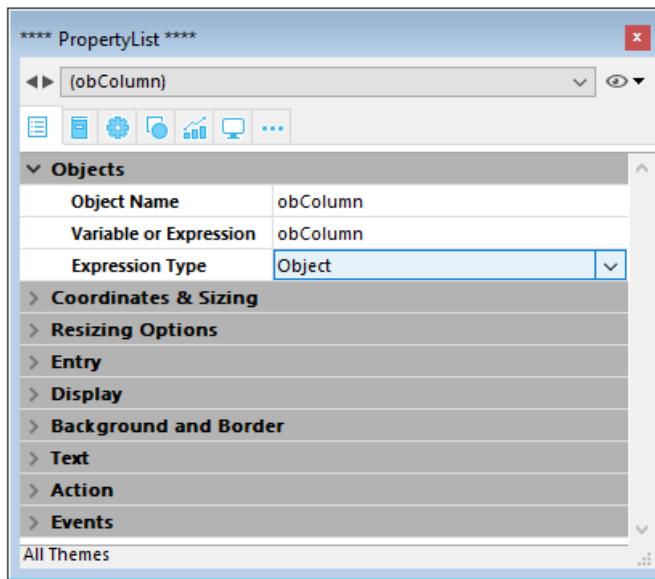
Les colonnes de list box peuvent être associées à des tableaux d'objets. Comme les tableaux d'objets peuvent contenir des données de types différents, cette puissante fonctionnalité vous permet de saisir et d'afficher divers types de valeurs dans les lignes d'une même colonne, ainsi que d'utiliser divers objets d'interface (widgets). Par exemple, vous pouvez placer une zone de saisie de texte dans la première ligne, une case à cocher dans la seconde, et une liste déroulante dans la troisième. Les tableaux d'objets vous donnent également accès à des widgets supplémentaires, tels que des boutons ou des sélecteurs de couleurs (color picker).

La list box suivante a été définie à l'aide d'un tableau d'objets :

Label	Value
Document Name	MyReport
Document Type	PDF ▾
Reference	123456
Category	[...]
Include Abstract	<input checked="" type="checkbox"/>
Printable area size (height)	297 mm
Printable area size (width)	210 mm
Show Preview	Preview...

Configuring an object array column

To assign an object array to a list box column, you just need to set the object array name in either the Property list ("Variable Name" field), or using the [LISTBOX INSERT COLUMN](#) command, like with any array-based column. In the Property list, you can now select Object as a "Expression Type" for the column:



Standard properties related to coordinates, size, and style are available for object columns. You can define them using the Property list, or by programming the style, font color, background color and visibility for each row of an object-type list box column. These types of columns can also be hidden.

However, the Data Source theme is not available for object-type list box columns. In fact, the contents of each column cell are based on attributes found in the corresponding element of the object array. Each array element can define:

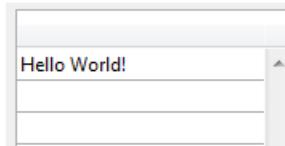
the value type (mandatory): text, color, event, etc. the value itself (optional): used for input/output. the cell content display (optional): button, list, etc. additional settings (optional): depend on the value type To define these properties, you need to set the appropriate attributes in the object (available attributes are listed below). For example, you can write "Hello World!" in an object column using this simple code:

```

ARRAY OBJECT(obColumn;0) //column array
C_OBJECT($ob1)
$entry:="Hello world!" OB SET($ob1;"valueType";"text")
OB SET($ob1;"value";$entry) // if the user enters a new value, $entry will contain the edited value
C_OBJECT($ob2)
OB SET($ob2;"valueType";"real")
OB SET($ob2;"value";2/3)
C_OBJECT($ob3)
OB SET($ob3;"valueType";"boolean")
OB SET($ob3;"value";True)

APPEND TO ARRAY(obColumn;$ob1)
APPEND TO ARRAY(obColumn;$ob2)
APPEND TO ARRAY(obColumn;$ob3)

```



Display format and entry filters cannot be set for an object column. They automatically depend on the value type.

valueType et affichage des données

When a list box column is associated with an object array, the way a cell is displayed, entered, or edited, is based on the valueType attribute of the array element. Supported valueType values are:

- "text": for a text value
- "real": for a numeric value that can include separators like a <space>, <.>, or <,>
- "integer": for an integer value
- "boolean": for a True/False value
- "color": to define a background color
- "event": to display a button with a label.

4D uses default widgets with regards to the "valueType" value (i.e., a "text" is displayed as a text input widget, a "boolean" as a check box), but alternate displays are also available through options (e.g., a real can also be represented as a drop-down menu). The following table shows the default display as well as alternatives for each type of value:

valueType	Default widget	Alternative widget(s)
Texte	text input	drop-down menu (required list) or combo box (choice list)
réel	controlled text input (numbers and separators)	drop-down menu (required list) or combo box (choice list)
entier	controlled text input (numbers only)	drop-down menu (required list) or combo box (choice list) or three-states check box
boolean	check box	drop-down menu (required list)
color	background color	Texte
event	button with label	
		All widgets can have an additional unit toggle button or ellipsis button attached to the cell.

You set the cell display and options using specific attributes in each object (see below).

Formats d'affichage et filtres de saisie

You cannot set display formats or entry filters for columns of object-type list boxes. They are automatically defined according to the value type. These are listed in the following table:

Value type	Default format	Entry control
Texte	same as defined in object	any (no control)
réel	same as defined in object (using system decimal separator)	"0-9" and "." and "-" "0-9" and "." if min>=0
entier	same as defined in object	"0-9" and "-" "0-9" if min>=0
Booléen	check box	N/A
color	N/A	N/A
event	N/A	N/A

Attributs

Chaque élément du tableau d'objets est un objet qui peut contenir un ou plusieurs attributs qui définiront le contenu de la cellule et l'affichage des données (voir exemple ci-dessus).

The only mandatory attribute is "valueType" and its supported values are "text", "real", "integer", "boolean", "color", and "event". The following table lists all the attributes supported in list box object arrays, depending on the "valueType" value (any other attributes are ignored). Display formats are detailed and examples are provided below.

	valueType	Texte	réel	entier	boolean	color	event
<i>Attributs</i>	<i>Description</i>						
value	cell value (input or output)	x	x	x			
min	minimum value		x	x			
max	maximum value		x	x			
behavior	"threeStates" value			x			
requiredList	drop-down list defined in object	x	x	x			
choiceList	combo box defined in object	x	x	x			
requiredListReference	4D list ref, depends on "saveAs" value	x	x	x			
requiredListName	4D list name, depends on "saveAs" value	x	x	x			
saveAs	"reference" or "value"	x	x	x			
choiceListReference	4D list ref, display combo box	x	x	x			
choiceListName	4D list name, display combo box	x	x	x			
unitList	array of X elements	x	x	x			
unitReference	index of selected element	x	x	x			
unitsListReference	4D list ref for units	x	x	x			
unitsListName	4D list name for units	x	x	x			
alternateButton	add an alternate button	x	x	x	x	x	

value

Cell values are stored in the "value" attribute. This attribute is used for input as well as output. It can also be used to define default values when using lists (see below).

```
C_OBJECT($ob1)
$entry:= "Hello world!"
OB SET($ob;"valueType";"text")
OB SET($ob;"alternateButton";True)
OB SET($ob;"value";$entry)
```

Null values are supported and result in an empty cell.

min et max

Lorsque le "valueType" est "real" ou "integer", l'objet accepte également les attributs min et max avec les valeurs appropriées (les valeurs doivent être du même type que valueType).

Ces attributs peuvent être utilisés pour contrôler la plage de valeurs d'entrée. Lorsqu'une cellule est validée (lorsqu'elle perd le focus), si la valeur de saisie est inférieure à la valeur minimale ou supérieure à la valeur maximale, elle est rejetée. Dans ce cas, la valeur précédente est conservée et une astuce affiche une explication.

```
C_OBJECT($ob3)
$entry3:=2015
OB SET($ob3;"valueType";"integer")
OB SET($ob3;"value";$entry3)
OB SET($ob3;"min";2000)
OB SET($ob3;"max";3000)
```

behavior

L'attribut behavior propose des variations de la représentation standard des valeurs. Une seule variation est possible :

Attribut	Valeur(s) disponible(s)	valueType(s)	Description
behavior	threeStates	entier	Représente une valeur numérique sous forme de case à cocher à trois états. 2=intermédiaire, 1=cochée, 0=non cochée, -1=invisible, -2=non cochée désactivée, -3=cochée désactivée, -4=intermédiaire désactivée

```
C_OBJECT($ob3)
OB SET($ob3;"valueType";"integer")

OB SET($ob3;"value";-3)
C_OBJECT($ob4)
OB SET($ob4;"valueType";"integer")
OB SET($ob4;"value";-3)
OB SET($ob4;"behavior";"threeStates")
```

requiredList et choiceList

Lorsqu'un attribut "choiceList" ou "requiredList" est présent dans l'objet, la zone de saisie de texte est remplacée par une liste déroulante ou une combo box, en fonction de l'attribut :

- Si l'attribut est "choiceList", la cellule est affichée sous forme de combo box. Cela signifie que l'utilisateur peut sélectionner ou saisir une valeur.
- Si l'attribut est "requiredList", la cellule est affichée sous forme de liste déroulante. Cela signifie que l'utilisateur peut uniquement sélectionner une des valeurs de la liste.

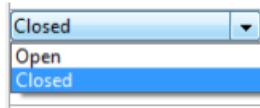
Dans les deux cas, vous pouvez utiliser un attribut "value" pour présélectionner une valeur dans le widget.

Les valeurs du widget sont définies via un tableau. Si vous souhaitez associer le widget à une énumération 4D existante, vous devez utiliser les attributs "requiredListReference", "requiredListName", "choiceListReference" ou "choiceListName".

Voici quelques exemples :

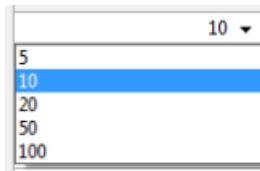
- Vous voulez afficher une liste déroulante avec juste deux options, "Open" ou "Closed". "Closed" doit être présélectionné :

```
ARRAY TEXT($RequiredList;0)
APPEND TO ARRAY($RequiredList;"Open")
APPEND TO ARRAY($RequiredList;"Closed")
C_OBJECT($ob)
OB SET($ob;"valueType";"text")
OB SET($ob;"value";"Closed")
OB SET ARRAY($ob;"requiredList";$RequiredList)
```



- Vous voulez accepter toute valeur entière, mais afficher une combo box contenant les valeurs les plus communes :

```
ARRAY LONGINT($ChoiceList;0)
APPEND TO ARRAY($ChoiceList;5)
APPEND TO ARRAY($ChoiceList;10)
APPEND TO ARRAY($ChoiceList;20)
APPEND TO ARRAY($ChoiceList;50)
APPEND TO ARRAY($ChoiceList;100)
C_OBJECT($ob)
OB SET($ob;"valueType";"integer")
OB SET($ob;"value";10) //10 comme valeur par défaut
OB SET ARRAY($ob;"choiceList";$ChoiceList)
```



requiredListName et requiredListReference

Les attributs "requiredListName" et "requiredListReference" vous permettent d'utiliser, dans une cellule de list box, une énumération définie dans 4D soit en mode Développement (via l'éditeur d'Enumérations de la Boîte à outils) soit par programmation (à l'aide de la commande New list). La cellule sera alors affichée sous forme de liste déroulante. Cela signifie que l'utilisateur pourra uniquement choisir une des valeurs fournies dans la liste.

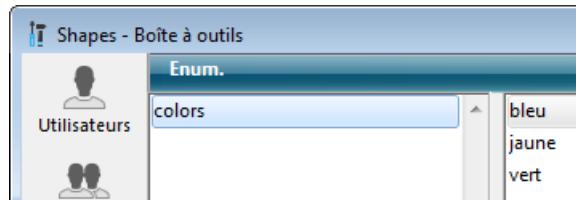
Utilisez "requiredListName" ou "requiredListReference" en fonction de la provenance de la liste : si la liste provient de la

Boîte à outils, utilisez son nom ; sinon, si la liste a été définie par programmation, passez sa référence. Dans les deux cas, vous pouvez utiliser un attribut "value" pour présélectionner une valeur dans le widget.

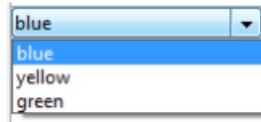
- Si vous souhaitez définir des valeurs d'énumération via un simple tableau, vous pouvez utiliser l'attribut "requiredList".
- Si la liste contient du texte représentant des valeurs réelles, le séparateur décimal doit être le point ("."), quels que soient les paramètres locaux, ex : "17.6" "1234.456".

Voici quelques exemples :

- Vous voulez afficher une liste déroulante basée sur une énumération nommée "colors" définie dans la Boîte à outils (contenant les valeurs "bleu", "jaune" et "vert"), la stocker en tant que valeur et afficher "bleu" par défaut :



```
C_OBJECT($ob)
OB SET($ob;"valueType";"text")
OB SET($ob;"saveAs";"value")
OB SET($ob;"value";"bleu")
OB SET($ob;"requiredListName";"colors")
```



- Vous voulez afficher une liste déroulante basée sur une liste créée par programmation, et la stocker en tant que référence :

```
<>List:=New list
APPEND TO LIST(<>List;"Paris";1)
APPEND TO LIST(<>List;"London";2)
APPEND TO LIST(<>List;"Berlin";3)
APPEND TO LIST(<>List;"Madrid";4)
C_OBJECT($ob)
OB SET($ob;"valueType";"integer")
OB SET($ob;"saveAs";"reference")
OB SET($ob;"value";2) //affiche London par défaut
OB SET($ob;"requiredListReference";<>List)
```

```
! [] (/docs/Rx/assets/en/FormObjects/listbox_column_objectArray_cities.png)
```

choiceListName et choiceListReference

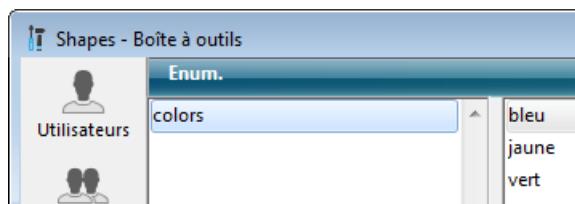
Les attributs "choiceListName" et "choiceListReference" permettent d'utiliser, dans une cellule de list box, une énumération définie dans 4D soit en mode Développement (via l'éditeur de la Boîte à outils) soit par programmation (à l'aide de la commande New list). La cellule sera alors affichée sous forme de combo box, ce qui signifie que l'utilisateur pourra choisir une des valeurs de la liste ou en saisir une.

Utilisez "choiceListName" ou "choiceListReference" en fonction de la provenance de la liste : si la liste provient de la Boîte à outils, utilisez son nom ; sinon, si la liste a été définie par programmation, passez sa référence. Dans les deux cas, vous pouvez utiliser un attribut "value" pour présélectionner une valeur dans le widget.

- Si vous souhaitez définir des valeurs d'énumération via un simple tableau, vous pouvez utiliser l'attribut "choiceList".
- Si la liste contient du texte représentant des valeurs réelles, le séparateur décimal doit être le point ("."), quels que soient les paramètres locaux, ex : "17.6" "1234.456".

Exemple :

Vous voulez afficher une combo box basée sur une énumération nommée "colors" définie dans la Boîte à outils (contenant les valeurs "bleu", "jaune" et "vert") et afficher "vert" par défaut :



```
C_OBJECT($ob)
OB SET($ob;"valueType";"text")
OB SET($ob;"value";"vert")
OB SET($ob;"choiceListName";"colors")
```

unitsList, unitsListName, unitsListReference et unitReference

Vous pouvez utiliser des attributs spécifiques afin d'associer des unités aux valeurs des cellules (par exemple "10 cm", "20 pixels", etc.). Pour définir une liste d'unités, vous pouvez utiliser l'un des attributs suivants :

- "unitsList" : un tableau contenant les x éléments définissant les unités disponibles (ex : "cm", "pouces", "km", "miles", etc.). Utilisez cet attribut pour définir des unités dans l'objet.
- "unitsListReference" : une référence de liste 4D contenant les unités disponibles. Utilisez cet attribut pour définir des unités à l'aide d'une liste 4D créée avec la commande [New list](#).
- "unitsListName" : un nom d'énumération 4D créée en mode Développement contenant les unités disponibles. Utilisez cet attribut pour définir des unités à l'aide d'une énumération 4D créée dans la Boîte à outils.

Quel que soit son mode de définition, la liste d'unités peut être associée à l'attribut suivant :

- "unitReference" : une valeur simple contenant l'indice (de 1 à x) de l'élément sélectionné dans la liste de valeurs "unitList", "unitsListReference" ou "unitsListName".

L'unité courante est affichée sous forme de bouton affichant successivement les valeurs de "unitList", "unitsListReference" ou "unitsListName" à chaque clic (par exemple "pixels" -> "lignes" -> "cm" -> "pixels" -> etc.)

Exemple :

Vous souhaitez définir une valeur de saisie numérique suivie d'une unité parmi deux possibles : "cm" ou "pixels". La valeur courante est "2" + "cm". Vous utilisez des valeurs définies directement dans l'objet (attribut "unitsList") :

```
ARRAY TEXT($_units;0)
APPEND TO ARRAY($_units;"cm")
APPEND TO ARRAY($_units;"pixels")
C_OBJECT($ob)
OB SET($ob;"valueType";"integer")
OB SET($ob;"value";2) // 2 "units"
OB SET($ob;"unitReference";1) //"cm"
OB SET ARRAY($ob;"unitsList";$_units)
```



alternateButton

Si vous souhaitez ajouter un bouton d'ellipse [...] dans une cellule, il suffit de passer l'attribut "alternateButton" avec la valeur vrai dans l'objet. Le bouton sera automatiquement affiché dans la cellule.

Lorsque l'utilisateur clique sur ce bouton, un événement `On Alternative Click` est généré, vous permettant de traiter cette action comme vous le souhaitez (reportez-vous ci-dessous au paragraphe "Gestion des événements" pour plus d'informations).

Exemple :

```
C_OBJECT($ob1)
$entry:="Hello world!"
OB SET($ob;"valueType";"text")
OB SET($ob;"alternateButton";True)
OB SET($ob;"value";$entry)
```



`valueType color`

L'attribut "valueType" de valeur "color" vous permet d'afficher soit une couleur, soit un texte.

- Si la valeur est un nombre, un rectangle de couleur est dessiné à l'intérieur de la cellule. Exemple :

```
C_OBJECT($ob4)
OB SET($ob4;"valueType";"color")
OB SET($ob4;"value";0x00FF0000)
```



- Si la valeur est un texte, le texte est simplement affiché (par exemple : "value";"Automatic").

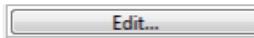
`valueType event`

L'attribut "valueType" de valeur "event" affiche un bouton qui génère simplement un événement `On Clicked` lorsque l'utilisateur clique dessus. Aucune donnée ou valeur ne peut être passée ou retournée.

Optionnellement, il est possible de passer un attribut "label".

Exemple :

```
C_OBJECT($ob)
OB SET($ob;"valueType";"event")
OB SET($ob;"label";"Edit...")
```



Gestion des événements

Plusieurs événements peuvent être gérés dans les colonnes de list box de type tableau d'objets. Voici une synthèse des événements spécifiques :

- **On Data Change:** An `On Data Change` event is triggered when any value has been modified either:
 - zone de saisie de texte
 - listes déroulante
 - zone de combo box
 - bouton d'unité (passage valeur x à valeur x+1)
 - case à cocher (passage cochée/non cochée)
- **On Clicked:** When the user clicks on a button installed using the "event" `valueType` attribute, an `On Clicked` event

will be generated. Cet événement doit être ensuite géré par le programmeur.

- On Alternative Click: When the user clicks on an ellipsis button ("alternateButton" attribute), an `On Alternative Click` event will be generated. Cet événement doit être ensuite géré par le programmeur.

Bouton image

Un bouton image est similaire à un [bouton standard](#). Cependant, contrairement à un bouton standard (qui accepte trois états : activé, désactivé et cliqué), un bouton image contient une image différente pour représenter chaque état.

Les boutons image peuvent être utilisés de deux manières :

- Comme boutons de commande dans un formulaire. Dans ce cas, le bouton image comprend généralement quatre états différents : activé, désactivé, cliqué et survolé.

Par exemple, un tableau de miniatures qui comporte une ligne de quatre colonnes, chaque miniature correspond aux états Par défaut, Cliqué, Survol et Désactivé.

Propriété	Nom JSON	Valeur
Rows	rowCount	1
Columns	columnCount	4
Switch back when Released	switchBackWhenReleased	true
Switch when Roll Over	switchWhenRollover	true
Use Last Frame as Disabled	useLastFrameAsDisabled	true

- Comme bouton permettant à l'utilisateur de choisir entre plusieurs options. Dans ce cas, le bouton image peut être utilisé à la place d'un pop-up menu image. Avec les [Pop-up menus image](#), tous les choix sont présentés simultanément (en tant que commandes du pop-up menu) ; avec un bouton image, les choix sont présentés consécutivement (à mesure que l'utilisateur clique sur le bouton).

Voici un exemple de bouton image. Vous souhaitez permettre aux utilisateurs de votre application de choisir la langue qui sera utilisée dans les menus, les boîtes de dialogue, etc. Vous pouvez implémenter cette option à l'aide d'un bouton image, placé dans une boîte de dialogue personnalisée de Propriétés :



Chaque clic modifie l'état du bouton.

Utiliser des boutons images

Un bouton image est créé de la manière suivante :

1. Tout d'abord, vous préparez une image, dans laquelle la série d'images est organisée en colonnes, en lignes, ou les deux.



Vous pouvez organiser les images sous la forme de colonnes, de lignes ou de tableaux. Dans ce dernier cas, les images sont alors numérotées de gauche à droite, ligne par ligne, en débutant par 0. Par exemple, la deuxième image de la deuxième ligne d'un tableau de 2 lignes et de 3 colonnes a pour numéro 4.

2. Puis, assurez-vous que l'image se trouve dans les ressources de votre projet et saisissez le chemin dans la propriété [Chemin d'accès image](#).
3. Définissez les propriétés de [lignes et colonnes](#) du graphique.

4. Spécifiez quand les images changent en sélectionnant les propriétés d' [animation](#) appropriées.

Animation

Outre les paramètres de positionnement et d'apparence standard, vous pouvez définir certaines propriétés spécifiques pour les boutons image, en particulier la manière et le moment où les images sont affichées. Ces options de propriétés peuvent être combinées pour améliorer vos boutons d'image.

- Par défaut ([aucune option cochée](#)), affiche l'image suivante de la série lorsque l'utilisateur clique sur le bouton; affiche l'image précédente de la série lorsque l'utilisateur effectue Majuscule+clic sur le bouton. La séquence d'images s'arrête lorsqu'on atteint la dernière image de la série. En d'autres termes, le bouton ne retourne pas à la première image de la série.

Les autres modes disponibles sont les suivants :

- [Recommencer la séquence](#)
- [Switch back when Released](#)
- [Switch when Roll Over](#)
- [Défilement continu sur clic](#)
- [Use Last Frame as Disabled](#)
- [Dernière imagette si désactivé](#)

Dernière imagette si désactivé > Dernière imagette si désactivé > Dernière imagette si désactivé > La variable associée à un bouton image retourne le numéro d'indice, dans le tableau d'imagettes, de l'image actuellement affichée. La numérotation des images dans le tableau débute à 0.

Propriétés prises en charge

[Bold](#) - [Style de la bordure](#) - [Bas](#) - [Style de bouton](#) - [Class](#) - [Colonnes](#) - [Déposable](#) - [Focusable](#) - [Police](#) - [Couleur de la police](#) - [Hauteur](#) - [Message d'aide](#) - [Dim. horizontal](#) - [Italique](#) - [Gauche](#) - [Recommencer la séquence](#) - [Nom](#) - [Chemin d'accès de l'image](#) - [Droite](#) - [Lignes](#) - [Raccouric](#) - [Action standard](#) - [Retour sur relâchement du clic](#) - [Défilement continu sur clic](#) - [Défilement tous les n ticks](#) - [Titre](#) - [Recommencer la séquence](#) - [Haut](#) - [Type](#) - [Dernière imagette si désactivé](#) - [Variable ou expression](#) - [Dim. vertical](#) - [Visibilité](#) - [Largeur](#)

Pop-up menu image

Un pop-up menu image affiche un tableau d'images bidimensionnel. Un pop-up menu image peut être utilisé à la place d'un [bouton image](#). Le mode de création d'une image destinée à être utilisée dans un pop-up menu image est identique à celui d'un bouton image. Le mode de fonctionnement de l'objet, quant à lui, s'apparente à la [Grille de boutons](#), à la différence près que l'image est utilisée comme un pop-up menu et non comme un objet du formulaire.

Utiliser des pop-up menus images

Pour créer un pop-up menu image, vous devez [faire référence à une image](#) conçue dans ce but. L'exemple suivant vous permet de sélectionner la langue d'interface de l'application à l'aide d'un pop-up menu image. Chaque langue est symbolisée par un drapeau :



Programmation

Vous pouvez gérer les pop-up menus image par l'intermédiaire de méthodes. A l'instar des [grilles de boutons](#), les variables associées au pop-up menu image prennent pour valeur le numéro de l'élément sélectionné et zéro (0) si aucun élément n'est sélectionné. et zéro (0) si aucun élément n'est sélectionné. Les éléments sont numérotés de gauche à droite et de haut en bas, à compter de l'élément situé en haut à gauche.

Aller à page

Vous pouvez associer l'[action standard](#) `Aller à page` à un pop-up menu image. Lorsque cette action est activée, 4D affiche automatiquement la page du formulaire correspondant à la position de l'image sélectionnée dans le tableau d'images. Les éléments sont numérotés de gauche à droite et de haut en bas, à compter de l'élément situé en haut à gauche.

Par exemple, si l'utilisateur clique sur le 3e élément, 4D affichera la page 3 du formulaire courant (si elle existe). Si vous souhaitez gérer vous-même l'effet du clic, conservez l'option par défaut `Pas d'action`.

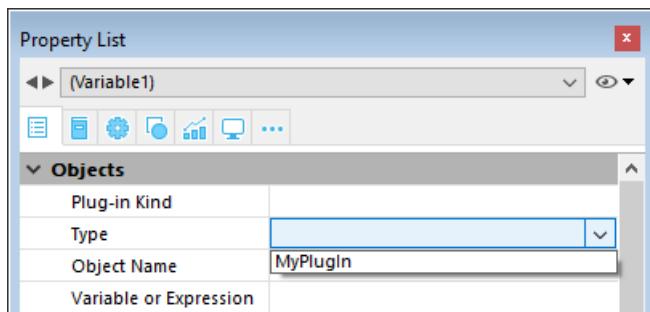
Propriétés prises en charge

[Gras](#) - [Style de la bordure](#) - [Bas](#) - [Classe](#) - [Colonnes](#) - [Hauteur](#) - [Message d'aide](#) - [Dim. horizontal](#) - [Gauche](#) - [Nom](#) - [Chemin d'accès](#) - [Droite](#) - [Lignes](#) - [Action standard](#) - [Haut](#) - [Type](#) - [Variable ou expression](#) - [Dim. vertical](#) - [Visibilité](#) - [Largeur](#)

Zone de plug-in

Une zone de plug-in est une zone du formulaire contrôlée par un plug-in. La capacité d'intégrer des plug-ins dans les formulaires permet d'accéder à des possibilités illimitées lorsque vous créez des applications personnalisées. Une zone de plug-in peut réaliser une tâche simple comme l'affichage d'une horloge numérique dans un formulaire, ou plus complexe comme proposer un environnement de traitement de textes, un tableau ou un éditeur graphique.

Lorsque vous ouvrez une application, 4D crée une liste interne des plug-ins [installés dans votre application](#). Une fois que vous avez inséré une Zone de plug-in dans un formulaire, vous pouvez sélectionner le plug-in à lui affecter via la liste Type dans la fenêtre de propriétés de l'objet :



Certains plug-ins, comme 4D Internet Commands, ne peuvent pas être utilisés dans des formulaires ou des fenêtres externes. Dans ce cas, ils n'apparaissent pas dans la liste de propriétés.

Si vous dessinez une zone de plug-in trop petite, 4D l'affiche sous forme de bouton dont le libellé est le nom de la variable associée à la zone. En exécution, l'utilisateur peut cliquer sur ce bouton afin d'ouvrir une fenêtre spécifique affichant le plug-in.

Propriétés avancées

Si les options avancées sont fournies par l'auteur du plug-in, un thème Plug-in contenant un bouton [Propriétés avancées](#) peut être activé dans la liste des propriétés. Dans ce cas, vous pouvez cliquer sur ce bouton pour définir ces options, généralement via une boîte de dialogue personnalisée.

Installer un plug-in

Pour ajouter un plug-in dans votre environnement 4D, vous devez dans un premier temps quitter votre application 4D. Le chargement des plug-ins s'effectue au lancement de l'application 4D. Pour plus d'informations sur l'installation d'un plug-in, reportez-vous à la section [Installer des plugins ou des composants](#).

Créer des plug-ins

Si vous souhaitez concevoir vos propres plug-ins, vous pouvez obtenir des informations détaillées sur l'écriture et la création de plug-ins. 4D fournit un [kit complet \(sur github\)](#) pour vous aider à écrire des plug-ins personnalisés.

Propriétés prises en charge

[Style de la bordure](#) - [Bas](#) - [Propriétés avancées](#) - [Class](#) - [Glissable](#) - [Déposable](#) - [Type d'expression](#) - [Focusable](#) - [Hauteur](#) - [Dim. horizontal](#) - [Gauche](#) - [Méthode](#) - [Nom](#) - [Type de Plug-in](#) - [Droite](#) - [Haut](#) - [Type](#) - [Variable ou expression](#) - [Dim. vertical](#) - [Visibilité](#) - [Largeur](#)

Indicateurs de progression

Un indicateur de progression (également appelé "thermomètre") est conçu pour afficher ou définir graphiquement des valeurs numériques ou date/heure.



Utiliser des indicateurs

Vous pouvez utiliser les jauge pour afficher ou définir des valeurs. Par exemple, si un indicateur de progression se voit affecter une valeur par une méthode, il affiche la valeur. Si l'utilisateur modifie manuellement la valeur indiquée par la jauge, la valeur contenue par l'objet est modifiée. Cette valeur peut être utilisée pour un autre objet tel qu'un champ, un objet saisissable ou un objet non saisissable.

La variable associée à l'indicateur contrôle l'affichage. Vous pouvez y placer des valeurs ou utiliser les valeurs qu'il stocke à l'aide des méthodes. Par exemple, la méthode suivante peut être utilisée pour contrôler un thermomètre :

```
vTherm:=[Employees]Salary
```

Cette méthode affecte à la variable vTherm la valeur du champ Salaire. Cette méthode est associée au champ Salaire.

Réciproquement, vous pouvez utiliser un indicateur pour contrôler la valeur d'un champ. L'utilisateur se sert alors de l'indicateur pour saisir la valeur du champ. Dans ce cas la méthode devient :

```
[Employees]Salary:=vTherm
```

La méthode affecte la valeur de l'indicateur au champ Salaire. Lorsque l'utilisateur modifie la valeur affichée par l'indicateur, la valeur du champ Salaire est modifiée.

Le thermomètre par défaut



Le thermomètre est l'indicateur de progression standard.

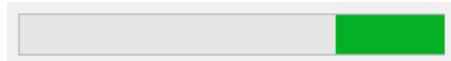
Vous pouvez afficher une barre de progression horizontale ou verticale. Ce paramètre est déterminé par la forme de l'objet que vous dessinez.

Plusieurs options graphiques sont disponibles : valeurs minimales/maximales, graduations, paliers.

Propriétés prises en charge

Barber shop - Gras - Style de bordure - Bas - Classe - Afficher graduation - Saisissable - Exécuter méthode objet - Type d'expression (uniquement "integer", "number", "date", ou "time") - Police - Couleur de police - Taille de police - Hauteur - Italique - Unité de graduation - Message d'aide - Dimensionnement horizontal - Emplacement étiquette - Gauche - Maximum - Minimum - Format numérique - Nom objet - Droite - Step - Haut - Type - Souligné - Variable ou Expression - Dimensionnement Vertical - Visibilité - Largeur

Barber shop



Le barber shop est une variante du thermomètre par défaut. Pour activer cette variante, vous devez définir la propriété du [Barber shop](#).

En code JSON, supprimez simplement la propriété "max" de l'objet thermomètre par défaut pour activer le barber shop.

Le barber shop affiche une animation continue, telle que le [spinner](#). Les thermomètres "Barber shop" sont généralement utilisés pour indiquer à l'utilisateur que le programme est en train d'effectuer une opération longue. Lorsque le thermomètre est sélectionné, le thème "[Graduations](#)" de la liste des propriétés est masqué.

A l'exécution du formulaire, l'objet n'est pas animé. Vous devez gérer l'animation en passant une valeur à [la variable ou expression qui lui est associée](#) :

- 1 (ou toute valeur différente de 0) = Démarrer l'animation,
- 0 = Stopper l'animation.

Propriétés prises en charge

[Barber shop](#) - [Gras](#) - [Style de bordure](#) - [Bas](#) - [Classe](#) - [Saisissable](#) - [Executer méthode objet](#) - [Type d'expression](#) (uniquement "integer", "number", "date", ou "time") - [Police](#) - [Couleur de police](#) - [Taille de police](#) - [Hauteur](#) - [Message d'aide](#) - [Dimensionnement horizontal](#) - [Italique](#) - [Gauche](#) - [Nom d'objet](#) - [Droite](#) - [Haut](#) - [Type](#) - [Souligné](#) - [Variable ou Expression](#) - [Dimensionnement vertical](#) - [Visibilité](#) - [Largeur](#)

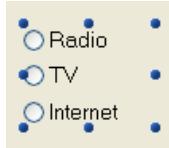
Voir aussi

- [règle](#)
- [stepper](#)

Bouton radio

Les boutons radio sont des objets qui permettent à l'utilisateur de sélectionner une valeur parmi un groupe de valeurs.

Un bouton radio apparaît sous la forme d'un texte suivi d'un cercle. Cependant, les boutons radio peuvent avoir différentes apparences.



Un bouton radio est sélectionné :

- lorsque l'utilisateur clique dessus
- lorsqu'il a le focus et que l'utilisateur appuie sur la touche Espace.

Configuration des boutons radio

Les boutons radio sont utilisés sous forme d'ensembles coordonnés : un seul bouton peut être sélectionné à la fois parmi l'ensemble. Afin de fonctionner de manière coordonnée, un ensemble de boutons radio doit partager la même propriété de **groupe radio**.

Les boutons radio sont contrôlés par des méthodes. Comme pour tous les boutons, la variable associée au bouton radio est initialisée à 0 (zéro) lorsque le formulaire est ouvert pour la première fois. Une méthode associée à un bouton radio est exécutée lorsqu'il est sélectionné. L'exemple suivant représente des boutons radio utilisés dans une base de données d'enregistrements audio et se rapporte à la vitesse d'enregistrement :



Sélectionner un bouton radio d'un groupe met ce bouton à 1 et les autres boutons du groupe à 0. Un seul bouton radio du groupe peut être sélectionné à la fois.

Vous pouvez associer **des expressions de type booléen** à des boutons radio. Dans ce cas, lorsqu'un bouton radio d'un groupe est sélectionné, sa variable est True et les variables des autres boutons radio du groupe sont False.

La valeur contenue dans un objet bouton radio n'est pas sauvegardée automatiquement (hormis s'il s'agit de la représentation d'un champ booléen) ; les valeurs des boutons radio doivent être stockées dans leurs variables et gérées à l'aide de méthodes.

Styles de bouton

Les **styles de bouton** radio contrôlent l'apparence générale du bouton radio ainsi que ses propriétés disponibles. Il est possible d'appliquer différents styles prédéfinis aux boutons radio. Cependant, le même style de bouton doit être appliqué à tous les boutons radio d'un groupe afin qu'ils fonctionnent comme prévu.

4D propose des boutons radio dans les styles prédéfinis suivants :

Classique

Le style de bouton radio Classique est un bouton système standard (c'est-à-dire une bulle avec un libellé) qui exécute le code lorsqu'un utilisateur clique dessus.



En plus de lancer l'exécution du code, le style de bouton radio Classique change la couleur de la bulle lors du survol.

A plat

Le style de bouton radio A plat est un bouton système standard (c'est-à-dire une bulle avec un libellé) qui exécute le code lorsqu'un utilisateur clique dessus.



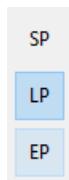
Par défaut, le style A plat a une apparence minimaliste. Le style graphique du bouton A plat est particulièrement utile pour les formulaires à imprimer.

Barre d'outils

Le style du bouton radio Barre d'outils est initialement destiné à être intégré dans une barre d'outils.

Par défaut, le style bouton Barre d'outils a un fond transparent avec un libellé au centre. En fonction du système d'exploitation, le design du bouton peut changer lorsque la souris le survole :

- *Sous Windows* - le contour du bouton apparaît.



- *Sous macOS* - le contour du bouton n'apparaît jamais.

Bevel

Le style de bouton radio Bevel est similaire au comportement du style [barre d'outils](#), à la seule différence qu'il possède un arrière-plan gris clair et un contour gris. En fonction du système d'exploitation, le design du bouton peut changer lorsque la souris le survole :

- *Sous Windows* - le contour du bouton apparaît.

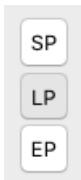


- *Sous macOS* - le contour du bouton n'apparaît jamais.

Bevel arrondi

Le style du bouton Bevel arrondi est presque identique au style [Bevel](#), à l'exception des coins du bouton qui peuvent, selon le système d'exploitation, être arrondis.

- *Sous Windows* - ce bouton est identique au style [Bevel](#).
- *Sous macOS* - les coins du bouton sont arrondis.



OS X Gradient

Le style du bouton OS X Gradient est presque identique au style [Bevel](#), à l'exception de son apparence qui peut, en fonction du système d'exploitation, avoir deux tons.

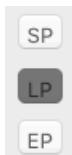
- *Sous Windows* - ce bouton est identique au style [Bevel](#).
- *Sous macOS* - le bouton s'affiche comme un bouton à deux tons.

OS X Texture

Le style du bouton radio OS X Textured est presque identique au style [Barre d'outils](#), à l'exception de son apparence qui peut, en fonction du système d'exploitation, être différente et ne pas afficher le survol.

Par défaut, le style OS X Textured apparaît comme :

- *Sous Windows* - un bouton en forme de barre d'outils avec une étiquette au centre et l'arrière-plan est toujours affiché.
- *Sous macOS* - un bouton système standard affichant un changement de couleur du gris clair au gris foncé. Sa hauteur est prédéfinie : il n'est pas possible de l'agrandir ou de la réduire.

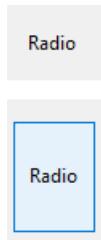


Office XP

The Office XP button style combines the appearance of the [Regular](#) style (standard system button) with the [Toolbar](#) style's behavior.

Les couleurs (surbrillance et arrière-plan) d'un bouton au style Office XP sont basées sur les couleurs du système. En fonction du système d'exploitation, le design du bouton peut changer lorsque la souris le survole :

- *Sous Windows* - son arrière-plan n'apparaît que lorsque la souris le survole.



- *Sous macOS* - son arrière-plan est toujours affiché.

Contracter/Déployer

Ce style de bouton peut être utilisé pour ajouter une icône standard contracter/déployer. Ces boutons sont utilisés nativement dans les listes hiérarchiques. Sous Windows, le bouton ressemble à un [+] ou un [-]; sous macOS, cela ressemble à un triangle pointant vers la droite ou vers le bas.



Bouton disclosure

Le style de bouton radio disclosure affiche le bouton radio comme un bouton disclosure standard, généralement utilisé pour afficher/masquer des informations supplémentaires. Le symbole du bouton pointe vers le bas avec la valeur 0 et vers le haut avec la valeur 1.



Personnalisé

Le style de bouton radio Personnalisé accepte une image d'arrière-plan personnalisée et permet de gérer des paramètres supplémentaires tels que le [décalage de l'icône](#) et les [marges](#).

Propriétés prises en charge

Tous les boutons radio partagent une même série de propriétés de base :

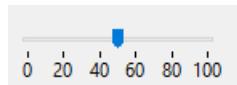
[Bold](#) - [Bottom](#) - [Button Style](#) - [Class](#) - [Expression Type](#) - [Focusable](#) - [Font](#) - [Font Color](#) - [Height](#) - [Help Tip](#) - [Horizontal Sizing](#) - [Italic](#) - [Left](#) - [Method](#) - [Object Name](#) - [Radio Group](#) - [Right](#) - [Save value](#) - [Shortcut](#) - [Title](#) - [Top](#) - [Type](#) - [Underline](#) - [Variable or Expression](#) - [Vertical Sizing](#) - [Visibility](#) - [Width](#)

Des propriétés spécifiques supplémentaires sont disponibles en fonction du [style de bouton](#) :

- [Chemin d'accès arrière-plan](#) - [Marge horizontale](#) - [Décalage icône](#) - [Marge verticale](#) (Personnalisé)
- [Nombre d'états](#) - [Chemin d'accès image](#) - [Position Titre/Image](#) (Bouton barre outils, Bevel, Bevel arrondi, OS X Gradient, OS X Textured, Office XP, Personnalisé)

Règle

La règle est un objet d'interface standard permettant de définir ou de lire une valeur à l'aide d'un curseur placé sur une règle généralement graduée.



La variable ou expression associée à l'objet peut être affectée à une zone saisissable (champ ou variable) afin de stocker ou modifier la valeur courante de l'objet.

Pour plus d'informations, veuillez vous reporter à la section [Utiliser des indicateurs](#) de la page "Indicateurs de progression".

Propriétés prises en charge

Gras - Style de la bordure - Bas - Class - Afficher graduation - Saisissable - Exécuter méthode objet - Type d'expression - Hauteur - Unité de graduation - Message d'aide - Dim. horizontal - Emplacement du libellé - Gauche - Maximum - Minimum - Format numérique - Nom - Droite - Pas - Haut - Type - Variable ou expression - Dim. vertical - Visibilité - Largeur

Voir aussi

- [indicateurs de progression](#)
- [stepper](#)

Formes

Les formes sont des [objets statiques](#) qui peuvent être ajoutées à des formes 4D.

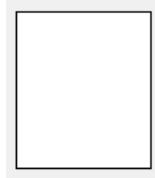
Les formes 4D prennent en charge les formes basiques suivantes :

- rectangles
- lignes
- ovales

Rectangle

Un rectangle statique est un objet décoratif contenu dans les formulaires. Les rectangles sont limités à des formes carrées.

Les rectangles sont créés à l'aide de plusieurs propriétés (couleur, épaisseur de ligne, motif, etc.). L' [arrondi](#), en particulier, des coins du rectangle peuvent être définis.



Exemple JSON :

```
"myRectangle": {  
    "type": "rectangle",      //définit le type d'objet  
    "left": 60,                //position sur la gauche du formulaire  
    "top": 160,                //position en haut du formulaire  
    "width": 100,               //largeur de l'objet  
    "height": 20,                //hauteur de l'objet  
    "borderRadius": 20        //définit le rayon d'arrondi  
}
```

Propriétés prises en charge

[Bas](#) - [Css Class](#) - [Rayon d'arrondi](#) - [Type de pointillé](#) - [Couleur de fond](#) - [Hauteur](#) - [Dimensionnement horizontal](#) - [Gauche](#) - [Couleur de ligne](#) - [Épaisseur du trait](#) - [Nom](#) - [Droite](#) - [Haut](#) - [Type](#) - [Dimensionnement vertical](#) - [Visibilité](#) - [Largeur](#)

Ligne

Une ligne statique est un objet décoratif pour les formulaires, entre deux tracés. Les lignes peuvent être horizontales, verticales ou de toute forme d'angle.

Les lignes sont créées à l'aide de plusieurs propriétés (couleur, épaisseur de ligne, etc.).

propriété startPoint

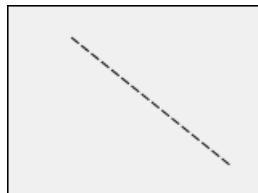
La propriété JSON `startPoint` définit à partir de quelle coordonnée la ligne peut être dessinée (voir l'exemple).

la propriété `startPoint` n'est pas exposée dans la liste des propriétés, où la direction du dessin de ligne est visible.

Exemple JSON :

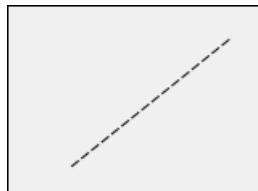
```
"myLine": {  
    "type": "line",  
    "left": 20,  
    "top": 40,  
    "width": 100,  
    "height": 80,  
    "startPoint": "topLeft", //première orientation  
    "strokeDashArray": "6 2" //pointillé  
}
```

Résultat :



```
"myLine": {  
    "type": "line",  
    "left": 20,  
    "top": 40,  
    "width": 100,  
    "height": 80,  
    "startPoint": "bottomLeft", //deuxième orientation  
    "strokeDashArray": "6 2" //pointillé  
}
```

Résultat :

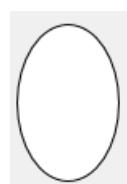


Propriétés prises en charge

[Bas](#) - [Css Class](#) - [Type de pointillé](#) - [Hauteur](#) - [Dimensionnement horizontal](#) - [Gauche](#) - [Couleur du trait](#) - [Épaisseur du trait](#) - [Nom](#) - [Droite](#) - [startPoint](#) - [Haut](#) - [Type](#) - [Dimensionnement vertical](#) - [Visibilité](#) - [Largeur](#)

Ovale

Un ovale statique est un objet décoratif contenu dans les formulaires. Les objets de forme ovale peuvent être utilisés pour dessiner des formes circulaires (lorsque les propriétés [largeur](#) et [hauteur](#) sont identiques).



Exemple JSON :

```
"myOval": {  
    "type": "oval",      //définit le type d'objet  
    "left": 60,          //position sur la gauche du formulaire  
    "top": 160,          //position en haut du formulaire  
    "width": 100,         //largeur de l'objet  
    "height": 20,         //hauteur de l'objet  
    "borderRadius": 20     //définit la couleur de fond  
}
```

Propriétés prises en charge

[Bas](#) - [Css Class](#) - [Type de pointillé](#) - [Couleur de fond](#) - [Hauteur](#) - [Dimensionnement horizontal](#) - [Gauche](#) - [Couleur du trait](#) - [Épaisseur du trait](#) - [Nom](#) - [Droite](#) - [Haut](#) - [Type](#) - [Dimensionnement vertical](#) - [Visibilité](#) - [Largeur](#)

Spinner

Le spinner est un indicateur circulaire qui affiche une animation continue, telle que le [Barber shop](#).



Ce type d'objet vous permet d'indiquer une opération telle que la recherche de connexion réseau ou le calcul est en cours. Lorsque cet indicateur est sélectionné, [les propriétés "Graduations"](#) ne sont pas disponibles.

A l'exécution du formulaire, l'objet n'est pas animé. Vous devez gérer l'animation en passant une valeur à [la variable ou expression qui lui est associée](#) :

- 1 (ou toute valeur différente de 0) = Démarrer l'animation,
- 0 = Stopper l'animation

Propriétés prises en charge

[Style de la bordure](#) - [Bas](#) - [Class](#) - [Type d'expression](#) - [Hauteur](#) - [Message d'aide](#) - [Dim. horizontal](#) - [Gauche](#) - [Nom](#) - [Droite](#) - [Haut](#) - [Type](#) - [Variable ou expression](#) - [Dim. vertical](#) - [Visibilité](#) - [Largeur](#)

Séparateur

Un séparateur divise un formulaire en deux zones. Il permet à l'utilisateur d'agrandir ou de réduire chaque zone en le déplaçant. Un séparateur peut être horizontal ou vertical. Il tient compte des propriétés de redimensionnement des objets, ce qui permet de personnaliser entièrement l'interface. Un séparateur peut être "pousseur" ou non.

L'utilisation type du séparateur est le formulaire de sortie dans lequel les colonnes peuvent être redimensionnées :

Job Title:	Company:
Secretary	Howard Battery Co.
Salesperson	Howard Battery Co.
Salesperson	Howard Battery Co.
Supervisor	Howard Battery Co.
Director	BluePines

Les caractéristiques générales des séparateurs sont les suivantes :

- Vous pouvez placer autant de séparateurs que vous voulez dans tout type de formulaire. De même, il est possible de mêler des séparateurs horizontaux et verticaux dans un même formulaire.
- Un séparateur peut traverser un objet. Celui-ci sera redimensionné lors du déplacement du séparateur.
- Les butées des séparateurs sont calculées de manière à ce que les objets déplacés restent entièrement visibles dans le formulaire ou ne passent pas sous/à côté d'un autre séparateur. Lorsque la propriété **Pousseur** est associée à un séparateur, son déplacement vers la droite ou vers le bas ne rencontre pas de butée.
- Les redimensionnements effectués dans les formulaires à l'aide des séparateurs ne sont conservés que durant l'affichage du formulaire. Une fois le formulaire refermé, les dimensions initiales sont restaurées.

Une fois inséré, un séparateur se présente sous la forme d'un trait. Vous pouvez modifier son [style de bordure](#) afin d'obtenir un trait plus ou moins épais, ou [modifier sa couleur](#).

Exemple JSON :

```
"mySplitter": {  
    "type": "splitter",  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20,  
    "splitterMode": "move" //pousseur  
}
```

Propriétés prises en charge

[Style de la bordure](#) - [Bas](#) - [Class](#) - [Hauteur](#) - [Message d'aide](#) - [Dim. horizontal](#) - [Gauche](#) - [Couleur du trait](#) - [Nom](#) - [Pousseur](#) - [Droite](#) - [Haut](#) - [Type](#) - [Dim. vertical](#) - [Variable ou expression](#) - [Visibilité](#) - [Largeur](#)

Interaction avec les propriétés des objets environnants

Dans un formulaire, les séparateurs interagissent sur les objets qui les entourent suivant les options de redimensionnement de ces objets :

Options de redimensionnement du ou des objet(s)	Objet(s) au-dessus du séparateur horizontal ou à gauche du séparateur vertical (1)	Object(s) below an horizontal <i>non-Pusher</i> splitter or to the right of a vertical <i>non-Pusher</i> splitter	Object(s) below an horizontal <i>Pusher</i> splitter or to the right of a vertical <i>Pusher</i> splitter
Aucun	Restent tel que	Sont déplacés avec le séparateur (conservent leur position relative) jusqu'à la butée suivante. La butée du déplacement vers le bas ou vers la droite est soit le bord de la fenêtre, soit un autre séparateur.	Sont déplacés sans limites avec le séparateur (conservent leur position relative). Aucune butée n'est appliquée (cf. paragraphe suivant)
Redimensionnement	Gardent leur position d'origine mais sont redimensionnés en fonction de la nouvelle position du séparateur		
Déplacer	Se déplacent avec le séparateur		

(1) Un objet situé à cet emplacement sert de butée en cas de déplacement vers le haut (séparateur horizontal) ou vers la gauche (séparateur vertical) s'il ne comporte aucune option de redimensionnement.

Un objet entièrement contenu dans le rectangle définissant le séparateur est déplacé en même temps que le séparateur lui-même.

Gestion programmée des séparateurs

Vous pouvez associer une méthode objet à un séparateur. Cette méthode sera appelée avec l'événement `On Clicked` durant tout le déplacement.

A `variable of the Longint type` is associated with each splitter. Cette variable peut être utilisée dans vos méthodes objet et/ou formulaire. Elle prend pour valeur le déplacement courant, en pixels, du séparateur.

- Si elle est négative : le déplacement a été effectué vers le haut ou vers la gauche,
- Si elle est positive : le déplacement a été effectué vers le bas ou vers la droite,
- Si elle est égale à 0 : le séparateur a été relâché à son emplacement d'origine.

Vous pouvez également déplacer le séparateur par programmation : il suffit de modifier la valeur de la variable associée. Imaginons par exemple qu'un séparateur vertical soit associé à la variable `sépara1`. Si vous écrivez `sépara1:=-10`, le séparateur sera déplacé de 10 pixels vers la gauche — comme si l'utilisateur l'avait fait manuellement. Le déplacement s'effectue au terme de l'exécution de la méthode objet ou formulaire contenant l'instruction.

Image statique

Les images statiques sont des [objets statiques](#) pouvant être utilisées à des fins diverses dans les formulaires 4D, notamment comme décor, arrière-plan ou interface utilisateur :



Les images statiques sont stockées à l'extérieur des formulaires et insérées par référence. Dans l'éditeur de formules, les objets image statique sont créées par copier-coller ou par glisser-déposer.

Si vous placez une image statique dans la page 0 d'un formulaire multi-pages, elle apparaîtra comme élément d'arrière-plan de toutes les pages. Vous pouvez également l'inclure dans un formulaire hérité, qui s'applique à l'arrière-plan de différents autres formulaires. Dans les deux cas, votre application s'exécutera plus rapidement.

Format et emplacement

L'image d'origine doit être stockée dans un format géré nativement par 4D (4D reconnaît les principaux formats d'image : JPEG, PNG, BMP, SVG, GIF, etc.).

Deux emplacements principaux peuvent être utilisés pour le chemin d'image statique :

- in the Resources folder of the project. Appropriate when you want to share static pictures between several forms in the project. Dans ce cas, le chemin d'accès se trouve dans "/RESOURCES/<picture path>".
- dans un dossier d'images (nommé Images par exemple) dans le dossier du formulaire. Convient lorsque les images statiques sont utilisées uniquement dans le formulaire et/ou lorsque vous souhaitez pouvoir déplacer ou dupliquer le formulaire entier dans un ou plusieurs projets. Dans ce cas, le chemin d'accès est "<¥picture path>" et est résolu à partir de la racine du dossier du formulaire.

Propriétés prises en charge

[Bas](#) - [CSS Class](#) - [Affichage](#) - [Hauteur](#) - [Dim. horizontal](#) - [Gauche](#) - [Nom](#) - [Pathname](#) - [Droite](#) - [Haut](#) - [Type](#) - [Dim. vertical](#) - [Visibilité](#) - [Largeur](#)

Stepper

Un stepper permet à l'utilisateur de faire défiler des valeurs numériques, des durées (heures) ou des dates par des étapes pré-définies en cliquant sur les boutons de direction.

Stepper associated with `vStep` variable



Utilisation du stepper

La variable associée à l'objet peut être affectée à une zone saisissable (champ ou variable) afin de stocker ou modifier la valeur courante de l'objet.

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

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

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

Pour plus d'informations, veuillez vous reporter à la section [Utiliser des indicateurs](#) de la page "Indicateurs de progression".

Propriétés prises en charge

[Style de la bordure](#) - [Bas](#) - [Class](#) - [Saisissable](#) - [Exécuter méthode objet](#) - [Type d'expression](#) (uniquement "entier", "numérique", "date", ou "heure") - [Hauteur](#) - [Message d'aide](#) - [Dim. horizontal](#) - [Gauche](#) - [Maximum](#) - [Minimum](#) - [Nom](#) - [Droite](#) - [Step](#) - [Haut](#) - [Type](#) - [Variable ou expression](#) - [Dim. vertical](#) - [Visibilité](#) - [Largeur](#)

Voir aussi

- [indicateurs de progression](#)
- [règle](#)

Sous-formulaire

Un sous-formulaire est un formulaire inclus dans un autre formulaire.

Terminologie

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

- Subform: a form intended for inclusion in another form, itself called the parent form.
- Parent form: a form containing one or more subform(s).
- Subform container: an object included in the parent form, displaying an instance of the subform.
- Subform instance: the representation of a subform in a parent form. Cette notion est importante car il est possible d'afficher plusieurs instances d'un même sous-formulaire dans un formulaire parent.
- List form: instance of subform displayed as a list.
- Detail form: page-type input form associated with a list-type subform that can be accessed by double-clicking in the list.

Sous-formulaires en liste

Un sous-formulaire en liste vous permet de saisir, visualiser et modifier des données dans d'autres tables. Les sous-formulaires en liste sont généralement utilisés avec les bases de données utilisant des liens de type 1 vers N. Un sous-formulaire en liste affiche les enregistrements de la table N liée par un lien automatique de type 1 vers N. Vous pouvez disposer de plusieurs sous-formulaires provenant de différentes tables dans le même formulaire. En revanche, il n'est pas possible de placer deux sous-formulaires appartenant à la même table dans une même page de formulaire.

Par exemple, une base de gestion de contacts peut utiliser une instance de sous-formulaire en liste pour afficher tous les contacts d'une société. Bien que les contacts apparaissent dans l'écran général, l'information est en fait stockée dans la table liée. A l'aide d'un lien 1 vers N, la conception de cette base de données rend facile le stockage d'un nombre illimité de contacts pour chacune des sociétés. Avec des liens automatiques, vous pouvez permettre la saisie de données dans la table liée sans programmation.

Bien que les sous-formulaires en liste soient généralement associés aux tables N, une instance de sous-formulaire peut afficher des enregistrements de toute autre table de la base de données.

Vous pouvez également permettre à l'utilisateur de saisir des données dans le formulaire liste. Suivant la configuration du sous-formulaire, l'utilisateur pourra afficher le formulaire détaillé en double-cliquant sur un sous-enregistrement ou en utilisant les commandes d'ajout et de modification des sous-enregistrements.

4D propose trois actions standard, permettant de répondre aux besoins élémentaires de gestion des sous-enregistrements : **Modifier sous-enregistrement**, **Supprimer sous-enregistrement** et **Ajouter sous-enregistrement**. Lorsque le formulaire comporte plusieurs instances de sous-formulaires, l'action s'applique au sous-formulaire ayant le focus.

Sous-formulaires en page

Les sous-formulaires en mode page peuvent afficher des données relatives à l'enregistrement courant ou toute valeur pertinente en fonction du contexte (variables, images, etc.). Ils peuvent également, et c'est là leur intérêt majeur, comporter des fonctionnalités avancées et interagir avec le formulaire parent (widgets). Les sous-formulaires en page bénéficient de propriétés et d'événements spécifiques, et peuvent être entièrement contrôlés par programmation.

Le sous-formulaire en page utilise le formulaire entrée désigné par la propriété **Formulaire détaillé**. A la différence d'un sous-formulaire en mode liste, le formulaire utilisé peut provenir de la même table que le formulaire parent. Il est également possible d'utiliser un formulaire projet. En exécution, un sous-formulaire en mode page dispose des caractéristiques d'affichage standard d'un formulaire entrée.

Les widgets 4D sont des objets composés prédéfinis. Ils sont décrits en détail dans le manuel [4D Widgets](#).

Using the bound variable or expression

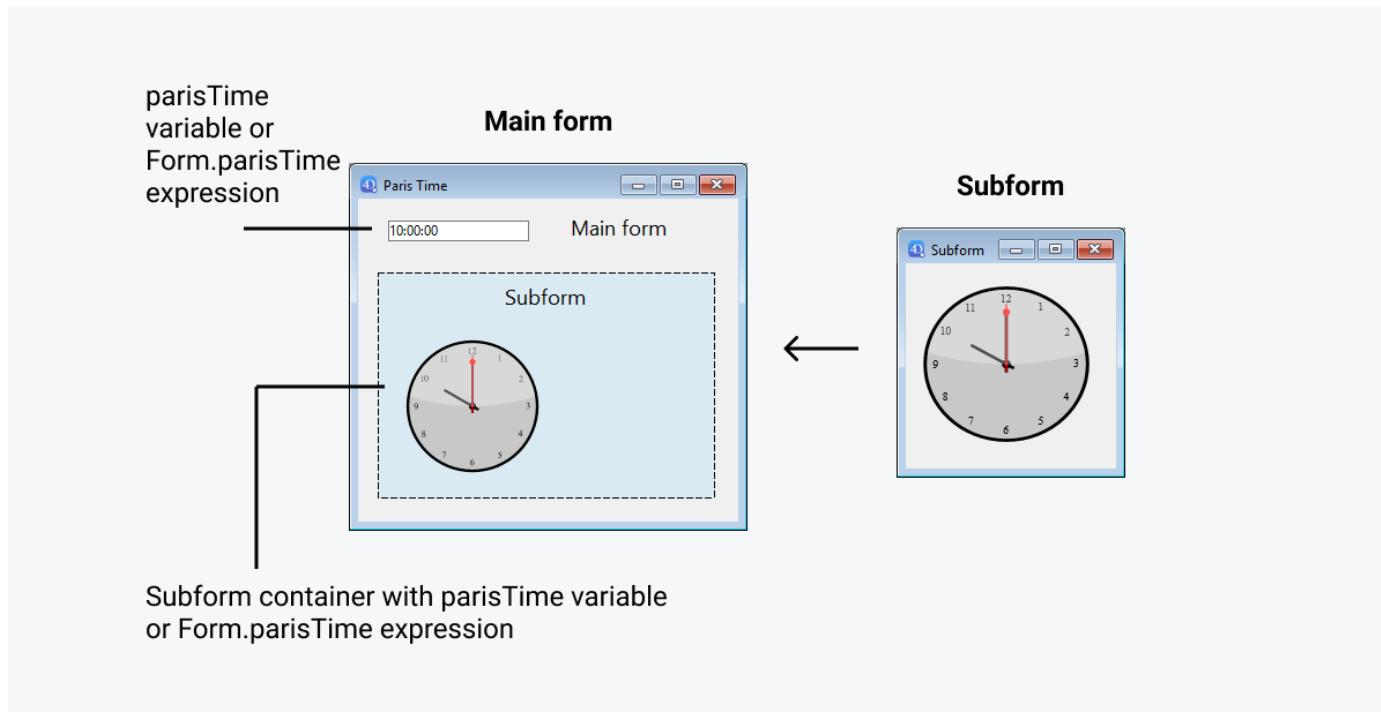
You can bind [a variable or an expression](#) to a subform container object. This is very useful to synchronize values from the parent form and its subform(s).

By default, 4D creates a variable or expression of [object type](#) for a subform container, which allows you to share values in the context of the subform using the [Form](#) command ([see below](#)). However, you can use a variable or expression of any scalar type (time, integer, etc.) especially if you only need to share a single value:

- Define a bound variable or expression of a scalar type and call the `OBJECT Get subform container value` and `OBJECT SET SUBFORM CONTAINER VALUE` commands to exchange values when [On Bound Variable Change](#) or [On Data Change](#) form events occur. This solution is recommended to synchronize a single value.
- Define a bound variable or expression of the object type and use the [Form](#) command to access its properties from the subform. This solution is recommended to synchronize several values.

Synchronizing parent form and subform (single value)

Binding the same variable or expression to your subform container and other objects of the parent form lets you link the parent form and subform contexts to put the finishing touches on sophisticated interfaces. Imagine a subform representing a clock, inserted into a parent form containing an enterable variable of the Time type:



In the parent form, both objects (time variable and subform container) ***have the same value as ***Variable or Expression*****. It can be a variable (e.g. `parisTime`), or an expression (e.g. `Form.parisTime`).

In the subform, the clock object is managed through the `Form.clockValue` property.

Updating the contents of a subform

Case 1: The value of the parent form variable or expression is modified and this modification must be passed on to a subform.

`Form.parisTime` changes to 12:15:00 in the parent form, either because the user entered it, or because it was updated dynamically (via the `Current time` command for example). This triggers the [On Bound Variable Change](#) event in the subform's `Form` method.

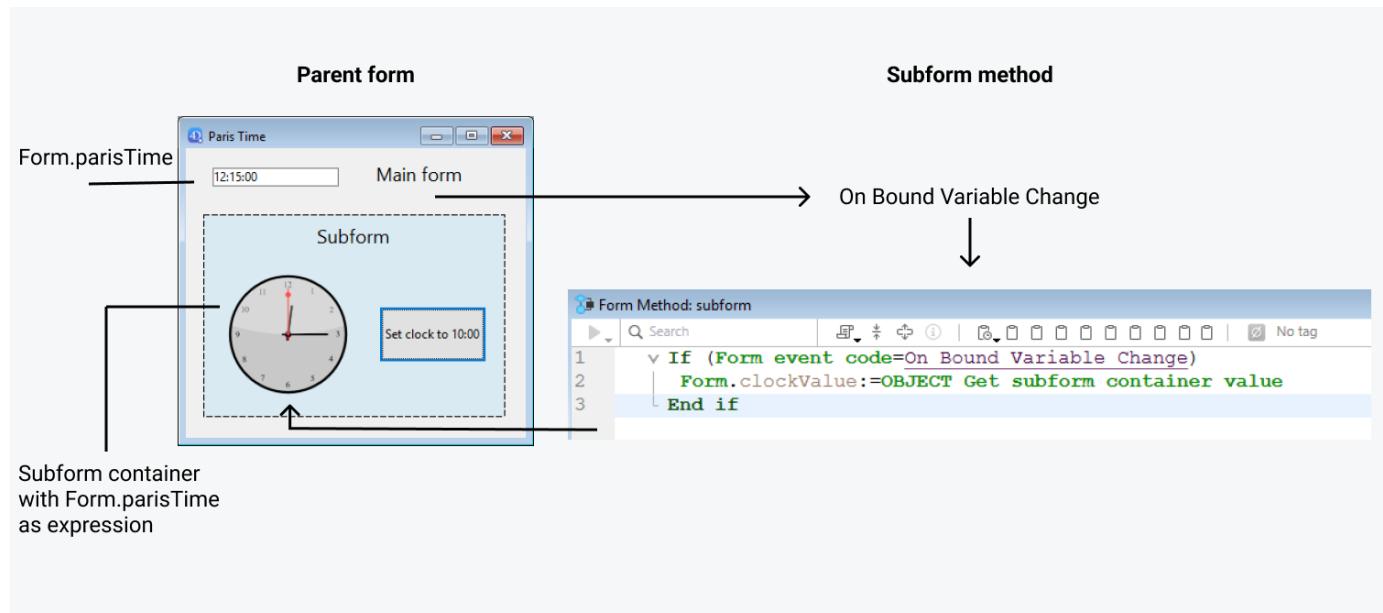
The following code is executed:

```

// Subform form method
If (Form event code=On Bound Variable Change) //bound variable or expression was modified in the parent
    Form.clockValue:=OBJECT Get subform container value //synchronize the local value
End if

```

It updates the value of `Form.clockValue` in the subform:



The [On Bound Variable Change](#) form event is generated:

- as soon as a value is assigned to the variable/expression of the parent form, even if the same value is reassigned
- si le sous-formulaire appartient à la page formulaire courante ou à la page 0.

Note that, as in the above example, it is preferable to use the `OBJECT Get subform container value` command which returns the value of the expression in the subform container rather than the expression itself because it is possible to insert several subforms in the same parent form (for example, a window displaying different time zones contains several clocks).

Modifying the bound variable or expression triggers form events which let you synchronize the parent form and subform values:

- Use the [On Bound Variable Change](#) form event to indicate to the subform (form method of subform) that the variable or expression was modified in the parent form.
- Use the [On Data Change](#) form event to indicate to the subform container that the variable or expression value was modified in the subform.

Updating the contents of a parent form

Scénario 2 : Le contenu du sous-formulaire est modifié et cette modification doit être répercutee dans le formulaire parent.

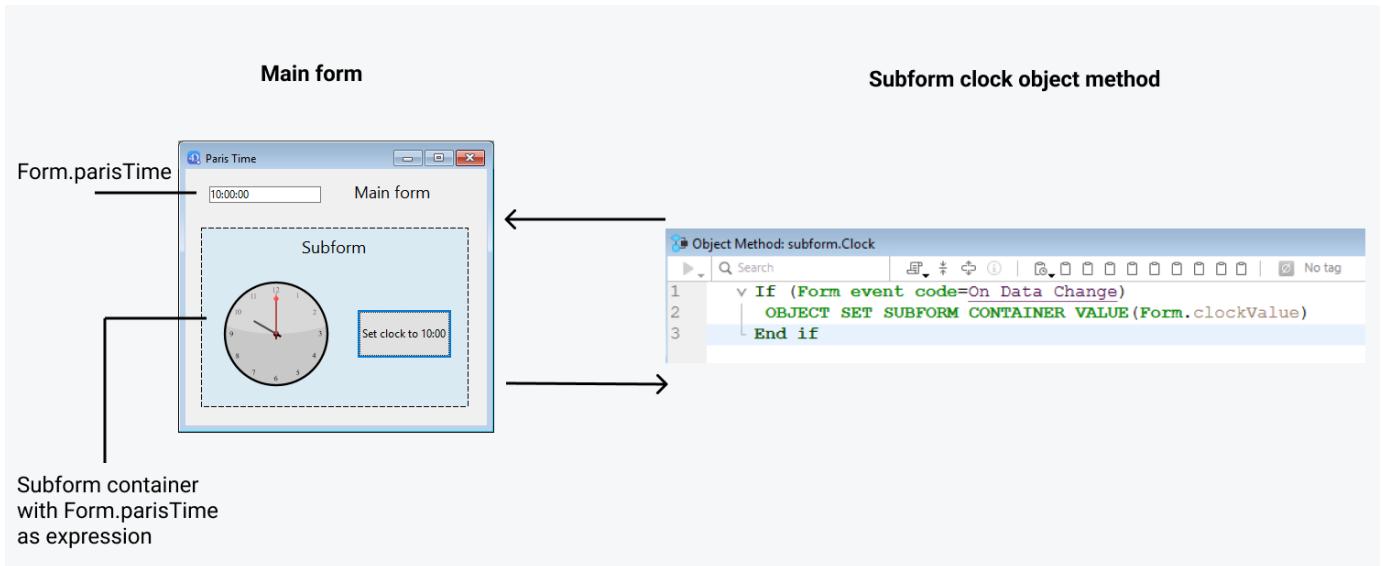
Inside the subform, the button changes the value of the `Form.clockValue` expression of type Time attached to the clock object. This triggers the [On Data Change](#) form event inside the clock object (this event must be selected for the object), which updates the `Form.parisTime` value in the main form.

The following code is executed:

```

// subform clock object method
If (Form event code=On Data Change) //whatever the way the value is changed
    OBJECT SET SUBFORM CONTAINER VALUE(Form.clockValue) //Push the value to the container
End if

```



Everytime the value of `Form.clockValue` changes in the subform, `Form.parisTime` in the subform container is also updated.

If the variable or expression value is set at several locations, 4D uses the value that was loaded last. It applies the following loading order: 1-Object methods of subform, 2-Form method of subform, 3-Object methods of parent form, 4-Form method of parent form

Synchronizing parent form and subform (multiple values)

By default, 4D binds a variable or expression of [object type](#) to each subform. The contents of this object can be read and/or modified from within the parent form and from the subform, allowing you to share multiple values in a local context.

When bound to the subform container, this object is returned by the `Form` command directly in the subform. Since objects are always passed by reference, if the user modifies a property value in the subform, it will automatically be saved in the object itself and thus, available to the parent form. On the other hand, if a property of the object is modified by the user in the parent form or by programming, it will be automatically updated in the subform. No event management is necessary.

For example, in a subform, inputs are bound to the `Form` object properties (of the subform form):



In the parent form, you display the subform twice. Each subform container is bound to an expression which is a property of the `Form` object (of the parent form):

Father

Form.father.lastname	Form or Expression: Form.father
Lastname:	Form.lastname
Firstname:	Form.firstname

Mother

Form.mother.lastname	Form or Expression: Form.mother
Lastname:	Form.lastname
Firstname:	Form.firstname

Add values

The button only creates `mother` and `father` properties in the parent's `Form` object:

```
//Add values button object method
Form.mother:=New object("lastname"; "Hotel"; "firstname"; "Anne")
Form.father:=New object("lastname"; "Golf"; "firstname"; "Félix")
```

When you execute the form and click on the button, you see that all values are correctly displayed:

The screenshot shows a 4D form with two subforms. The top subform is titled 'Father' and contains fields for Lastname ('Golf') and Fristname ('Félix'). The bottom subform is titled 'Mother' and contains fields for Lastname ('Hotel') and Fristname ('Anne'). At the bottom of the main form is a blue-bordered button labeled 'Add values'.

If you modify a value either in the parent form or in the subform, it is automatically updated in the other form because the same object is used:

The screenshot shows the same 4D form after modifications. The 'Father' subform's Lastname field now contains 'Wolf'. The 'Mother' subform's Lastname field now contains 'Hotelle'. Red arrows point from the modified values back to their respective source fields in the other form, illustrating bidirectional synchronization.

Using pointers (compatibility)

In versions prior to 4D v19 R5, synchronization between parent forms and subforms was handled through pointers. For example, to update a subform object, you could call the following code:

```
// Subform form method
If (Form event code=On Bound Variable Change)
    ptr:=OBJECT Get pointer(Object subform container)
    clockValue:=ptr->
End if
```

This principle is still supported for compatibility but is now deprecated since it does not allow binding expressions to subforms. It should no longer be used in your developments. In any cases, we recommend to use the `Form` command or the `OBJECT Get subform container value` and `OBJECT SET SUBFORM CONTAINER VALUE` commands to synchronize form and subform values.

Programmation inter-formulaires avancée

Communication between the parent form and the instances of the subform may require going beyond the exchange of a values through the bound variable. En effet, vous pouvez souhaiter mettre à jour des variables dans les sous-

formulaires en fonction d'actions effectuées dans le formulaire parent et inversement. Si l'on reprend l'exemple du sous-formulaire de type "pendule dynamique", on peut souhaiter définir une ou plusieurs heures d'alerte par pendule.

Pour répondre à ces besoins, 4D propose les mécanismes suivants :

- Calling of a container object from the subform using the `CALL SUBFORM CONTAINER` command
- Execution of a method in the context of the subform via the `EXECUTE METHOD IN SUBFORM` command

La commande `GOTO OBJECT` peut rechercher l'objet de destination dans le formulaire parent même si elle exécutée depuis un sous-formulaire.

Commande CALL SUBFORM CONTAINER

The `CALL SUBFORM CONTAINER` command lets a subform instance send an `event` to the subform container object, which can then process it in the context of the parent form. L'événement est reçu dans la méthode de l'objet conteneur. Il peut s'agir à l'origine de tout événement détecté par le sous-formulaire (clic, glisser-déposer, etc.).

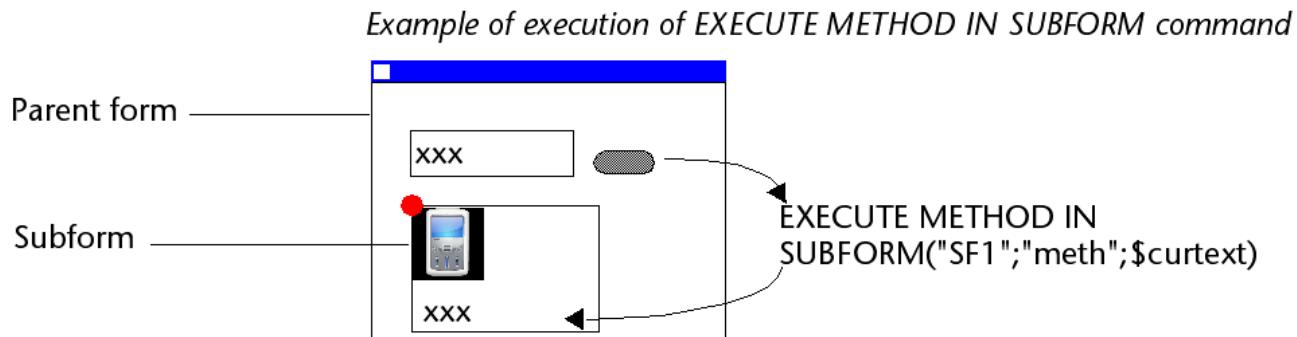
Le code de l'événement est libre (par exemple, 20000 ou -100). Vous pouvez soit utiliser un code correspondant à un événement existant (par exemple, 3 pour `Sur validation`), soit utiliser un code personnalisé. Dans le premier cas, seuls les événements présents dans la liste des événements "cochables" des conteneurs de sous-formulaire peuvent être utilisés (cf. Liste des propriétés). Dans le second cas, le code ne doit correspondre à aucun événement formulaire existant. Il est conseillé d'utiliser une valeur négative pour avoir l'assurance que 4D n'utilisera pas ce code dans les versions futures.

Pour plus d'informations, reportez-vous à la description de la commande `CALL SUBFORM CONTAINER`.

Commande EXECUTE METHOD IN SUBFORM

La commande `EXECUTE METHOD IN SUBFORM` permet à un formulaire ou à l'un de ses objets de demander l'exécution d'une méthode dans le contexte de l'instance du sous-formulaire, ce qui lui donne accès aux variables, objets, etc., du sous-formulaire. Cette méthode peut en outre recevoir des paramètres.

Ce mécanisme est illustré dans le schéma suivant :



Pour plus d'informations, reportez-vous à la description de la commande `EXECUTE METHOD IN SUBFORM`.

Propriétés prises en charge

[Border Line Style](#) - [Bottom](#) - [Class](#) - [Detail Form](#) - [Double click on empty row](#) - [Double click on row](#) - [Enterable in list](#) - [Expression Type](#) - [Focusable](#) - [Height](#) - [Hide focus rectangle](#) - [Horizontal Scroll Bar](#) - [Horizontal Sizing](#) - [Left](#) - [List Form](#) - [Method](#) - [Object Name](#) - [Print Frame](#) - [Right](#) - [Selection mode](#) - [Source](#) - [Top](#) - [Type](#) - [Variable or Expression](#) - [Vertical Scroll Bar](#) - [Vertical Sizing](#) - [Visibility](#) - [Width](#)

Onglets

Un onglet crée un objet qui permet à l'utilisateur de choisir entre plusieurs écrans virtuels affichés dans les limites de l'onglet. L'utilisateur accède à chaque écran en cliquant sur l'onglet correspondant.

Le formulaire multi-pages suivant utilise un onglet :

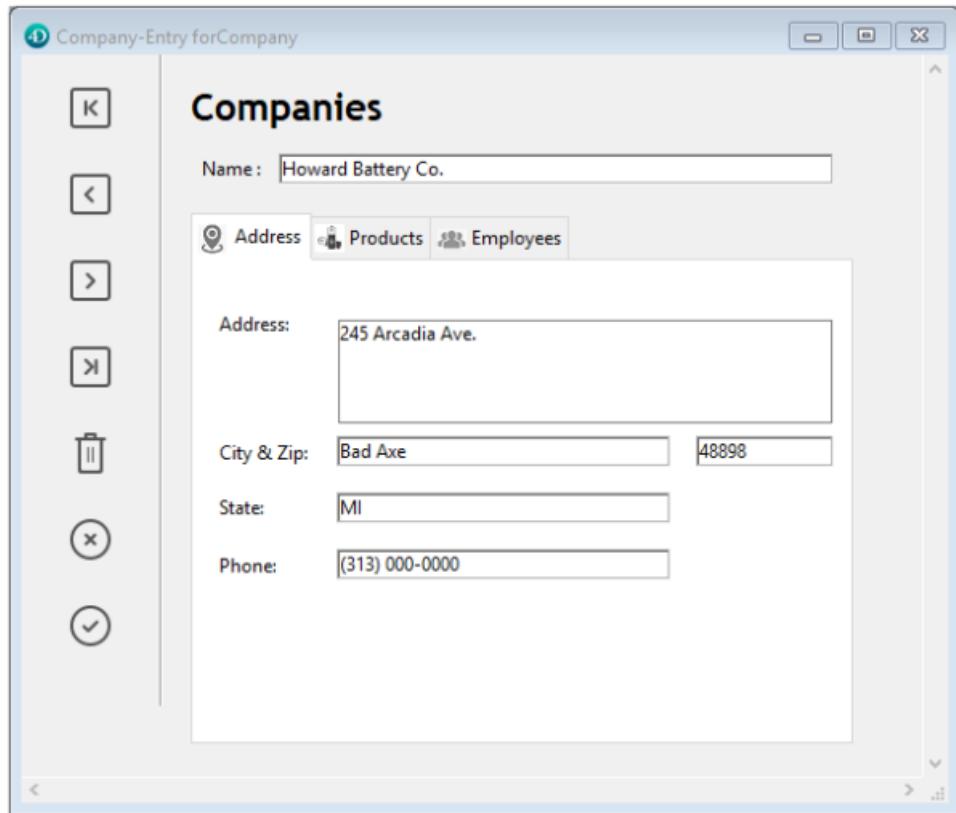
The screenshot shows a Windows-style application window titled "Company-Entry". On the left side, there is a vertical toolbar with several icons: a letter "K", left and right arrows, a trash can, and a checkmark. The main area is titled "Companies". It contains a text input field for "Name" with the value "Howard Battery Co.". Below this is a horizontal tab bar with three tabs: "Address", "Products", and "Employees". The "Address" tab is currently selected. Under the "Address" tab, there are four input fields: "Address" (containing "245 Arcadia Ave."), "City & Zip" (containing "Bad Axe" and "48898"), "State" (containing "MI"), and "Phone" (containing "(313) 000-0000").

Pour passer d'un écran à l'autre, l'utilisateur clique simplement sur l'onglet correspondant.

Un onglet peut être utilisé, entre autres, pour gérer la navigation entre les pages d'un formulaire multi-pages. Dans ce cas, la commande `FORM GOTO PAGE` ou l'action standard `gotoPage` devra être appelée lorsque l'utilisateur cliquera sur l'onglet.

Un onglet peut aussi être utilisé pour contrôler les données qui sont affichées dans un sous-formulaire. On peut, par exemple, implémenter un rolodex à l'aide d'un onglet. Chaque onglet afficherait alors une des lettres de l'alphabet et l'action de l'onglet serait de charger les informations correspondantes à la lettre sur lequel l'utilisateur a cliqué.

Chaque onglet peut afficher des intitulés ou des intitulés et des petites icônes. Si vous placez des icônes, elles apparaissent à gauche de chaque intitulé. Voici un exemple d'onglet qui utilise des icônes :



Lorsque vous créez un onglet, 4D gère l'espacement et le placement des onglets. Vous n'avez à fournir à 4D que les intitulés sous la forme d'un tableau ou les icônes et intitulés sous la forme d'une énumération hiérarchique.

Si l'onglet est assez large, il affiche les intitulés et les icônes. S'il ne peut pas afficher toutes les icônes à la fois, il place des flèches de défilement à droite du dernier onglet visible. Les flèches de défilement permettent à l'utilisateur de faire défiler des onglets vers la droite ou vers la gauche.

Sous macOS, les onglets peuvent être orientés, en plus de la position standard (en haut), à droite, à gauche ou en bas.

Exemple JSON :

```
"myTab": {
    "type": "tab",
    "left": 60,
    "top": 160,
    "width": 100,
    "height": 20,
    "labelsPlacement": "bottom" //définit l'orientation
}
```

Ajouter les intitulés dans un onglet

Pour fournir les étiquettes d'un onglet, vous pouvez utiliser :

- un objet
- une liste déroulante
- un tableau

Utilisation d'un objet

Vous pouvez affecter un [objet](#) encapsulant une [collection](#) comme [source de données](#) de l'onglet. The object must contain the following properties:

Propriété	Type	Description
values	Collection	Mandatory - Collection of scalar values. Seules les valeurs de type chaîne sont prises en charge. Si elle est invalide, vide ou non définie, l'onglet est vide
index	number	Indice de la page de l'onglet en cours (valeur comprise entre 0 et <code>collection.length-1</code>)
currentValue	Text	Valeur courante sélectionnée

Le code d'initialisation doit être exécuté avant que le formulaire ne soit présenté à l'utilisateur.

Dans l'exemple suivant, `Form.tabControl` a été défini comme `expression` de l'onglet. Vous pouvez associer l'`action standard gotoPage` à l'objet form :

```
Form.tabControl:=New object
Form.tabControl.values:=New collection("Page 1"; "Page 2"; "Page 3")
Form.tabControl.index:=2 //démarrage à la page 3
```

Utiliser une énumération

Vous pouvez associer à l'onglet `une liste de valeurs`, accessible via une collection (liste statique) ou un pointeur JSON vers une liste json ("\$ref"). Les icônes associées à des éléments de liste dans l'éditeur de listes seront affichées dans l'onglet.

Utiliser un tableau texte

Vous pouvez créer un tableau Texte qui contient les noms de chaque page du formulaire. Le code doit être exécuté avant que le formulaire soit présenté à l'utilisateur. Par exemple, vous pouvez placer ce code dans l'événement formulaire `Sur chargement`.

```
ARRAY TEXT(arrPages;3)
arrPages{1}:="Name"
arrPages{2}:="Address"
arrPages{3}:="Notes"
```

Vous pouvez également stocker les noms des pages dans une liste hiérarchique et utiliser la commande `LIST TO ARRAY` pour charger les valeurs dans le tableau.

Fonctionnalités de Goto page

Commande FORM GOTO PAGE

Vous pouvez utiliser la commande `FORM GOTO PAGE` dans la méthode de l'onglet pour naviguer parmi les pages du formulaire :

```
FORM GOTO PAGE(arrPages)
```

Cette commande est exécutée dans l'événement formulaire `Sur clic`. Il est préférable d'effacer le tableau dans l'événement formulaire `Sur libération`.

Vous pouvez, par exemple, écrire le code suivant :

```
Case of
:(Form event=On Load)
  LIST TO ARRAY("Tab Labels";arrPages)
:(Form event=On Clicked)
  FORM GOTO PAGE(arrPages)
:(Form event=On Unload)
  CLEAR VARIABLE(arrPages)
End case
```

Action Goto Page

Lorsque vous associez l'action standard

gotoPage à un objet de type Onglet, 4D affiche automatiquement la page du formulaire correspondant au numéro de l'onglet sélectionné.

Par exemple, si l'utilisateur clique sur le 3e onglet, 4D affichera la page 3 du formulaire courant (si elle existe).

Propriétés prises en charge

Gras - Bas - Énumération - Css Class - Type d'expression - Police - Taille - Hauteur - Message d'aide - Dim. horizontal - Italique - Gauche - Nom - Droite - Valeur standard - Action standard - Orientation onglets - Haut - Type - Souligné - Dim. vertical - Variable ou expression - Visibilité - Largeur

Text

Un objet texte vous permet d'afficher du contenu écrit statique (ex : instructions, titres, étiquettes, etc.) dans un formulaire. Ces zones de texte statique peuvent devenir dynamiques lorsqu'elles incluent des références dynamiques. Pour plus d'informations, reportez-vous à la section [Utiliser des références dans les textes statiques](#).

Exemple JSON :

```
"myText": {  
    "type": "text",  
    "text": "Hello World!",  
    "textAlign": "center",  
    "left": 60,  
    "top": 160,  
    "width": 100,  
    "height": 20,  
    "stroke": "#ff0000"      //couleur texte  
    "fontWeight": "bold"  
}
```

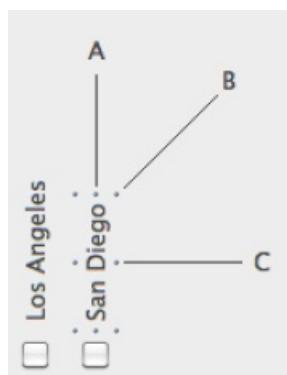
Rotation

4D vous permet d'effectuer des rotations de zones de texte dans vos formulaires à l'aide de la propriété [Orientation](#).

	New York	Chicago	Los Angeles	San Diego
Alpha	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Bravo	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Charlie	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Delta	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Echo	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

La fonctionnalité de rotation de texte est disponible via la commande [OBJECT SET TEXT ORIENTATION](#).

Une fois qu'un texte a été orienté, il reste possible de modifier sa taille ou sa position ainsi que toutes ses propriétés. A noter que les propriétés de hauteur et de largeur de la zone de texte ne sont pas dépendantes de l'orientation :



- Si l'objet est redimensionné dans la direction A, sa [largeur](#) sera modifiée ;
- Si l'objet est redimensionné dans la direction C, sa [hauteur](#) sera modifiée ;
- Si l'objet est redimensionné dans la direction B, sa [largeur](#) et sa [hauteur](#) seront modifiées.

Propriétés prises en charge

Gras - Style de la bordure - Bas - Css Class - Couleur de fond - Police - Couleur de la police - Taille Hauteur -
Alignement horizontal - Dim. horizontal - Italique - Gauche - Nom - Orientation - Droite - Titre - Haut - Type -
Souligné - Dim. vertical - Visibilité - Largeur

Zone Web

Les zones Web (Web Areas) peuvent afficher tout type de contenu Web à l'intérieur de vos formulaires : pages HTML au contenu statique ou dynamique, fichiers, images, JavaScript, etc. Le moteur de rendu de la zone web dépend de la plate-forme d'exécution de l'application et de l'[option de moteur de rendu](#) sélectionnée.

Il est possible de créer plusieurs zones web dans un même formulaire. A noter cependant que l'utilisation de zones web est soumise à [quelques limitations](#).

Plusieurs [actions standard](#), de nombreuses [commandes de langage](#) et [événements formulaires](#) génériques et dédiés permettent au développeur de contrôler le fonctionnement des zones web. Des variables spécifiques permettent d'échanger des informations entre la zone et l'environnement 4D.

Propriétés spécifiques

Variables associées

Deux variables spécifiques sont automatiquement associées à chaque zone web :

- [URL](#) -- pour contrôler l'URL affiché par la zone web
- [Progression](#) -- pour contrôler le pourcentage de chargement de la page affichée dans la zone web.

À partir de 4D v19 R5, la variable de Progression n'est plus mise à jour dans les zones Web utilisant le [moteur de rendu du système Windows](#).

Moteur de rendu Web

Vous pouvez choisir entre [deux moteurs de rendus](#) pour la zone web, en fonction des spécificités de votre application.

La sélection du moteur de rendu Web intégré vous permet d'appeler des méthodes 4D à partir de la zone Web et de vous assurer que les fonctionnalités sur macOS et Windows sont similaires. La sélection du moteur de rendu système est recommandée lorsque la zone web est connectée à Internet car elle bénéficie toujours des dernières mises à jour de sécurité.

Accéder aux méthodes 4D

Lorsque la propriété [Accès méthodes 4D](#) est cochée, vous pouvez appeler des méthodes 4D à partir d'une zone web.

Cette propriété n'est disponible que si la zone web [utilise le moteur de rendu web intégré](#).

Objet \$4d

Le [moteur de rendu Web intégré 4D](#) fournit à la zone un objet JavaScript nommé \$4d que vous pouvez associer à n'importe quelle méthode projet 4D à l'aide de la notation objet ".".

Par exemple, pour appeler la méthode 4D `HelloWorld`, vous devez simplement exécuter la déclaration suivante :

```
$4d.HelloWorld();
```

JavaScript est sensible à la casse. Il est donc important de noter que l'objet est nommé \$4d (avec un "d" minuscule).

La syntaxe des appels aux méthodes 4D est la suivante :

```
$4d.4DMethodName(param1,paramN,function(result){})
```

- `param1...paramN` : Vous pouvez passer autant de paramètres que vous le souhaitez dans la méthode 4D. Ces paramètres peuvent être de n'importe quel type pris en charge par JavaScript (chaîne, numérique, tableau, objet).
- `function(result)` : Fonction à passer comme dernier argument. Cette fonction "callback" est appelée de manière synchronisée une fois que la méthode 4D a fini de s'exécuter. Elle reçoit le paramètre `result`.
- `result` : Résultat de l'exécution de la méthode 4D, retournée dans l'expression "\$0". Ce résultat peut être de n'importe quel type pris en charge par JavaScript (chaîne, numérique, tableau, objet). Vous pouvez utiliser la commande `C_OBJECT` pour retourner les objets.

Par défaut, 4D opère en UTF-8. Lorsque vous retournez du texte contenant des caractères étendus, tels que des caractères avec des accents, assurez-vous que l'encodage de la page affiché dans la zone Web est déclaré en UTF-8, sinon les caractères risquent de ne pas être retournés correctement. Dans ce cas, ajoutez la ligne suivante dans la page HTML pour déclarer l'encodage : `<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />`

Exemple 1

Considérons une méthode projet 4D nommée `today` qui ne reçoit pas de paramètres et qui retourne la date courante dans une chaîne.

Code 4D de la méthode `today` :

```
C_TEXT($0)
$0:=String(Current date;System date long)
```

Dans la zone web, la méthode 4D peut être appelée avec la syntaxe suivante :

```
$4d.today()
```

La méthode 4D ne reçoit aucun paramètre mais elle retourne la valeur \$0 à la fonction callback appelée par 4D après avoir exécuté la méthode. Nous souhaitons afficher la date dans la page HTML qui est chargée par la zone web.

Voici le code de la page HTML :

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<script type="text/javascript">
$4d.today(function(dollarZero)
{
    var curDate = dollarZero;
    document.getElementById("mydiv").innerHTML=curDate;
});
</script>
</head>
<body>Today is: <div id="mydiv"></div>
</body>
</html>
```

Exemple 2

La méthode projet 4D `calcSum` reçoit des paramètres (`$1...$n`) et retourne leur somme dans `$0` :

Code 4D de la méthode `calcSum` :

```

C_REAL(${1}) // reçoit n paramètres de type REEL
C_REAL($0) // retourne un Réel
C_LONGINT($i;$n)
$n:=Count parameters
For($i;1;$n)
    $0:=$0+${$i}
End for

```

Le code d'exécution JavaScript dans la zone web est le suivant :

```

$4d.calcSum(33, 45, 75, 102.5, 7, function(dollarZero)
{
    var result = dollarZero // le résultat est 262.5
});

```

Actions standard

Quatre actions standard sont disponibles pour gérer automatiquement les zones Web : Open Back URL , Open Forward URL , Refresh Current URL et Stop Loading URL . Ces actions peuvent être associées à des boutons ou des commandes de menu et permettre une implémentation rapide d'interfaces Web basiques. Ces actions sont décrites dans [Actions standard](#).

Événements formulaire

Des événements formulaire spécifiques sont destinés à la gestion programmée des zones web, concernant notamment l'activation des liens :

- [On Begin URL Loading](#)
- [On URL Resource Loading](#)
- [On End URL Loading](#)
- [On URL Loading Error](#)
- [On URL Filtering](#)
- [On Open External Link](#)
- [On Window Opening Denied](#)

En outre, les zones web prennent en charge les événements formulaire génériques suivants :

- [On Load](#)
- [On Unload](#)
- [On Getting Focus](#)
- [On Losing Focus](#)

Notes d'utilisation des zones Web

Interface utilisateur

Lors de l'exécution du formulaire, l'utilisateur dispose des fonctions d'interface standard des navigateurs dans la zone web, ce qui lui permet d'interagir avec les autres zones du formulaire :

- Commandes Edit menu : lorsque la zone web a le focus, les commandes du menu Edit permettent d'effectuer les actions de copier, coller, tout sélectionner, etc., en fonction de la sélection.
- Le menu contextuel : il est possible d'utiliser le [menu contextuel](#) standard du système avec la zone web. L'affichage de ce menu peut également être contrôlé via la commande WA SET PREFERENCE .
- Glisser-déposer : l'utilisateur peut effectuer des glisser-déposer de textes, d'images ou de documents à l'intérieur d'une zone web ou entre une zone web et les objets des formulaires 4D, en fonction des propriétés des objets 4D.

Pour des raisons de sécurité, le changement du contenu d'une zone web via le glisser-déposer d'un fichier ou d'un URL n'est pas autorisé par défaut. Dans ce cas, le curseur affiche une icône d'interdiction  . Vous devez utiliser l'instruction `WA SET PREFERENCE(*;"warea";WA enable URL drop;True)` pour afficher une icône "drop" et générer l'événement `On Window Opening Denied` . Dans cet événement, vous pouvez appeler la commande `WA OPEN URL` ou définir la variable URL en réponse à un dépôt utilisateur.

Les fonctions de glisser-déposer décrites ci-dessus ne sont pas prises en charge dans les zones Web utilisant le moteur de rendu du système macOS .

Sous-formulaires

Pour des raisons liées aux mécanismes de redessinancement des fenêtres, l'insertion d'une zone web dans un sous-formulaire est soumise aux contraintes suivantes :

- Le sous-formulaire ne doit pas pouvoir défiler,
- Les limites de la zone web ne doivent pas dépasser de la zone du sous-formulaire

La superposition d'une zone web au dessus ou en-dessous d'autres objets formulaires n'est pas prise en charge.

Conflit Zone Web et serveur Web (Windows)

Sous Windows, il est déconseillé d'accéder via une zone web au serveur Web de l'application 4D contenant la zone car cette configuration peut provoquer un conflit paralysant l'application. Bien entendu, un 4D distant peut accéder au serveur Web du 4D Server, mais pas à son propre serveur web.

Insertion du protocole (macOS)

Les URLs manipulés par programmation dans les zones web sous macOS doivent débuter par le protocole. Par exemple, vous devez passer la chaîne "<http://www.monsite.fr>" et non uniquement "www.monsite.fr".

Accès à l'inspecteur web

Vous pouvez visualiser et utiliser un inspecteur web dans les zones web de vos formulaires ou dans les zones web hors écran. Il permet d'analyser le code et les flux d'information des pages web.

Pour afficher l'inspecteur web, vous pouvez soit exécuter la commande `WA OPEN WEB INSPECTOR`, soit utiliser le menu contextuel de la zone web.

- Execute the `WA OPEN WEB INSPECTOR` command
This command can be used directly with onscreen (form object) and offscreen web areas.
- Use the web area context menu
This feature can only be used with onscreen web areas and requires that the following conditions are met:
 - le **menu contextuel** de la zone Web est activé
 - l'utilisation de l'inspecteur est expressément autorisée dans la zone via la déclaration suivante :

```
WA SET PREFERENCE(*;"WA";WA enable Web inspector;True)
```

Avec le moteur de rendu du système Windows , une modification de cette préférence nécessite une action de navigation dans la zone (par exemple, un rafraîchissement de la page) pour être prise en compte.

Pour plus d'informations, reportez-vous à la description de la commande `WA SET PREFERENCE`.

Lorsque les paramétrages décrits ci-dessus sont effectués, vous disposez de nouvelles options telles que **Inspect Element** dans le menu contextuel de la zone. Lorsque vous sélectionnez cette option, le débogueur de la zone web est alors affiché.

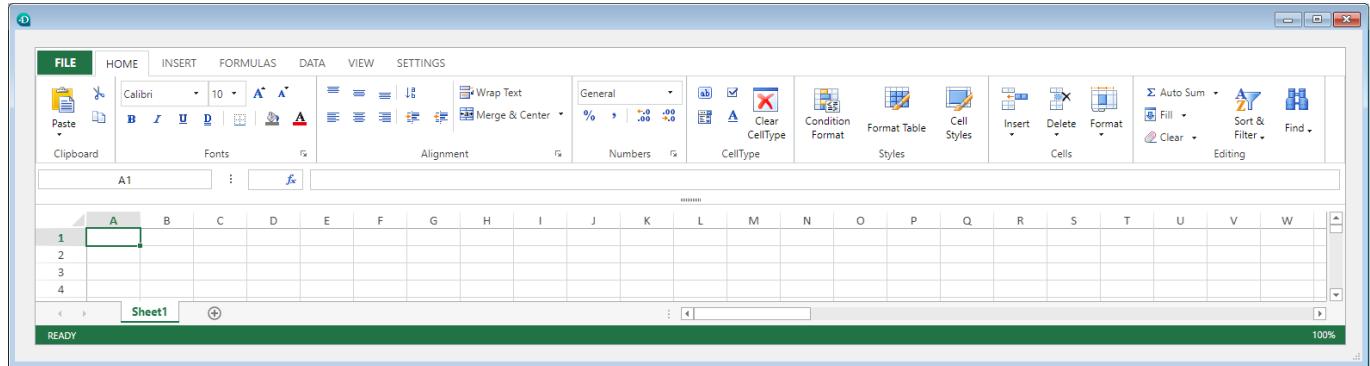
Pour une description détaillée des fonctionnalités de ce débogueur, veuillez vous reporter à la documentation du moteur de rendu web utilisé.

Propriétés prises en charge

[Style de la bordure](#) - [Bas](#) - [CSS Class](#) - [Menu contextuel](#) - [Hauteur](#) - [Dim. horizontal](#) - [Gauche](#) - [Méthode](#) - [Nom](#) - [Progression](#) - [Droite](#) - [Haut](#) - [Type](#) - [URL](#) - [Utiliser le moteur de rendu Web intégré](#) - [Variable ou expression](#) - [Dim. vertical](#) - [Visibilité](#) - [Largeur](#)

4D View Pro area

4D View Pro vous permet d'insérer et d'afficher une zone de tableau dans vos formulaires 4D. Une tableur est une application contenant une grille de cellules dans lesquelles vous pouvez saisir des informations, effectuer des calculs ou afficher des images.



Une fois que vous utilisez les zones 4D View Pro dans vos formulaires, vous pouvez importer et exporter des feuilles de calcul.

Utiliser des zones 4D View Pro

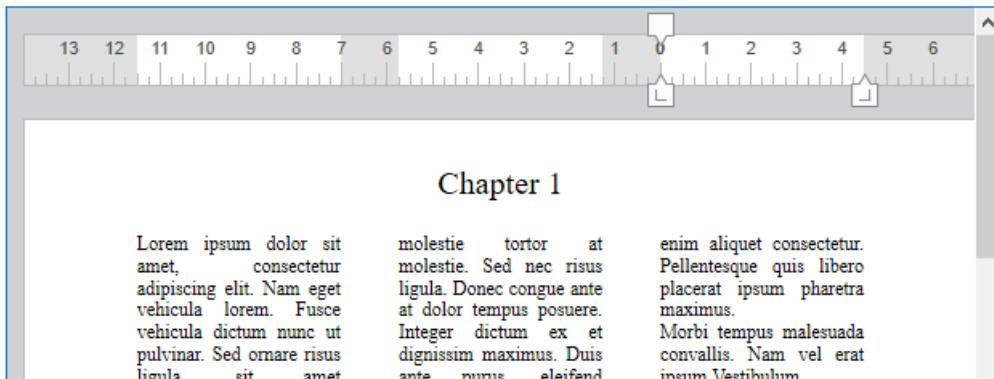
Les zones 4D View Pro sont documentées dans [la section 4D View Pro](#).

Propriétés prises en charge

Style de la bordure - Bas - Css Class - Haut - Dim. horizontal - Gauche - Méthode - Nom - Droite - Afficher barre de formule - Type - Interface utilisateur - Dim. vertical - Visibilité - Largeur

4D Write Pro area

4D Write Pro offers 4D users an advanced word-processing tool, fully integrated with your 4D application. En utilisant 4D Write Pro, vous pouvez rédiger des e-mails et/ou des lettres pré-configurés contenant des images, une signature, du texte formaté et des placeholders pour des variables dynamiques. Vous pouvez également générer dynamiquement des factures ou des rapports, contenant du texte et des images formatés.



Utiliser des zones 4D Write Pro

Les zones 4D Write Pro sont documentées dans le manuel [4D Write Pro](#).

Propriétés prises en charge

Correction orthographique - Ligne de bordure - Bas - CSS Class - Menu contextuel - Glissable - Déposable - Saisissable - Focusable - Hauteur - Cacher rectangle de focus - Barre de défilement horizontal - Dimensionnement horizontal - Configuration du clavier - Gauche - Méthode - Nom - Impression taille variable - Résolution - Droite - Sélection toujours visible - Montrer le fond - Montrer les pieds de page - Montrer les entêtes - Montrer les caractères cachés - Montrer la règle horizontale - Montrer HTML WYSIWYG - Montrer le cadre de la page - Montrer les références - Montrer la règle verticale - Type - Dimensionnement vertical - Barre de défilement vertical - Mode d'affichage - Visibilité - Largeur - Zoom

Liste de propriétés JSON

Vous trouverez dans cette page une liste complète de toutes les propriétés d'objets triées selon leur nom JSON. Cliquez sur un nom de propriété pour accéder à sa description détaillée.

Dans le chapitre "Propriétés des objets de formulaire", les propriétés sont triées en fonction des noms et des thèmes de la liste des propriétés.

a - b - c - d - e - f - g - h - i - j - k - l - m - n - p - r - s - t - u - v - w - z

Propriété	Description	Valeurs possibles
a		
<code>action</code>	Une action à exécuter.	Nom d'une action standard valide.
<code>allowFontColorPicker</code>	Permet d'afficher le sélecteur de polices système ou le sélecteur de couleurs pour modifier les attributs d'un objet	true, false (par défaut)
<code>alternateFill</code>	Permet de définir une couleur d'arrière-plan différente pour les lignes / colonnes impaires dans une list box.	Toutes les valeurs css; "transparent"; "automatic"; "automaticAlternate"
<code>automaticInsertion</code>	Permet d'ajouter automatiquement une valeur à une liste lorsqu'un utilisateur saisit une valeur qui ne se trouve pas dans l'énumération associée à l'objet.	true, false
b		
<code>booleanFormat</code>	Indique seulement deux valeurs possibles.	true, false
<code>borderRadius</code>	La valeur du rayon d'arrondi pour les rectangles à coins arrondis.	minimum : 0
<code>borderStyle</code>	Permet de définir un style standard pour la bordure de l'objet.	"system", "none", "solid", "dotted", "raised", "sunken", "double"
<code>bottom</code>	Positionne un objet en bas (centré).	minimum : 0
c		
<code>choiceList</code>	Associe une énumération à un objet	Une énumération
<code>class</code>	Une liste de mots séparés par des espaces utilisés comme sélecteurs de classe dans les fichiers css.	Une liste de noms de classes
<code>columnCount</code>	Nombre de colonnes.	minimum: 1
<code>columns</code>	Une collection de colonnes list box	Collection d'objets colonne avec des propriétés de colonnes définies
<code>contextMenu</code>	Fournit à l'utilisateur l'accès à un menu contextuel standard dans la zone sélectionnée.	"automatic", "none"
<code>continuousExecution</code>	Indique s'il faut exécuter non la méthode d'un objet pendant que l'utilisateur écrit le contrôle	true, false

Propriété	l'utilisateur suit le contrôle.	
<code>controlType</code>	Description Indique comment la valeur doit être retournée dans une cellule de listbox.	Valeurs possibles "input", "checkbox" (pour les colonnes booléen / numérique), "automatic", "popup" (uniquement pour les colonnes booléens)
<code>currentItemSource</code>	Le dernier élément sélectionné dans une list box.	Expression d'objet
<code>currentItemPositionSource</code>	La position du dernier élément sélectionné dans une listbox.	Expression numérique
<code>customBackgroundPicture</code>	Définit l'image qui sera dessinée en arrière-plan du bouton.	Chemin relatif en syntaxe POSIX. Doit être utilisé avec l'option "Personnalisé" de la propriété "Style".
<code>customBorderX</code>	Définit la taille (en pixels) des marges horizontales internes d'un objet. Doit être utilisé avec l'option "Personnalisé" de la propriété "Style".	minimum : 0
<code>customBorderY</code>	Définit la taille (en pixels) des marges verticales internes d'un objet. Doit être utilisé avec l'option "Personnalisé" de la propriété "Style".	minimum : 0
<code>customOffset</code>	Définit une valeur d'offset personnalisée en pixels. Doit être utilisé avec l'option "Personnalisé" de la propriété "Style".	minimum : 0
<code>customProperties</code>	Propriétés avancées (le cas échéant)	Chaîne JSON ou chaîne encodée en base64
d		
<code>dataSource</code> (objects) <code>dataSource</code> (subforms) <code>dataSource</code> (array list box) <code>dataSource</code> (Collection or entity selection list box) <code>dataSource</code> (list box column) <code>dataSource</code> (hierarchical list box)	Indique la source des données.	A 4D variable, field name, or an arbitrary complex language expression.
<code>dataSourceTypeHint</code> (objects) <code>dataSourceTypeHint</code> (list box column, drop-down list)	Indique le type de variable.	"integer", "boolean", "number", "picture", "text", date", "time", "arrayText", "arrayDate", "arrayTime", "arrayNumber", "collection", "object", "undefined"
<code>dateFormat</code>	Controls the way dates appear when displayed or printed. Must only be selected among the 4D built-in formats.	"systemShort", "systemMedium", "systemLong", "iso8601", "rfc822", "short", "shortCentury", "abbreviated", "long", "blankIfNull" (peut être combiné avec les autres valeurs possibles)
<code>defaultButton</code>	Modifies a button's appearance in order to indicate the recommended choice to the user.	true, false
<code>defaultValue</code>	Defines a value or a stamp to be entered by default in an input object	Chaîne ou "#D", "#H", "#N"
<code>deletableInList</code>	Specifies if the user can delete subrecords in a list subform	true, false

Propriété		
<code>form</code> (list box)	Associates a detail form with a list	Valeurs possibles (string) of table or project
<code>detailForm</code> (subform)	subform.	form, a POSIX path (string) to a .json file describing the form, or an object describing the form
<code>display</code>	The object is drawn or not on the form.	true, false
<code>doubleClickInEmptyAreaAction</code>	Action to perform in case of a double-click on an empty line of a list subform.	"addSubrecord" ou "" to do nothing
<code>doubleClickInRowAction</code> (list box) <code>doubleClickInRowAction</code> (subform)	Action to perform in case of a double-click on a record.	"editSubrecord", "displaySubrecord"
<code>dpi</code>	Screen resolution for the 4D Write Pro area contents.	0=automatic, 72, 96
<code>dragging</code>	Enables dragging function.	"none", "custom", "automatic" (hors énumération, list box)
<code>dropping</code>	Enables dropping function.	"none", "custom", "automatic" (hors énumération, list box)
e		
<code>enterable</code>	Indicates whether users can enter values into the object.	true, false
<code>enterableInList</code>	Indicates whether users can modify record data directly in the list subform.	true, false
<code>entryFilter</code>	Associates an entry filter with the object or column cells. This property is not accessible if the Enterable property is not enabled.	Text to narrow entries
<code>events</code>	Liste de tous les événements sélectionnés pour l'objet ou le formulaire	Collection de noms d'événements, ex : ["onClick", "onDataChange"...].
<code>excludedList</code>	Allows setting a list whose values cannot be entered in the column.	Une liste de valeurs à exclure.
f		
<code>border-style</code>	Définit la couleur de fond d'un objet.	Any CSS value, "transparent", "automatic"
<code>focusable</code>	Indicates whether the object can have the focus (and can thus be activated by the keyboard for instance)	true, false
<code>fontFamily</code>	Specifies the name of font family used in the object.	Nom d'une famille de police CSS
<code>fontSize</code>	Sets the font size in points when no font theme is selected	minimum : 0
<code>fontStyle</code>	Le texte sélectionné est légèrement penché vers la droite.	"normal", "italic"
<code>fontTheme</code>	Sets the automatic style	"normal", "main", "additional"
<code>fontWeight</code>	Le texte sélectionné est plus foncé et plus épais.	"normal", "bold"

<code>rowHeight</code>	Used to set the row height	pattern <code>^(¥d+)(px em)?\$</code> (positive decimal + px/em)
<code>frameDelay</code>	Permet de parcourir le contenu du bouton d'image à la vitesse spécifiée (en graduations).	minimum : 0
<code>g</code>		
<code>graduationStep</code>	Mesure de l'affichage de l'échelle.	minimum : 0
<code>h</code>		
<code>header</code>	Defines the header of a list box column	Object with properties "text", "name", "icon", "dataSource", "fontWeight", "fontStyle", "tooltip"
<code>headerHeight</code>	Used to set the row height	pattern <code>^(¥d+)(px em)?\$</code> (positive decimal + px/em)
<code>height</code>	Designates an object's vertical size	minimum : 0
<code>hideExtraBlankRows</code>	Deactivates the visibility of extra, empty rows.	true, false
<code>hideFocusRing</code>	Hides the selection rectangle when the object has the focus.	true, false
<code>hideSystemHighlight</code>	Used to specify hiding highlighted records in the list box.	true, false
<code>highlightSet</code>	string	Name of the set.
<code>horizontalLineStroke</code>	Définit la couleur des lignes horizontales dans une list box (gris par défaut).	Any CSS value, "transparent", "automatic"
<code>i</code>		
<code>icon</code>	The pathname of the picture used for buttons, check boxes, radio buttons, list box headers.	Chemin relatif ou filesystem en syntaxe POSIX.
<code>iconFrames</code>	Sets the exact number of states present in the picture.	minimum: 1
<code>iconPlacement</code>	Désigne l'emplacement d'une icône par rapport à l'objet formulaire.	"aucun", "gauche", "droite"
<code>k</code>		
<code>keyboardDialect</code>	To associate a specific keyboard layout to an input.	A keyboard code string, e.g. "ar-ma"
<code>l</code>		
<code>labels</code>	A list of values to be used as tab control labels	ex: "a", "b", "c", ...
<code>labelsPlacement</code> (objects) <code>labelsPlacement</code> (tab control)	Indique l'emplacement du texte d'un objet.	"none", "top", "bottom", "left", "right"
<code>layoutMode</code>	Mode for displaying the 4D Write Pro document in the form area.	"page", "draft", "embedded"
<code>left</code>	Positions an object on the left.	minimum : 0
<code>list</code> , see <code>choiceList</code>	A list of choices associated with a hierarchical list	Une énumération
<code>listboxType</code>	The list box data source.	"array", "currentSelection", "namedSelection", "collection"

Propriété <code>listForm</code>	Description List form to use in the subform.	Valeurs possibles Name (string) of table or project form, a POSIX path (string) to a .json file describing the form, or an object describing the form
<code>lockedColumnCount</code>	Number of columns that must stay permanently displayed in the left part of a list box.	minimum : 0
<code>loopBackToFirstFrame</code>	Les images sont affichées en boucle continue.	true, false
m		
<code>max</code>	The maximum allowed value. For numeric steppers, these properties represent seconds when the object is associated with a time type value and are ignored when it is associated with a date type value.	minimum: 0 (pour les types de données numériques)
<code>maxWidth</code>	Designates the largest size allowed for list box columns.	minimum : 0
<code>metaSource</code>	A meta object containing style and selection settings.	An object expression
<code>method</code>	Le nom d'une méthode projet.	Le nom d'une méthode projet existante
<code>methodsAccessibility</code>	Which 4D methods can be called from a Web area	"none" (par défaut), "all"
<code>min</code>	The minimum allowed value. For numeric steppers, these properties represent seconds when the object is associated with a time type value and are ignored when it is associated with a date type value.	minimum: 0 (pour les types de données numériques)
<code>minWidth</code>	Designates the smallest size allowed for list box columns.	minimum : 0
<code>movableRows</code>	Autorise le déplacement des lignes pendant l'exécution.	true, false
<code>multiline</code>	Handles multiline contents.	"yes", "no", "automatic"
n		
<code>name</code>	The name of the form object. (Optional for the form)	Any name which does not belong to an already existing object
<code>numberFormat</code>	Controls the way the alphanumeric fields and variables appear when displayed or printed.	Numbers (including a decimal point or minus sign if necessary)
p		
<code>picture</code>	The pathname of the picture for picture buttons, picture pop-up menus, or static pictures	Relative or filesystem path in POSIX syntax, or "var:<variableName>" for picture variable.
<code>pictureFormat</code> (input, list box column ou footer) <code>pictureFormat</code> (static picture)	Controls how pictures appear when displayed or printed.	"truncatedTopLeft", "scaled", "truncatedCenter", "tiled", "proportionalTopLeft" (excluding static pictures), "proportionalCenter" (excluding static pictures)

<code>placeholder</code>	Specifies text when the data source value is empty.	Text to be displayed out.
<code>pluginAreaKind</code>	Describes the type of plug-in.	The type of plug-in.
<code>popupPlacement</code>	Allows displaying a symbol that appears as a triangle in the button, which indicates that there is a pop-up menu attached.	"None", "Linked", "Separated"
<code>printFrame</code>	Print mode for objects whose size can vary from one record to another depending on their contents	"fixed", "variable", (sous-formulaire uniquement) "fixedMultiple"
<code>progressSource</code>	A value between 0 and 100, representing the page load completion percentage in the Web area. La variable est mise à jour automatiquement par 4D. Il n'est pas possible de la modifier manuellement.	minimum : 0
r		
<code>radioGroup</code>	Enables radio buttons to be used in coordinated sets: only one button at a time can be selected in the set.	Radio group name
<code>requiredList</code>	Allows setting a list where only certain values can be inserted.	Une liste de valeurs obligatoires.
<code>resizable</code>	Designates if the size of an object can be modified by the user.	"true", "false"
<code>resizingMode</code>	Specifies if a list box column should be automatically resized	"rightToLeft", "legacy"
<code>right</code>	Positions an object on the right.	minimum : 0
<code>rowControlSource</code>	A 4D array defining the list box rows.	Tableau
<code>rowCount</code>	Sets the number of rows.	minimum: 1
<code>rowFillSource</code> (array list box) <code>rowFillSource</code> (selection ou collection list box)	The name of an array or expression to apply a custom background color to each row of a list box.	The name of an array or expression.
<code>rowHeight</code>	Sets the height of list box rows.	CSS value unit "em" or "px" (default)
<code>rowHeightAuto</code>	boolean	"true", "false"
<code>rowHeightAutoMax</code>	Designates the largest height allowed for list box rows.	CSS value unit "em" or "px" (default). minimum : 0
<code>rowHeightAutoMin</code>	Designates the smallest height allowed for list box rows.	CSS value unit "em" or "px" (default). minimum : 0
<code>rowHeightSource</code>	An array defining different heights for the rows in a list box.	Nom d'une variable tableau 4D.
<code>rowStrokeSource</code> (array list box) <code>rowStrokeSource</code> (selection ou collection/entity selection list box)	An array or expression for managing row colors.	Name of array or expression.
<code>rowStyleSource</code> (array list box) <code>rowStyleSource</code> (selection ou collection/entity selection list box)	An array or expression for managing row styles.	Name of array or expression.

Propriété	Description	Valeurs possibles
<code>saveAs</code> (list box column) <code>saveAs</code> (drop-down list)	The type of contents to save in the field or variable associated to the form object	"value", "reference"
<code>scrollbarHorizontal</code>	A tool allowing the user to move the viewing area to the left or right.	"visible", "hidden", "automatic"
<code>scrollbarVertical</code>	A tool allowing the user to move the viewing area up or down.	"visible", "hidden", "automatic"
<code>selectedItemsSource</code>	Collection of the selected items in a list box.	Collection expression
<code>selectionMode</code> (hierarchical list) <code>selectionMode</code> (list box) <code>selectionMode</code> (subform)	Allows the selection of multiple records/rows.	"multiple", "single", "none"
<code>shortcutAccel</code>	Specifies the system to use, Windows or Mac.	true, false
<code>shortcutAlt</code>	Designates the Alt key	true, false
<code>shortcutCommand</code>	Designates the Command key (macOS)	true, false
<code>shortcutControl</code>	Designates the Control key (Windows)	true, false
<code>shortcutKey</code>	The letter or name of a special meaning key.	"[F1]" -> "[F15]", "[Return]", "[Enter]", "[Backspace]", "[Tab]", "[Esc]", "[Del]", "[Home]", "[End]", "[Help]", "[Page up]", "[Page down]", "[left arrow]", "[right arrow]", "[up arrow]", "[down arrow]"
<code>shortcutShift</code>	Designates the Shift key	true, false
<code>showFooters</code>	Displays or hides column footers.	true, false
<code>showGraduations</code>	Affiche/masque les graduations à côté des étiquettes.	true, false
<code>showHeaders</code>	Displays or hides column headers.	true, false
<code>showHiddenChars</code>	Displays/hides invisible characters.	true, false
<code>showHorizontalRuler</code>	Displays/hides the horizontal ruler when the document view is in Page view mode	true, false
<code>showHTMLWysiwyg</code>	Enables/disables the HTML WYSIWYG view	true, false
<code>showPageFrames</code>	Displays/hides the page frame when the document view is in Page view mode	true, false
<code>showReferences</code>	Displays all 4D expressions inserted in the 4D Write Pro document as references	true, false
<code>showSelection</code>	Keeps the selection visible within the object after it has lost the focus	true, false
<code>showVerticalRuler</code>	Displays/hides the vertical ruler when the document view is in Page view mode	true, false

<code>singleClickEdit</code>	Enables direct passage to edit mode. Description	true, false Valeurs possibles
<code>sizingX</code>	Specifies if the horizontal size of an object should be moved or resized when a user resizes the form.	"grow", "move", "fixed"
<code>sizingY</code>	Specifies if the vertical size of an object should be moved or resized when a user resizes the form.	"grow", "move", "fixed"
<code>sortable</code>	Allows sorting column data by clicking the header.	true, false
<code>spellcheck</code>	Activates the spell-check for the object	true, false
<code>splitterMode</code>	Lorsqu'un objet splitter a cette propriété, les autres objets à sa droite (splitter vertical) ou en dessous (splitter horizontal) sont poussés en même temps que le splitter, sans arrêt.	"grow", "move", "fixed"
<code>startPoint</code>	Starting point for drawing a line object (only available in JSON Grammar).	"bottomLeft", topLeft"
<code>staticColumnCount</code>	Number of columns that cannot be moved during execution.	minimum : 0
<code>step</code>	Intervalle minimum accepté entre les valeurs pendant l'utilisation. Pour les steppers numériques, cette propriété représente les secondes lorsque l'objet est associé à une valeur de type heure et représente les jours lorsqu'il est associé à une valeur de type date.	minimum: 1
<code>storeDefaultStyle</code>	Store the style tags with the text, even if no modification has been made	true, false
<code>stroke</code> (text) <code>stroke</code> (lines) <code>stroke</code> (list box)	Specifies the color of the font or line used in the object.	Any CSS value, "transparent", "automatic"
<code>strokeDashArray</code>	Describes dotted line type as a sequence of black and white points	Number array or string
<code>strokeWidth</code>	Désigne l'épaisseur d'une ligne.	An integer or 0 for smallest width on a printed form
<code>style</code>	Permet de définir l'apparence générale du bouton. Pour plus d'informations, voir Style de bouton.	"regular", "flat", "toolbar", "bevel", "roundedBevel", "gradientBevel", "texturedBevel", "office", "help", "circular", "disclosure", "roundedDisclosure", "custom"
<code>styledText</code>	Permet d'utiliser des styles spécifiques dans la zone sélectionnée.	true, false
<code>switchBackWhenReleased</code>	Affiche la première image en permanence, sauf lorsque l'utilisateur clique sur le bouton. Affiche la deuxième image jusqu'à ce que le bouton de la souris soit relâché.	true, false
<code>switchContinuously</code>	Permet à l'utilisateur de maintenir le	true, false

Propriété	Description	Valeurs possibles
	bouton de la souris enfoncé pour afficher les images en continu (c'est-à-dire sous forme d'animation).	
<code>switchWhenRollover</code>	Modifie le contenu du bouton image lorsque le curseur de la souris passe dessus. L'image initiale s'affiche lorsque le curseur quitte la zone du bouton.	true, false
t		
<code>table</code>	La table à laquelle appartient le sous-formulaire Liste (le cas échéant).	Nom de table 4D, ou ""
<code>Texte</code>	Le titre de l'objet formulaire	Tout type de texte
<code>textAlign</code>	Emplacement horizontal du texte dans la zone où il apparaît.	"automatic", "right", "center", "justify", "left"
<code>textAngle</code>	Modifies the orientation (rotation) of the text area.	0, 90, 180, 270
<code>textDecoration</code>	Sets the selected text to have a line running beneath it.	"normal", "underline"
<code>textFormat</code>	Controls the way the alphanumeric fields and variables appear when displayed or printed.	"#### ####", "(###) ### ####", "### ####", "### ## ####", "00000", custom formats
<code>textPlacement</code>	Relative location of the button title in relation to the associated icon.	"left", "top", "right", "bottom", "center"
<code>threeState</code>	Allows a check box object to accept a third state.	true, false
<code>timeFormat</code>	Controls the way times appear when displayed or printed. Must only be selected among the 4D built-in formats.	"systemShort", "systemMedium", "systemLong", "iso8601", "hh_mm_ss", "hh_mm", "hh_mm_am", "mm_ss", "HH_MM_SS", "HH_MM", "MM_SS", "blankIfNull" (can be combined with the other possible values)
<code>truncateMode</code>	Controls the display of values when list box columns are too narrow to show their full contents.	"withEllipsis", "none"
<code>type</code>	Obligatoire. Désigne le type de données de l'objet formulaire.	"text", "rectangle", "groupBox", "tab", "line", "button", "checkbox", "radio", "dropdown", "combo", "webArea", "write", "subform", "plugin", "splitter", "buttonGrid", "progress", "ruler", "spinner", "stepper", "list", "pictureButton", "picturePopup", "listbox", "input", "view"
<code>tooltip</code>	Fournit aux utilisateurs des informations supplémentaires sur un champ.	Informations supplémentaires destinées à aider l'utilisateur
<code>top</code>	Positionne un objet en haut (centré).	minimum : 0
u		
<code>urlSource</code>	Désigne l'URL chargée ou en cours de chargement par la zone Web associée.	Une URL.

<code>useLastFrameAsDisabled</code>	Permet de définir la dernière vignette comme étant celle à afficher lorsque le bouton est désactivé.	true, raise Valeurs possibles
<code>userInterface</code>	Interface de la zone 4D View Pro.	"none" (par défaut), "ribbon", "toolbar"
v		
<code>values</code>	Liste des valeurs par défaut pour les colonnes de listbox de type tableau	ex : "A","B","42"...
<code>variableCalculation</code>	Permet d'effectuer des calculs mathématiques.	"none", "minimum", "maximum", "sum", "count", "average", "standardDeviation", "variance", "sumSquare"
<code>verticalAlign</code>	Emplacement vertical du texte dans la zone où il apparaît.	"automatic", "top", "middle", "bottom"
<code>verticalLineStroke</code>	Définit la couleur des lignes verticales d'une list box (gris par défaut).	Any CSS value, "transparent", "automatic"
<code>visibility</code>	Permet de masquer l'objet dans l'environnement d'application.	"visible", "hidden", "selectedRows", "unselectedRows"
w		
<code>webEngine</code>	Permet de choisir entre deux moteurs de rendu pour la zone Web, en fonction des spécificités de l'application.	"embedded", "system"
<code>width</code>	Désigne la taille horizontale d'un objet	minimum : 0
<code>withFormulaBar</code>	Gère l'affichage d'une barre de formule avec l'interface Toolbar dans la zone 4D View Pro.	true, false
<code>wordwrap</code>	Manages the display of contents when it exceeds the width of the object.	"automatic" (à l'exception de list box), "normal", "none"
z		
<code>zoom</code>	Pourcentage de zoom pour l'affichage de la zone 4D Write Pro	numérique (minimum=0)

Action

Glissable

Contrôlez si l'utilisateur peut faire glisser l'objet et comment il peut le faire. Par défaut, aucune opération de glisser n'est autorisée.

Deux modes de glisser-déposer sont proposés dans 4D :

- Un mode personnalisé, dans lequel le glisser déclenche l'événement formulaire `Sur début glisser` dans le contexte de l'objet. Vous gérez ensuite le glisser à l'aide d'une méthode.
En mode personnalisé, le glisser-déposer est géré par le programmeur. Ce mode vous permet de mettre en place des interfaces basées sur le glisser-déposer, y compris des interfaces qui ne déplacent pas nécessairement des données mais qui peuvent effectuer tout type d'action, telle que l'ouverture de fichiers ou le lancement d'un calcul. Ce mode est basé sur un ensemble de propriétés, d'événements et de commandes spécifiques à partir du thème `Conteneur de données`.
- Un mode automatique, dans lequel 4D copie du texte ou des images directement à partir de l'objet formulaire. Il peut alors être utilisé dans la même zone 4D, entre deux zones 4D, ou entre 4D et une autre application. Par exemple, le glisser-déposer automatique vous permet de copier une valeur entre deux champs, sans programmation :



Dans ce mode, l'événement de formulaire `Sur début glisser` n'est PAS généré. Si vous souhaitez "forcer" l'utilisation du glissement personnalisé alors que le glissement automatique est activé, maintenez la touche Alt (Windows) ou Option (macOS) enfonce pendant l'action. Cette option n'est pas disponible pour les images.

Pour plus d'informations, reportez-vous à [Glisser-déposer](#) dans le manuel *Langage 4D*.

Grammaire JSON

Nom	Type de données	Valeurs possibles
dragging	Texte	"none" (par défaut), "custom", "automatic" (hors list box)

Objets pris en charge

[Zones 4D Write Pro](#) - [Zone de saisie](#) - [Liste hiérarchique](#) - [List Box](#) - [Zone de plug-in](#)

Voir aussi

[Déposable](#)

Déposable

Contrôlez si et comment l'objet peut être la destination d'une opération de glisser-déposer.

Deux modes de glisser-déposer sont proposés dans 4D :

- Un mode personnalisé, dans lequel le déposer déclenche les événements formulaire `Sur glisser` et `Sur déposer` dans le contexte de l'objet. Vous gérez ensuite le déposer à l'aide d'une méthode.
En mode personnalisé, le glisser-déposer est géré par le programmeur. Ce mode vous permet de mettre en place des interfaces basées sur le glisser-déposer, y compris des interfaces qui ne déplacent pas nécessairement des

données mais qui peuvent effectuer tout type d'action, telle que l'ouverture de fichiers ou le lancement d'un calcul. Ce mode est basé sur un ensemble de propriétés, d'événements et de commandes spécifiques à partir du thème **Conteneur de données**.

- Un mode automatique, dans lequel 4D gère automatiquement — si possible — l'insertion des données glissées de type texte ou image et déposées sur l'objet (les données sont collées dans l'objet). Les événements **Sur glisser** et **Sur déposer** ne sont pas générés. En revanche, les événements **Sur après modification** (lors du déposer) et **Sur données modifiées** (lorsque l'objet perd le focus) sont générés.

Pour plus d'informations, reportez-vous à [Glisser-déposer](#) dans le manuel *Langage 4D*.

Grammaire JSON

Nom	Type de données	Valeurs possibles
dropping	Texte	"none" (par défaut), "custom", "automatic" (hors list box)

Objets pris en charge

[Zones 4D Write Pro](#) - [Bouton](#) - [Zone de saisie](#) - [Liste hiérarchique](#) - [List Box](#) - [Zone de plug-in](#)

Voir aussi

[Glissable](#)

Exécuter méthode objet

Lorsque cette option est activée, la méthode objet est exécutée avec l'événement **Sur données modifiées au même moment** où l'utilisateur change la valeur de l'indicateur. Lorsque l'option est désactivée, la méthode est exécutée *après* la modification.

Grammaire JSON

Nom	Type de données	Valeurs possibles
continuousExecution	boolean	true, false

Objets pris en charge

[Indicateur de progression](#) - [Règle](#) - [Stepper](#)

Méthode

Référence d'une méthode attachée à l'objet. Les méthodes d'objet "gèrent" généralement l'objet pendant que le formulaire est affiché ou imprimé. Vous nappelez pas de méthode objet - 4D l'appelle automatiquement lorsqu'un événement implique l'objet auquel la méthode objet est rattachée.

Plusieurs types de références de méthode sont pris en charge :

- un chemin de fichier de méthode objet standard, c'est-à-dire qui utilise le modèle suivant : **ObjectMethods/objectName.4dm** ... où **objectName** est le **nom de l'objet**. Ce type de référence indique que le fichier de méthode se trouve à l'emplacement par défaut ("sources/forms/formName/ObjectMethods/"). Dans ce cas, 4D gère automatiquement la méthode objet lorsque des opérations sont exécutées sur l'objet formulaire (renommage, duplication, copier/coller, etc.)
- un nom de méthode projet : nom d'une méthode projet existante sans extension de fichier, c'est-à-dire : **maMéthode** Dans ce cas, 4D ne prend pas en charge automatiquement les opérations objet.
- un chemin d'accès du fichier de méthode personnalisé comprenant l'extension **.4dm**, par exemple :

`.../CustomMethods/myMethod.4dm` Vous pouvez également utiliser un filesystem :
`/RESOURCES/Buttons/b0K.4dm` Dans ce cas, 4D ne prend pas en charge automatiquement les opérations sur les objets.

Grammaire JSON

Nom	Type de données	Valeurs possibles
method	Texte	Chemin de fichier standard ou personnalisé de la méthode objet ou nom de la méthode projet

Objets pris en charge

Zone 4D View Pro - Zone 4D Write Pro - Bouton - Grille de boutons - Case à cocher - Combo Box - Liste déroulante - Formulaires - Liste hiérarchique - Zone de saisie - List Box - Colonne List Box - Bouton image - Pop up menu image - Zone de plug-in - Indicateur de progression - Bouton radio - Règle - Spinner - Splitter - Stepper - Sous-formulaire - Onglet - Zone Web

Lignes déplaçables

List box de type tableau

Autorise le déplacement des lignes pendant l'exécution. Cette option est sélectionnée par défaut. Il n'est pas disponible pour les [list box de type sélection](#) ni pour les [list box en mode hiérarchique](#).

Grammaire JSON

Nom	Type de données	Valeurs possibles
movableRows	boolean	true, false

Objets pris en charge

List Box

Multi-sélectionnable

Allows the selection of multiple records/options in a [hierarchical list](#).

Grammaire JSON

Nom	Type de données	Valeurs possibles
selectionMode	Texte	"multiple", "single", "none"

Objets pris en charge

Liste hiérarchique

Triable

Permet de trier les données de colonne en cliquant sur un en-tête de [Listbox](#). Cette option est sélectionnée par défaut. Les tableaux de types d'image (colonnes) ne peuvent pas être triés à l'aide de cette fonction.

Dans les list box basées sur une sélection d'enregistrements, la fonction de tri standard est disponible uniquement :

- Lorsque la source de données est *Sélection courante*,
- Avec des colonnes associées à des champs (de type Alpha, Numérique, Date, Heure ou Booléen).

Dans d'autres cas (list box basées sur des sélections nommées, colonnes associées à des expressions), la fonction de tri standard n'est pas disponible. Un tri de list box standard modifie l'ordre de la sélection courante dans la base de données. Cependant, les enregistrements en surbrillance et l'enregistrement courant ne sont pas modifiés. Un tri standard synchronise toutes les colonnes de la list box, y compris les colonnes calculées.

Grammaire JSON

Nom	Type de données	Valeurs possibles
sortable	boolean	true, false

Objets pris en charge

[List Box](#)

Action standard

Activités typiques à réaliser par les objets actifs (par exemple, permettre à l'utilisateur d'accepter, d'annuler ou de supprimer des enregistrements, de se déplacer entre les enregistrements ou de page en page dans un formulaire multi-pages, etc.) ont été prédéfinies par 4D comme actions standard. Elles sont décrites en détail dans la section [Actions standard](#) du *manuel de développement*.

Vous pouvez associer à la fois une action standard et la méthode projet d'un objet. Dans ce cas, l'action standard est généralement exécutée après la méthode et 4D utilise cette action pour activer/désactiver l'objet en fonction du contexte courant. Lorsqu'un objet est désactivé, la méthode projet associée ne peut être exécutée.

Vous pouvez également définir cette propriété à l'aide de la commande `OBJECT SET ACTION`.

Grammaire JSON

Nom	Type de données	Valeurs possibles
action	string	Le nom d'une action standard valide .

Objets pris en charge

[Bouton](#) - [Grille de boutons](#) - [Check Box](#) - [Liste déroulante](#) - [List Box](#) - [Bouton image](#) - [Pop-up Menu image](#) - [Onglet](#)

Animation

Recommencer la séquence

Les images sont affichées en boucle continue. Lorsque l'utilisateur atteint la dernière image et clique à nouveau, la première image apparaît, et ainsi de suite.

Grammaire JSON

Nom	Type de données	Valeurs possibles
loopBackToFirstFrame	boolean	true, false

Objets pris en charge

[Bouton image](#)

Retour sur relâchement du clic

Affiche la première image en permanence, sauf lorsque l'utilisateur clique sur le bouton. Affiche la deuxième image jusqu'à ce que le bouton de la souris soit relâché. Ce mode vous permet de créer un bouton d'action avec une image différente pour chaque état (inactif et cliqué). Vous pouvez utiliser ce mode pour créer un effet 3D ou afficher n'importe quelle image illustrant l'action du bouton.

Grammaire JSON

Nom	Type de données	Valeurs possibles
switchBackWhenReleased	boolean	true, false

Objets pris en charge

[Bouton image](#)

Défilement continu sur clic

Permet à l'utilisateur de maintenir le bouton de la souris enfoncé pour afficher les images en continu (c'est-à-dire sous forme d'animation). Lorsque l'utilisateur atteint la dernière image, l'objet ne revient pas à la première image.

Grammaire JSON

Nom	Type de données	Valeurs possibles
switchContinuously	boolean	true, false

Objets pris en charge

[Bouton image](#)

Défilement tous les n ticks

Permet de parcourir le contenu du bouton d'image à la vitesse spécifiée (en graduations). Dans ce mode, toutes les autres options sont ignorées.

Grammaire JSON

Nom	Type de données	Valeurs possibles
frameDelay	entier	minimum : 0

Objets pris en charge

[Bouton image](#)

Bascule sur passage du curseur

Modifie le contenu du bouton image lorsque le curseur de la souris passe dessus. L'image initiale s'affiche lorsque le curseur quitte la zone du bouton.

Grammaire JSON

Nom	Type de données	Valeurs possibles
switchWhenRollover	boolean	true, false

Objets pris en charge

[Bouton image](#)

Dernière imagette si désactivé

Permet de définir la dernière vignette comme étant celle à afficher lorsque le bouton est désactivé. La vignette utilisée lorsque le bouton est désactivé est traitée séparément par 4D : lorsque vous combinez cette option avec "Basculer en continu" et "Revenir en boucle à la première image", la dernière image est exclue de la séquence associée au bouton et n'apparaît que lorsqu'elle est désactivée.

Grammaire JSON

Nom	Type de données	Valeurs possibles
useLastFrameAsDisabled	boolean	true, false

Objets pris en charge

[Bouton image](#)

Apparence

Bouton par défaut

La propriété de bouton par défaut désigne le bouton qui obtient le focus initial à l'exécution lorsqu'aucun bouton du formulaire ne possède la propriété [Focusable](#).

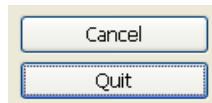
Il ne peut y avoir qu'un seul bouton par défaut par page de formulaire.

De plus, sous macOS, la propriété du bouton par défaut modifie l'apparence du bouton afin d'indiquer un «choix recommandé» à l'utilisateur. Le bouton par défaut peut être différent du bouton sélectionné. Les boutons par défaut ont une apparence bleue spécifique sur macOS :



Le bouton doit avoir une hauteur standard pour obtenir l'apparence du bouton par défaut.

Sous Windows, le concept de "choix recommandé" n'est pas pris en charge: seul le bouton focalisé a une apparence différente à l'exécution. Cependant, dans l'éditeur de formulaires 4D, le bouton par défaut est représenté par un contour bleu :



Grammaire JSON

Nom	Type de données	Valeurs possibles
defaultButton	booléen	true, false

Objets pris en charge

[Bouton - Bouton plat](#)

Cacher rectangle de focus

A l'exécution, un champ ou toute zone saisissable est délimité par un rectangle de sélection lorsqu'il a le focus (via la touche Tab ou un simple clic). Vous pouvez masquer ce rectangle en activant cette propriété. Masquer le rectangle de focus peut être utile dans le cas d'interfaces spécifiques.

Grammaire JSON

Nom	Type de données	Valeurs possibles
hideFocusRing	boolean	true, false

Objets pris en charge

[Zones 4D Write Pro](#) - [Liste hiérarchique](#) - [Zone de saisie](#) - [List Box](#) - [Sous-formulaire](#)

Cacher surlignage sélection

List boxes de type sélection

Cette propriété est utilisée pour désactiver la mise en évidence de la sélection dans les list box.

Lorsque cette option est activée, la surbrillance de la sélection n'est plus visible pour les sélections effectuées dans les list box. Les sélections elles-mêmes sont toujours valides et fonctionnent exactement de la même manière que précédemment; cependant, ils ne sont plus représentés graphiquement à l'écran et vous devrez [définir leur apparence par programmation](#).

Par défaut, cette option n'est pas activée.

Grammaire JSON

Nom	Type de données	Valeurs possibles
hideSystemHighlight	boolean	true, false

Objets pris en charge

List Box

Barre de défilement horizontale

Un outil d'interface permettant à l'utilisateur de déplacer la zone de visualisation vers la gauche ou la droite.

Valeurs disponibles :

Liste de propriétés	Valeur JSON	Description
Oui	"visible"	La barre de défilement est toujours visible, même lorsqu'elle n'est pas nécessaire (en d'autres termes, lorsque la taille du contenu de l'objet est inférieure à celle du cadre).
Non	"hidden"	La barre de défilement n'est jamais visible
Automatique	"automatic"	La barre de défilement apparaît automatiquement chaque fois que nécessaire et l'utilisateur peut saisir du texte plus grand que la largeur de l'objet

Les objets image peuvent avoir des barres de défilement lorsque le format d'affichage de l'image est défini sur "Tronqué (non centré)"

Grammaire JSON

Nom	Type de données	Valeurs possibles
scrollbarHorizontal	Texte	"visible", "hidden", "automatic"

Objets pris en charge

[Liste hiérarchique](#) - [Sous-formulaire](#) - [List Box](#) - [Zone de saisie](#) - [Zone 4D Write Pro](#)

Voir aussi

[Barre de défilement verticale](#)

Resolution

Définit la résolution d'écran pour le contenu de la zone 4D Write Pro. Par défaut, elle est définie sur 72 dpi (macOS), qui est la résolution standard des formulaires 4D sur toutes les plateformes. La définition de cette propriété sur 96 dpi définira un rendu Windows/Web sur les plateformes macOS et Windows. La définition de cette propriété sur automatique signifie que le rendu du document sera différent entre les plates-formes macOS et Windows.

Grammaire JSON

Nom	Type de données	Valeurs possibles
dpi	number	0=automatic, 72, 96

Objets pris en charge

[Zone 4D Write Pro](#)

Afficher l'arrière-plan

Affiche/masque les images d'arrière-plan et la couleur d'arrière-plan.

Grammaire JSON

Nom	Type de données	Valeurs possibles
showBackground	booléen	true (par défaut), false

Objets pris en charge

[Zone 4D Write Pro](#)

Afficher les pieds de page

Affiche/masque les pieds de page lorsque le [mode d'affichage de la page](#) est défini sur "Page".

Grammaire JSON

Nom	Type de données	Valeurs possibles
showFooters	booléen	true (par défaut), false

Objets pris en charge

[Zone 4D Write Pro](#)

Afficher la barre de formule

Lorsqu'elle est activée, la barre de formule est visible sous l'interface de la barre d'outils dans la zone 4D View Pro. Si elle n'est pas sélectionnée, la barre de formule est masquée.

Cette propriété est disponible uniquement pour l'interface de la [barre d'outils](#).

Grammaire JSON

Nom	Type de données	Valeurs possibles
withFormulaBar	booléen	true (par défaut), false

Objets pris en charge

[Zone 4D View Pro](#)

Montrer les entêtes

Affiche/masque les en-têtes de la page lorsque le [mode d'affichage de la page](#) est défini sur "Page".

Grammaire JSON

Nom	Type de données	Valeurs possibles
showHeaders	booléen	true (par défaut), false

Objets pris en charge

[Zone 4D Write Pro](#)

Montrer les caractères cachés

Affiche/masque les caractères visibles

Grammaire JSON

Nom	Type de données	Valeurs possibles
showHiddenChars	booléen	true (par défaut), false

Objets pris en charge

[Zone 4D Write Pro](#)

Montrer la règle horizontale

Affiche/masque la règle horizontale lorsque la vue du document est en mode [Page](#).

Grammaire JSON

Nom	Type de données	Valeurs possibles
showHorizontalRuler	booléen	true (par défaut), false

Objets pris en charge

[Zone 4D Write Pro](#)

Montrer HTML WYSIWIG

Active/désactive la vue HTML WYSIWYG, dans laquelle tous les attributs avancés de 4D Write Pro qui ne sont pas compatibles avec tous les navigateurs sont supprimés.

Grammaire JSON

Nom	Type de données	Valeurs possibles
showHTMLWysiwyg	booléen	true, false (par défaut)

Objets pris en charge

[Zone 4D Write Pro](#)

Afficher le cadre de la page

Affiche/masque le cadre de la page lorsque le [mode d'affichage de la page](#) est défini sur "Page".

Grammaire JSON

Nom	Type de données	Valeurs possibles
showPageFrames	booléen	true, false

Objets pris en charge

[Zone 4D Write Pro](#)

Afficher les références

Affiche toutes les expressions 4D insérées dans le document 4D Write Pro comme *références*. Lorsque cette option est désactivée, les expressions 4D sont affichées sous forme de *valeurs*. Par défaut, lorsque vous insérez un champ ou une expression 4D, 4D Write Pro calcule et affiche sa valeur actuelle. Sélectionnez cette propriété si vous souhaitez savoir quel champ ou quelle expression est affiché(e). Les références de champ ou d'expression apparaissent alors dans votre document, sur fond gris.

Par exemple, vous avez inséré la date courante avec un format, la date s'affiche :

July 11, 2016

Lorsque la propriété Afficher les références est activée, la référence s'affiche :

String(Current date;Internal date long)

Les expressions 4D peuvent être insérées à l'aide de la commande [ST INSERT EXPRESSION](#).

Grammaire JSON

Nom	Type de données	Valeurs possibles
showReferences	booléen	true, false (par défaut)

Objets pris en charge

[Zone 4D Write Pro](#)

Afficher règle verticale

Affiche/masque la règle verticale lorsque la vue du document est en mode [Page](#).

Grammaire JSON

Nom	Type de données	Valeurs possibles
showVerticalRuler	booléen	true (par défaut), false

Objets pris en charge

[Zone 4D Write Pro](#)

Onglets

Vous pouvez définir la direction des onglets dans vos formulaires. Cette propriété est disponible sur toutes les plateformes mais ne peut être affichée que sous macOS. Vous pouvez choisir de placer les onglets en haut (standard) ou en bas.

Lorsque des onglets sont affichés avec une direction personnalisée sous Windows, ils retournent automatiquement à la direction standard (en haut).

Grammaire JSON

Nom	Type de données	Valeurs possibles
labelsPlacement	booléen	"top", "bottom"

Objets pris en charge

[Onglets](#)

Interface utilisateur

Vous pouvez ajouter une interface aux zones 4D View Pro pour permettre aux utilisateurs finaux d'effectuer des modifications de base et des manipulations de données. 4D View Pro propose deux interfaces en option, le ruban et la barre d'outils.

Grammaire JSON

Nom	Type de données	Valeurs possibles
userInterface	texte	"none" (par défaut), "ribbon", "toolbar"

Objets pris en charge

[Zone 4D View Pro](#)

Voir aussi

[guide de référence 4D View Pro](#)

Barre de défilement verticale

Un outil d'interface permettant à l'utilisateur de déplacer la zone de visualisation de haut en bas.

Valeurs disponibles :

Liste de propriétés	Valeur JSON	Description
Oui	"visible"	La barre de défilement est toujours visible, même lorsqu'elle n'est pas nécessaire (en d'autres termes, lorsque la taille du contenu de l'objet est inférieure à celle du cadre).
Non	"hidden"	La barre de défilement n'est jamais visible
Automatique	"automatic"	La barre de défilement apparaît automatiquement chaque fois que nécessaire (en d'autres termes, lorsque la taille du contenu de l'objet est supérieure à celle du cadre)

Les objets image peuvent avoir des barres de défilement lorsque le format d'affichage de l'image est défini sur "Tronqué (non centré)"

Si un objet de saisie de texte n'a pas de barre de défilement, l'utilisateur peut faire défiler les informations à l'aide des flèches du clavier.

Grammaire JSON

Nom	Type de données	Valeurs possibles
scrollbarVertical	Texte	"visible", "hidden", "automatic"

Objets pris en charge

[Liste hiérarchique](#) - [Sous-formulaire](#) - [List Box](#) - [Zone de saisie](#) - [Zone 4D Write Pro](#)

Voir aussi

[Barre de défilement horizontale](#)

Mode d'affichage

Définit le mode d'affichage du document 4D Write Pro dans la zone de formulaire. Trois valeurs sont disponibles :

- Page : le mode d'affichage le plus complet, qui comprend les contours de page, l'orientation, les marges, les sauts de page, les en-têtes et pieds de page, etc.
- Brouillon : mode brouillon avec propriétés de base du document
- Embedded : mode d'affichage adapté aux zones intégrées; il n'affiche pas les marges, les pieds de page, les en-têtes, les cadres, etc. Ce mode peut également être utilisé pour produire un affichage de type Web (si vous sélectionnez également la [résolution de 96 dpi](#) et les propriétés [Afficher HTML WYSIWYG](#)).

La propriété Mode d'affichage est utilisée uniquement pour le rendu à l'écran. Concernant les paramètres d'impression, des règles de rendu spécifiques sont automatiquement utilisées.

Grammaire JSON

Nom	Type de données	Valeurs possibles
layoutMode	texte	"page", "draft", "embedded"

Objets pris en charge

[Zone 4D Write Pro](#)

Zoom

Définit le pourcentage de zoom pour l'affichage du contenu de la zone 4D Write Pro.

Grammaire JSON

Nom	Type de données	Valeurs possibles
zoom	numérique	minimum = 0

Objets pris en charge

[Zone 4D Write Pro](#)

Fond et bordure

Couleur de fond alternée

Permet de définir une couleur d'arrière-plan différente pour les lignes / colonnes impaires dans une list box. Par défaut, *Automatique* est sélectionné : la colonne utilise la couleur de fond alternative définie au niveau de la list box.

Grammaire JSON

Nom	Type de données	Valeurs possibles
alternateFill	string	toutes les valeurs css; "transparent"; "automatic"; "automaticAlternate"

Objets pris en charge

[List Box - Colonne List Box](#)

Couleur de fond / Couleur de remplissage

Définit la couleur de fond d'un objet.

Dans le cas d'une list box, par défaut *Automatique* est sélectionné : la colonne utilise la couleur de fond définie au niveau de la list box.

Grammaire JSON

Nom	Type de données	Valeurs possibles
border-style	string	une valeur css; "transparent"; "automatic"

Objets pris en charge

[Liste Hiérarchique](#) - [List Box](#) - [Colonne List Box](#) - [Pied List Box](#) - [Ovale](#) - [Rectangle](#) - [Zone de texte](#)

Voir aussi

[Transparent](#)

Expression couleur de fond

`List box de type collection et de type sélection d'entité`

Une expression ou une variable (les variables de tableau ne peuvent pas être utilisées) pour appliquer une couleur d'arrière-plan personnalisée à chaque ligne de la list box. L'expression ou la variable sera évaluée pour chaque ligne affichée et doit retourner une valeur de couleur RGB. Pour plus d'informations, reportez-vous à la description de la commande `OBJECT SET RGB COLORS` dans le *manuel de langage 4D*.

Vous pouvez également définir cette propriété à l'aide de la commande `LISTBOX SET PROPERTY` avec la constante `lk background color expression`.

Avec les list box de type collection ou sélection d'entité, cette propriété peut également être définie à l'aide d'une [Meta Info Expression](#).

Grammaire JSON

Nom	Type de données	Valeurs possibles
rowFillSource	string	Une expression retournant une valeur de couleur RGB

Objets pris en charge

[List Box](#) - [Colonne List Box](#)

Style de la bordure

Permet de définir un style standard pour la bordure de l'objet.

Grammaire JSON

Nom	Type de données	Valeurs possibles
borderStyle	Texte	"system", "none", "solid", "dotted", "raised", "sunken", "double"

Objets pris en charge

[Zone 4D View Pro](#) - [Zone 4D Write Pro](#) - [Boutons](#) - [Grille de boutons](#) - [Case à cocher](#) - [Zone de saisie](#) - [List Box](#) - [Bouton image](#) - [Pop up menu image](#) - [Zone de plug-in](#) - [Indicateur de progression](#) - [Règle](#) - [Spinner](#) - [Stepper](#) - [Sous-formulaire](#) - [Onglet](#) - [Zone Web](#)

Dotted Line Type

Décrit le type de ligne en pointillé comme une séquence de points noirs et blancs.

Grammaire JSON

Nom	Type de données	Valeurs possibles
strokeDashArray	number array or string	Ex : Ex : Ex : Ex : "6 1" ou [6,1] pour une séquence de points noirs et un point blanc

Objets pris en charge

[Rectangle](#) - [Ovale](#) - [Ligne](#)

Masquer lignes vides finales

Contrôle l'affichage des lignes vides supplémentaires ajoutées au bas d'un objet list box. Par défaut, 4D ajoute ces lignes supplémentaires pour remplir la zone vide :

Days	Values
Monday	5
Tuesday	6
Wednesday	41
Thursday	66
Friday	12
Saturday	2
Sunday	88

Extra blank rows

Vous pouvez supprimer ces lignes vides en sélectionnant cette option. Le bas de l'objet list box est alors laissé vide :

Days	Values
Monday	5
Tuesday	6
Wednesday	41
Thursday	66
Friday	12
Saturday	2
Sunday	88

Grammaire JSON

Nom	Type de données	Valeurs possibles
hideExtraBlankRows	boolean	true, false

Objets pris en charge

List Box

Line Color

Désigne la couleur des lignes de l'objet. La couleur peut être spécifiée par :

- un nom de couleur - comme "red"
- une valeur HEX - comme "# ff0000"
- une valeur RVB - comme "rgb (255,0,0)"

Vous pouvez également définir cette propriété à l'aide de la commande [OBJECT SET RGB COLORS](#).

Grammaire JSON

Nom	Type de données	Valeurs possibles
stroke	string	une valeur css; "transparent"; "automatic"

Cette propriété est également disponible pour les objets à base de texte, auquel cas elle désigne à la fois la couleur de la police et les lignes de l'objet, voir [Couleur de la police](#).

Objets pris en charge

Line Width

Désigne l'épaisseur d'une ligne.

Grammaire JSON

Nom	Type de données	Valeurs possibles
strokeWidth	number	0 pour la plus petite largeur dans un formulaire imprimé, ou toute valeur d'entier < 20

Objets pris en charge

Ligne - Ovale - Rectangle

Tableau couleurs de fond

List box de type tableau

Le nom d'un tableau pour appliquer une couleur d'arrière-plan personnalisée à chaque ligne ou colonne de la list box.

Le nom d'un tableau Entier long doit être saisi. Chaque élément de ce tableau correspond à une ligne de la zone de list box (si elle est appliquée à la liste box) ou à une cellule de la colonne (si elle est appliquée à une colonne), le tableau doit donc avoir la même taille que le tableau associé à la colonne. Vous pouvez utiliser les constantes du thème [SET RGB COLORS](#). Si vous souhaitez que la cellule hérite de la couleur d'arrière-plan définie au niveau supérieur, passez la valeur -255 à l'élément de tableau correspondant.

Par exemple, considérons une list box où les lignes ont une couleur alternée gris/gris clair, définie dans les propriétés de la list box. Un tableau de couleurs d'arrière-plan a également été défini pour la list box afin de changer en orange clair la couleur des lignes où au moins une valeur est négative :

```
<>_BgnColors{$i}:=0x00FFD0B0 // orange  
<>_BgnColors{$i}:=-255 // valeur par défaut
```

Header1	Header2	Header3
21483	9031	27290
24151	21990	-923
21351	2982	18009
8089	12898	20941
13001	-802	22059
4321	16826	11303
24082	26214	22380
16680	23651	20403
-2678	24818	29896
25639	2691	9687
28794	26941	21486
26083	21092	13476
27928	4092	15441
19987	28211	21191
7996	6300	4089
-1063	13388	23683
13008	7470	19897
5388	26918	13547
28559	27007	8365
28454	22646	13824

Vous souhaitez ensuite colorer les cellules avec des valeurs négatives en orange foncé. Pour ce faire, définissez un tableau de couleurs d'arrière-plan pour chaque colonne, par exemple <>_BgnColor_1, <>_BgnColor_2 et <>_BgnColor_3. Les valeurs de ces tableaux ont la priorité sur celles définies dans les propriétés de list box ainsi que sur celles du tableau de couleurs d'arrière-plan général :

```
<>_BgnColorsCol_3{2}:=0x00FF8000 // orange foncé
<>_BgnColorsCol_2{5}:=0x00FF8000
<>_BgnColorsCol_1{9}:=0x00FF8000
<>_BgnColorsCol_1{16}:=0x00FF8000
```

Header1	Header2	Header3
21483	9031	27290
24151	21990	-923
21351	2982	18009
8089	12898	20941
13001	-802	22059
4321	16826	11303
24082	26214	22380
16680	23651	20403
-2678	24818	29896
25639	2691	9687
28794	26941	21486
26083	21092	13476
27928	4092	15441
19987	28211	21191
7996	6300	4089
-1063	13388	23683
13008	7470	19897
5388	26918	13547
28559	27007	8365
28454	22646	13824

Vous pouvez obtenir le même résultat en utilisant les commandes `LISTBOX SET ROW FONT STYLE` et `LISTBOX SET ROW COLOR`. Elles ont l'avantage de vous permettre d'éviter d'avoir à prédefinir des tableaux de style/couleur pour les colonnes : ils sont plutôt créés dynamiquement par les commandes.

Nom	Type de données	Valeurs possibles
rowFillSource	string	Nom d'un tableau entier long.

Objets pris en charge

[List Box - Colonne List Box](#)

Transparent

Définit l'arrière-plan de la list box sur "Transparent". Lorsqu'elle est définie, toute [autre couleur d'arrière-plan](#) ou [couleur d'arrière-plan](#) définie pour la colonne est ignorée.

Grammaire JSON

Nom	Type de données	Valeurs possibles
border-style	Texte	"transparent"

Objets pris en charge

[List Box](#)

Voir aussi

[Couleur de fond / Couleur de remplissage](#)

Coordonnées et dimensions

Hauteur de ligne automatique

Cette propriété n'est disponible que pour les list box de type tableau, non hiérarchiques. Par défaut, cette option n'est pas sélectionnée.

Lorsqu'elle est utilisée, la hauteur de chaque ligne de la colonne est automatiquement calculée par 4D, et le contenu de la colonne est pris en compte. A noter que seules les colonnes avec l'option sélectionnée seront prises en compte pour calculer la hauteur de ligne.

Lors du redimensionnement du formulaire, si la propriété de [dimensionnement horizontal "Agrandir"](#) a été affectée à la list box, la colonne la plus à droite sera agrandie, allant au-delà de sa largeur maximale, si nécessaire.

Lorsque cette propriété est activée, la hauteur de chaque ligne est automatiquement calculée afin d'ajuster entièrement le contenu de la cellule ajusté sans être tronqué (sauf si l'option [Wordwrap](#) est désactivée).

- Le calcul de la hauteur de ligne prend en compte :
 - tout type de contenu (texte, numérique, dates, heures, images (le calcul dépend du format de l'image), objets),
 - tout types de contrôle (zones de saisie, cases à cocher, listes, listes déroulantes),
 - polices, styles de polices et tailles de polices,
 - l'option [Retour à la ligne](#) : si elle est désactivée, la hauteur est basée sur le nombre de paragraphes (les lignes sont tronquées); si elle est activée, la hauteur est basée sur le nombre de lignes (non tronquées).
- Le calcul de la hauteur de ligne ne tient pas compte de :
 - du contenu de colonne masqué
 - des propriétés du tableau [Hauteur de ligne](#) et [Tableau hauteur de lignes](#) (le cas échéant) définies dans la liste de propriété ou par programmation.

Etant donné qu'elle nécessite des calculs supplémentaires lors de l'exécution, l'option "hauteur de ligne automatique" peut avoir une incidence sur la fluidité du défilement de votre list box, en particulier lorsqu'elle contient un grand nombre de lignes.

Grammaire JSON

Nom	Type de données	Valeurs possibles
rowHeightAuto	boolean	true, false

Objets pris en charge

[Colonne de list box](#)

Bas

Coordonnées inférieures de l'objet dans le formulaire.

Grammaire JSON

Nom	Type de données	Valeurs possibles
bottom	number	minimum : 0

Objets pris en charge

Zone 4D View Pro - Zone 4D Write Pro - Bouton - Grille de boutons - Case à cocher - Combo Box - Liste déroulante - Group Box - Liste hiérarchique - Zone de saisie - List Box - Ligne - Colonne List Box - Ovale - Bouton image - Pop up menu image - Zone de plug-in - Indicateur de progression - Bouton radio - Rectangle - Règle - Spinner - Splitter - Image statique Stepper - Sous-formulaire - Onglet - Zone de texte - Zone Web

Gauche

Coordonnées de gauche de l'objet dans le formulaire.

Grammaire JSON

Nom	Type de données	Valeurs possibles
left	number	minimum : 0

Objets pris en charge

Zone 4D View Pro - Zone 4D Write Pro - Bouton - Grille de boutons - Case à cocher - Combo Box - Liste déroulante - Group Box - Liste hiérarchique - Zone de saisie - List Box - Ligne - Colonne List Box - Ovale - Bouton image - Pop up menu image - Zone de plug-in - Indicateur de progression - Bouton radio - Rectangle - Règle - Spinner - Splitter - Image statique Stepper - Sous-formulaire - Onglet - Zone de texte - Zone Web

Droite

Coordonnées de droite de l'objet dans le formulaire.

Grammaire JSON

Nom	Type de données	Valeurs possibles
right	number	minimum : 0

Objets pris en charge

Zone 4D View Pro - Zone 4D Write Pro - Bouton - Grille de boutons - Case à cocher - Combo Box - Liste déroulante - Group Box - Liste hiérarchique - Zone de saisie - List Box - Ligne - Colonne List Box - Ovale - Bouton image - Pop up menu image - Zone de plug-in - Indicateur de progression - Bouton radio - Rectangle - Règle - Spinner - Splitter - Image statique Stepper - Sous-formulaire - Onglet - Zone de texte - Zone Web

Haut

Coordonnées supérieures de l'objet dans le formulaire.

Grammaire JSON

Nom	Type de données	Valeurs possibles
top	number	minimum : 0

Objets pris en charge

Zone 4D View Pro - Zone 4D Write Pro - Bouton - Grille de boutons - Case à cocher - Combo Box - Liste déroulante - Group Box - Liste hiérarchique - Zone de saisie - List Box - Ligne - Colonne List Box - Ovale - Bouton image - Pop up menu image - Zone de plug-in - Indicateur de progression - Bouton radio - Règle - Rectangle - Spinner - Splitter - Image statique Stepper - Sous-formulaire - Onglet - Zone de texte - Zone Web

Rayon d'arrondi

Définit l'arrondi des coins (en pixels) des objets de type [rectangle](#). Par défaut, la valeur du rayon des rectangles est de 0 pixel. Vous pouvez modifier cette propriété pour dessiner des rectangles arrondis avec des formes personnalisées :



La valeur minimale est 0, dans ce cas un rectangle standard non arrondi est dessiné. La valeur maximale dépend de la taille du rectangle (elle ne peut pas dépasser la moitié de la taille du côté le plus court du rectangle) et est calculée dynamiquement.

Vous pouvez également définir cette propriété à l'aide des commandes [OBJECT Get corner radius](#) et [OBJECT SET CORNER RADIUS](#).

Grammaire JSON

Nom	Type de données	Valeurs possibles
borderRadius	entier	minimum : 0

Objets pris en charge

Rectangle

Hauteur

Cette propriété désigne la taille verticale d'un objet.

Certains objets peuvent avoir une hauteur prédéfinie qui ne peut pas être modifiée.

Grammaire JSON

Nom	Type de données	Valeurs possibles
height	number	minimum : 0

Objets pris en charge

Zone 4D View Pro - Zone 4D Write Pro - Bouton - Grille de boutons - Case à cocher - Combo Box - Liste déroulante - Group Box - Liste hiérarchique - Zone de saisie - List Box - Ligne - Colonne List Box - Ovale - Bouton image - Pop up menu image - Zone de plug-in - Indicateur de progression - Bouton radio - Règle - Rectangle - Spinner - Splitter - Image statique Stepper - Sous-formulaire - Onglet - Zone de texte - Zone Web

Largeur

Cette propriété désigne la taille horizontale d'un objet.

- Certains objets peuvent avoir une hauteur prédéfinie qui ne peut pas être modifiée.
- Si la propriété **Resizable** est utilisée pour une [colonne de list box](#), l'utilisateur peut également redimensionner manuellement la colonne.
- Lors du redimensionnement du formulaire, si la propriété de [dimensionnement horizontal "Agrandir"](#) a été affectée à la list box, la colonne la plus à droite sera agrandie, allant au-delà de sa largeur maximale, si nécessaire.

Grammaire JSON

Nom	Type de données	Valeurs possibles
width	number	minimum : 0

Objets pris en charge

[Zone 4D View Pro](#) - [Zone 4D Write Pro](#) - [Bouton](#) - [Grille de boutons](#) - [Case à cocher](#) - [Combo Box](#) - [Liste déroulante](#) - [Group Box](#) - [Liste hiérarchique](#) - [Zone de saisie](#) - [List Box](#) - [Ligne](#) - [Colonne List Box](#) - [Ovale](#) - [Bouton image](#) - [Pop up menu image](#) - [Zone de plug-in](#) - [Indicateur de progression](#) - [Bouton radio](#) - [Règle](#) - [Rectangle](#) - [Spinner](#) - [Splitter](#) - [Image statique Stepper](#) - [Sous-formulaire](#) - [Onglet](#) - [Zone de texte](#) - [Zone Web](#)

Largeur maxi

La largeur maximale de la colonne (en pixels). La largeur de la colonne ne peut pas être augmentée au-delà de cette valeur lors du redimensionnement de la colonne ou du formulaire.

Lors du redimensionnement du formulaire, si la propriété de [dimensionnement horizontal "Agrandir"](#) a été affectée à la list box, la colonne la plus à droite sera agrandie, allant au-delà de sa largeur maximale, si nécessaire.

Grammaire JSON

Nom	Type de données	Valeurs possibles
maxWidth	number	minimum : 0

Objets pris en charge

[Colonne de list box](#)

Largeur mini

La largeur minimale de la colonne (en pixels). La largeur de la colonne ne peut pas être réduite en dessous de cette valeur lors du redimensionnement de la colonne ou du formulaire.

Lors du redimensionnement du formulaire, si la propriété de [dimensionnement horizontal "Agrandir"](#) a été affectée à la list box, la colonne la plus à droite sera agrandie, allant au-delà de sa largeur maximale, si nécessaire.

Grammaire JSON

Nom	Type de données	Valeurs possibles
minWidth	number	minimum : 0

Objets pris en charge

[Colonne de list box](#)

Hauteur des lignes

Définit la hauteur des lignes de list box (hors en-têtes et pieds de page). Par défaut, la hauteur de ligne est définie en fonction de la plate-forme et de la taille de la police.

Grammaire JSON

Nom	Type de données	Valeurs possibles
rowHeight	string	valeur css dans l'unité "em" ou "px" (par défaut)

Objets pris en charge

[List Box](#)

Voir aussi

[Tableau hauteurs des lignes](#)

Tableau hauteurs des lignes

Cette propriété est utilisée pour indiquer le nom d'un tableau de hauteur de ligne que vous souhaitez associer à la list box. Un tableau de hauteur de ligne doit être de type numérique (entier long par défaut).

Lorsqu'un tableau de hauteur de ligne est défini, chacun de ses éléments dont la valeur est différente de 0 (zéro) est pris en compte pour déterminer la hauteur de la ligne correspondante dans la list box, en fonction de l'unité hauteur de ligne courante.

Par exemple, vous pouvez écrire :

```
ARRAY LONGINT(RowHeights;20)
RowHeights{5}:=3
```

En supposant que l'unité des lignes soit «lignes», alors la cinquième ligne de la list box aura une hauteur de trois lignes, tandis que chaque autre ligne conservera sa hauteur par défaut.

- La propriété Row Height Array n'est pas prise en compte pour les list box hiérarchiques.
- Pour les list box de type tableau, cette propriété n'est disponible que si l'option [Hauteur de ligne automatique](#) n'est pas sélectionnée.

Grammaire JSON

Nom	Type de données	Valeurs possibles
rowHeightSource	string	Nom d'une variable tableau 4D.

Objets pris en charge

[List Box](#)

Voir aussi

Hauteur des lignes

Découpage

Columns

Définit le nombre de colonnes dans un tableau d'imagettes.

Grammaire JSON

Nom	Type de données	Valeurs possibles
columnCount	entier	minimum: 1

Objets pris en charge

[Bouton image](#) - [Grille de boutons](#) - [Pop-up Menu image](#)

Rows

Définit le nombre de lignes dans un tableau d'imagettes.

Grammaire JSON

Nom	Type de données	Valeurs possibles
rowCount	entier	minimum: 1

Objets pris en charge

[Bouton image](#) - [Grille de boutons](#) - [Pop-up Menu image](#)

Source de données

Insertion automatique

Lorsque cette option est sélectionnée, si un utilisateur saisit une valeur introuvable dans la liste associée à l'objet, cette valeur est automatiquement ajoutée à la liste stockée en mémoire.

Lorsque l'option d'insertion automatique n'est pas définie (par défaut), la valeur saisie est stockée dans l'objet formulaire mais pas dans la liste en mémoire.

Cette propriété est prise en charge par :

- [Combo box](#) and [list box column](#) form objects associated to a choice list.
- [Combo box](#) form objects whose associated list is filled by their array or object datasource.

Par exemple, pour une liste de choix contenant "France, Allemagne, Italie" associée à une combo box "Pays" : si la propriété d'insertion automatique est définie et qu'un utilisateur saisit "Espagne", la valeur "Espagne" est alors automatiquement ajoutée à la liste en mémoire :



Si la liste déroulante a été créée à partir d'une liste définie en mode Développement, la liste d'origine n'est pas modifiée.

Grammaire JSON

Nom	Type de données	Valeurs possibles
automaticInsertion	boolean	true, false

Objets pris en charge

[Combo Box - Colonne List Box](#)

Enumération

Associe une liste de choix à un objet. Il peut s'agir d'un nom de liste de choix (une référence de liste) ou d'une collection de valeurs par défaut.

You can also associate choice lists to objects using the [OBJECT SET LIST BY NAME](#) or [OBJECT SET LIST BY REFERENCE](#) commands.

Grammaire JSON

Nom	Type de données	Valeurs possibles
choiceList	liste, collection	Une liste de valeurs possibles
liste	liste, collection	Une liste de valeurs possibles (listes hiérarchiques uniquement)

Objets pris en charge

[Drop-down List](#) - [Combo Box](#) - [Hierarchical List](#) - [List Box Column](#)

Enumération (liste statique)

Liste de valeurs statiques à utiliser comme étiquettes pour l'objet onglet.

Grammaire JSON

Nom	Type de données	Valeurs possibles
labels	liste, collection	Une liste de valeurs à saisir dans l'onglet

Objets pris en charge

[Onglets](#)

Élément courant

`Listbox de type collection ou entity selection`

Indique une variable ou une expression qui se verra attribuer l'élément/l'entité de collection sélectionné(e) par l'utilisateur. Vous devez utiliser une variable objet ou une expression assignable qui accepte des objets. Si l'utilisateur ne sélectionne rien ou si vous avez utilisé une collection de valeurs scalaires, la valeur Null est affectée.

Cette propriété est en "lecture seule", elle est automatiquement mise à jour en fonction des actions de l'utilisateur dans la list box. Vous ne pouvez pas modifier sa valeur pour modifier l'état de sélection de la list box.

Grammaire JSON

Nom	Type de données	Valeurs possibles
currentItemSource	string	Expression d'objet

Objets pris en charge

[List Box](#)

Position élément courant

`Listbox de type collection ou entity selection`

Indique une variable ou une expression qui se verra attribuer un entier long indiquant la position de l'élément/l'entité de collection sélectionné(e) par l'utilisateur.

- si aucun(e) élément/entité n'est sélectionné(e), la variable ou l'expression reçoit zéro,
- si un(e) seul(e) élément/entité est sélectionné(e), la variable ou l'expression reçoit son emplacement,
- si plusieurs éléments/entités sont sélectionnés, la variable ou l'expression reçoit la position de l'élément/entité qui a été sélectionné(e) en dernier.

Cette propriété est en "lecture seule", elle est automatiquement mise à jour en fonction des actions de l'utilisateur dans la list box. Vous ne pouvez pas modifier sa valeur pour modifier l'état de sélection de la list box.

Grammaire JSON

Nom	Type de données	Valeurs possibles
currentItemPositionSource	string	Expression numérique

Objets pris en charge

List Box

Data Type (expression type)

Defines the data type for the displayed expression. This property is used with:

- [List box columns](#) of the selection and collection types.
- [Drop-down lists](#) associated to objects or arrays.

See also [Expression Type](#) section.

Grammaire JSON

Nom	Type de données	Valeurs possibles
dataSourceTypeHint	string	<ul style="list-style-type: none">• list box columns: "boolean", "number", "picture", "text", "date", "time". <i>Array/selection list box only:</i> "integer", "object"• drop-down lists: "object", "arrayText", "arrayDate", "arrayTime", "arrayNumber"

Objets pris en charge

[Drop-down Lists](#) associated to objects or arrays - [List Box column](#)

Data Type (list)

Defines the type of data to save in the field or variable associated to the [drop-down list](#). This property is used with:

- Drop-down lists [associated to a choice list](#).
- Drop-down lists [associated to a hierarchical choice list](#).

Trois options sont disponibles :

- List reference: declares that the drop-down list is hierarchical. It means that the drop-down list can display up to two hierarchical levels and its contents can be managed by the 4D language commands of the Hierarchical Lists theme.
- Selected item value (default): the drop-down list is not hierarchical and the value of the item chosen in the list by the user is saved directly. For example, if the user chooses the value "Blue", then this value is saved in the field.
- Selected item reference: the drop-down list is not hierarchical and the reference of the choice list item is saved in the object. This reference is the numeric value associated with each item either through the *itemRef* parameter of the [APPEND TO LIST](#) or [SET LIST ITEM](#) commands, or in the list editor. This option lets you optimize memory usage: storing numeric values in fields uses less space than storing strings. It also makes it easier to translate applications: you just create multiple lists in different languages but with the same item references, then load the list based on the language of the application.

Using the Selected item reference option requires compliance with the following principles:

- To be able to store the reference, the field or variable data source must be of the Number type (regardless of the type of value displayed in the list). The [expression](#) property is automatically set.
- Valid and unique references must be associated with list items.

- The drop-down list must be associated with a field or a variable.

Grammaire JSON

Nom	Type de données	Valeurs possibles
saveAs	string	"value", "reference"

Setting only `"dataSourceTypeHint" : "integer"` with a `"type": "dropdown"` form object will declare a hierarchical drop-down list.

Objets pris en charge

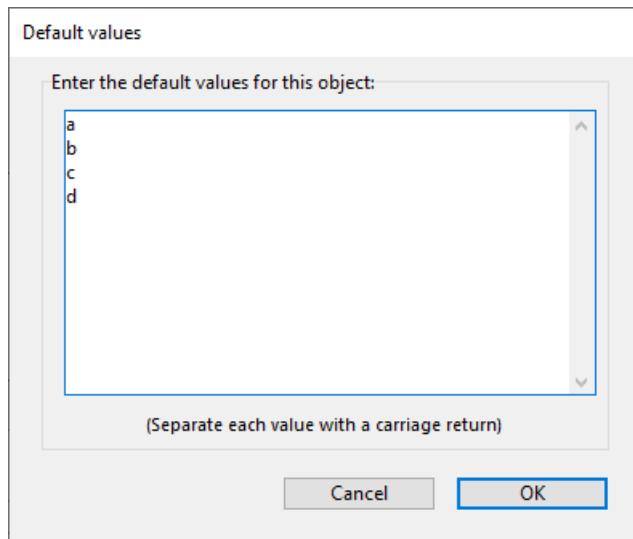
[Listes déroulantes](#) associées aux listes

Valeurs par défaut

List of values that will be used as default values for the list box column (array type only). Ces valeurs seront automatiquement accessibles dans la [variable tableau](#) associée à la colonne lors de l'exécution du formulaire. Using the language, you can manage the object by referring to this array.

Do not make confusion between this property and the "[default value](#)" property that allows to define a field value in new records.

Vous devez saisir une liste de valeurs. In the Form editor, a specific dialog box allows you to enter values separated by carriage returns:



You can also define a [choice list](#) with the list box column. However, a choice list will be used as list of selectable values for each column row, whereas the default list fill all column rows.

Grammaire JSON

Nom	Type de données	Valeurs possibles
values	collection	A collection of default values (strings), ex: "a", "b", "c", "d"

Objets pris en charge

[List Box Column \(array type only\)](#)

Expression

This description is specific to [selection](#) and [collection](#) type list box columns. See also [Variable or Expression](#) section.

A 4D expression to be associated with a column. You can enter:

- A simple variable (in this case, it must be explicitly declared for compilation). You can use any type of variable except BLOBs and arrays. The value of the variable will be generally calculated in the [On Display Detail](#) event.
- A field using the standard [Table]Field syntax ([selection type list box](#) only), for example: `[Employees]LastName`. The following types of fields can be used:

- Chaine
- Numeric
- Date
- Heure
- Image
- Boolean

You can use fields from the Master Table or from other tables.

- A 4D expression (simple expression, formula or 4D method). The expression must return a value. The value will be evaluated in the [On Display Detail](#) and [On Data Change](#) events. The result of the expression will be automatically displayed when you switch to Application mode. The expression will be evaluated for each record of the selection (current or named) of the Master Table (for selection type list boxes), each element of the collection (for collection type list boxes) or each entity of the selection (for entity selection list boxes). If it is empty, the column will not display any results.

The following expression types are supported:

- Chaine
- Numeric
- Date
- Image
- Booléen

For collection/entity selection list boxes, Null or unsupported types are displayed as empty strings.

When using collections or entity selections, you will usually declare the element property or entity attribute associated to a column within an expression containing `This`. `This` is a dedicated 4D command that returns a reference to the currently processed element. For example, you can use `This.<propertyPath>` where `<propertyPath>` is the path of a property in the collection or an entity attribute path to access the current value of each element/entity.

If you use a collection of scalar values, 4D will create an object for each collection element with a single property (named "value"), filled with the element value. In this case, you will use `This.value` as expression.

If a [non-assignable expression](#) is used (e.g. `[Person]FirstName + [Person]LastName`), the column is never enterable even if the [Enterable](#) property is enabled.

If a field, a variable, or an assignable expression (e.g. `Person.lastName`) is used, the column can be enterable or not depending on the [Enterable](#) property.

Grammaire JSON

Nom	Type de données	Valeurs possibles
dataSource	string	A 4D variable, field name, or an arbitrary complex language expression.

Objets pris en charge

Colonne de list box

Table principale

Current selection list boxes

Specifies the table whose current selection will be used. This table and its current selection will form the reference for the fields associated with the columns of the list box (field references or expressions containing fields). Even if some columns contain fields from other tables, the number of rows displayed will be defined by the master table.

All database tables can be used, regardless of whether the form is related to a table (table form) or not (project form).

Grammaire JSON

Nom	Type de données	Valeurs possibles
table	number	Numéro de la table

Objets pris en charge

List Box

Enregistrer comme

This property is available in the following conditions:

- a [choice list](#) is associated with the object
- for [inputs](#) and [list box columns](#), a [required list](#) is also defined for the object (both options should use usually the same list), so that only values from the list can be entered by the user.

This property specifies, in the context of a field or variable associated with a list of values, the type of contents to save:

- Save as Value (default option): the value of the item chosen in the list by the user is saved directly. For example, if the user chooses the value "Blue", then this value is saved in the field.
- Save as Reference: the reference of the choice list item is saved in the object. This reference is the numeric value associated with each item either through the *itemRef* parameter of the [APPEND TO LIST](#) or [SET LIST ITEM](#) commands, or in the list editor.

This option lets you optimize memory usage: storing numeric values in fields uses less space than storing strings. It also makes it easier to translate applications: you just create multiple lists in different languages but with the same item references, then load the list based on the language of the application.

Using this property requires compliance with the following principles:

- To be able to store the reference, the field or variable data source must be of the Number type (regardless of the type of value displayed in the list). The [expression](#) property is automatically set.
- Valid and unique references must be associated with list items.

Grammaire JSON

Nom	Type de données	Valeurs possibles
saveAs	string	"value", "reference"

Objets pris en charge

Zone de saisie - Colonne List Box

Eléments sélectionnés

Listbox de type collection ou entity selection

Specifies a variable or expression that will be assigned the elements or entities selected by the user.

- for a collection list box, you must use a collection variable or an assignable expression that accepts collections,

- for an entity selection list box, an entity selection object is built. Vous devez utiliser une variable objet ou une expression assignable qui accepte des objets.

Cette propriété est en "lecture seule", elle est automatiquement mise à jour en fonction des actions de l'utilisateur dans la list box. Vous ne pouvez pas modifier sa valeur pour modifier l'état de sélection de la list box.

Grammaire JSON

Nom	Type de données	Valeurs possibles
selectedItemsSource	string	Collection expression

Objets pris en charge

[List Box](#)

Selection Name

Named selection list boxes

Specifies the named selection to be used. You must enter the name of a valid named selection. It can be a process or interprocess named selection. The contents of the list box will be based on this selection. The named selection chosen must exist and be valid at the time the list box is displayed, otherwise the list box will be displayed blank.

Named selections are ordered lists of records. They are used to keep the order and current record of a selection in memory. For more information, refer to Named Selections section in the *4D Language Reference manual*.

Grammaire JSON

Nom	Type de données	Valeurs possibles
namedSelection	string	Nom de la sélection

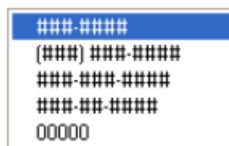
Objets pris en charge

[List Box](#)

Affichage

Alpha Format

Alpha formats control the way the alphanumeric fields and variables appear when displayed or printed. Here is a list of formats provided for alphanumeric fields:



You can choose a format from this list or use any custom format. The default list contains formats for some of the most common alpha fields that require formats: US telephone numbers (local and long distance), Social Security numbers, and zip codes. You can also enter a custom format name set in the Filters and formats editor of the tool box. In this case, the format cannot be modified in the object properties. Any custom formats or filters that you have created are automatically available, preceded by a vertical bar (|).

The number sign (#) is the placeholder for an alphanumeric display format. You can include the appropriate dashes, hyphens, spaces, and any other punctuation marks that you want to display. You use the actual punctuation marks you want and the number sign for each character you want to display.

For example, consider a part number with a format such as "RB-1762-1".

The alpha format would be:

```
##-#####-#
```

When the user enters "RB17621," the field displays:

```
RB-1762-1
```

The field actually contains "RB17621".

If the user enters more characters than the format allows, 4D displays the last characters. For example, if the format is:

```
(#####)
```

and the user enters "proportion", the field displays:

```
(portion)
```

The field actually contains "proportion". 4D accepts and stores the entire entry no matter what the display format. No information is lost.

Grammaire JSON

Nom	Type de données	Valeurs possibles
textFormat	string	"### ####", "(###) ###-####", "###-###-####", "###-##-####", "00000", custom formats

Date Format

Date formats control the way dates appear when displayed or printed. For data entry, you enter dates in the MM/DD/YYYY format, regardless of the display format you have chosen.

Unlike [Alpha](#) and [Number](#) formats, display formats for dates must only be selected among the 4D built-in formats.

The table below shows choices available:

Nom du format	Chaine JSON	Exemple (système US)
System date short	- (default)	03/25/20
System date abbreviated (1)	systemMedium	Wed, Mar 25, 2020
System date long	systemLong	Wednesday, March 25, 2020
RFC 822	rfc822	Tue, 25 Mar 2020 22:00:00 GMT
Short Century	shortCentury	03/25/20 but 04/25/2032 (2)
Internal date long	long	March 25, 2020
Internal date abbreviated (1)	abbreviated	Mar 25, 2020
Internal date short	short	03/25/2020
ISO Date Time (3)	iso8601	2020-03-25T00:00:00

(1) Pour éviter toute ambiguïté et conformément à la pratique actuelle, les formats de date abrégés affichent "jun" pour juin et "jul" pour juillet. Cette particularité ne s'applique qu'aux versions françaises de 4D.

(2) L'année est affichée avec deux chiffres lorsqu'elle appartient à l'intervalle (1930;2029), sinon elle est affichée avec quatre chiffres. Ce modèle est par défaut mais il peut être modifié à l'aide de la commande [SET DEFAULT CENTURY](#).

(3) Le format `ISO Date Time` correspond à la norme XML de représentation de la date et de l'heure (ISO8601). Il est principalement destiné à être utilisé lors de l'import/export de données au format XML et dans les services Web.

Quel que soit le format d'affichage, si l'année est saisie avec deux chiffres, 4D considère que le siècle est le 21ème si l'année appartient à l'intervalle (00;29) et le 20e si elle appartient à l'intervalle (30;99). Il s'agit du paramètre par défaut, mais il peut être modifié à l'aide de la commande [SET DEFAULT CENTURY](#).

Grammaire JSON

Nom	Type de données	Valeurs possibles
dateFormat	string	"systemShort", "systemMedium", "systemLong", "iso8601", "rfc822", "short", "shortCentury", "abbreviated", "long", "blankIfNull" (peut être combiné avec les autres valeurs possibles)

Number Format

Number fields include the Integer, Long integer, Integer 64 bits, Real and Float types.

Number formats control the way numbers appear when displayed or printed. For data entry, you enter only the numbers (including a decimal point or minus sign if necessary), regardless of the display format you have chosen.

4D provides various default number formats.

Placeholders

In each of the number display formats, the number sign (#), zero (0), caret (^), and asterisk (*) are used as placeholders. You create your own number formats by using one placeholder for each digit you expect to display.

Placeholder	Effect for leading or trailing zero
#	Displays nothing
0	Displays 0
^	Displays a space (1)
*	Displays an asterisk

(1) The caret (^) generates a space character that occupies the same width as a digit in most fonts.

For example, if you want to display three-digit numbers, you could use the format ###. If the user enters more digits than the format allows, 4D displays <<< in the field to indicate that more digits were entered than the number of digits specified in the display format.

If the user enters a negative number, the leftmost character is displayed as a minus sign (unless a negative display format has been specified). If ##0 is the format, minus 26 is displayed as -26 and minus 260 is displayed as <<< because the minus sign occupies a placeholder and there are only three placeholders.

No matter what the display format, 4D accepts and stores the number entered in the field. No information is lost.

Each placeholder character has a different effect on the display of leading or trailing zeros. A leading zero is a zero that starts a number before the decimal point; a trailing zero is a zero that ends a number after the decimal point.

Suppose you use the format ##0 to display three digits. If the user enters nothing in the field, the field displays 0. If the user enters 26, the field displays 26.

Separator characters

The numeric display formats (except for scientific notations) are automatically based on regional system parameters. 4D replaces the “.” and “,” characters by, respectively, the decimal separator and the thousand separator defined in the operating system. The period and comma are thus considered as placeholder characters, following the example of 0 or #.

On Windows, when using the decimal separator key of the numeric keypad, 4D makes a distinction depending on the type of field where the cursor is located: * in a Real type field, using this key will insert the decimal separator defined in the system, * in any other type of field, this key inserts the character associated with the key, usually a period (.) or comma (,).

Decimal points and other display characters

You can use a decimal point in a number display format. If you want the decimal to display regardless of whether the user types it in, it must be placed between zeros.

You can use any other characters in the format. When used alone, or placed before or after placeholders, the characters

always appear. For example, if you use the following format:

```
$##0
```

a dollar sign always appears because it is placed before the placeholders.

If characters are placed between placeholders, they appear only if digits are displayed on both sides. For example, if you define the format:

```
###.##0
```

the point appears only if the user enters at least four digits.

Spaces are treated as characters in number display formats.

Formats for positive, negative, and zero

A number display format can have up to three parts allowing you to specify display formats for positive, negative, and zero values. You specify the three parts by separating them with semicolons as shown below:

```
Positive;Negative;Zero
```

You do not have to specify all three parts of the format. If you use just one part, 4D uses it for all numbers, placing a minus sign in front of negative numbers.

If you use two parts, 4D uses the first part for positive numbers and zero and the second part for negative numbers. If you use three parts, the first is for positive numbers, the second for negative numbers, and the third for zero.

The third part (zero) is not interpreted and does not accept replacement characters. If you enter `###;###;#`, the zero value will be displayed "#". In other words, what you actually enter is what will be displayed for the zero value.

Here is an example of a number display format that shows dollar signs and commas, places negative values in parentheses, and does not display zeros:

```
$###,##0.00;($###,##0.00);
```

Notice that the presence of the second semicolon instructs 4D to use nothing to display zero. The following format is similar except that the absence of the second semicolon instructs 4D to use the positive number format for zero:

```
$###,##0.00;($###,##0.00)
```

In this case, the display for zero would be \$0.00.

Scientific notation

If you want to display numbers in scientific notation, use the ampersand (&) followed by a number to specify the number of digits you want to display. For example, the format:

```
&3
```

would display 759.62 as:

7.60e+2

The scientific notation format is the only format that will automatically round the displayed number. Note in the example above that the number is rounded up to 7.60e+2 instead of truncating to 7.59e+2.

Hexadecimal formats

You can display a number in hexadecimal using the following display formats:

- `&x` : This format displays hexadecimal numbers using the “0xFFFF” format.
- `&$` : This format displays hexadecimal numbers using the “\$FFFF” format.

XML notation

The `&xml` format will make a number compliant with XML standard rules. In particular, the decimal separator character will be a period “.” in all cases, regardless of the system settings.

Displaying a number as a time

You can display a number as a time (with a time format) by using `&/` followed by a digit. Time is determined by calculating the number of seconds since midnight that the value represents. The digit in the format corresponds to the order in which the time format appears in the Format drop-down menu.

For example, the format:

`&/5`

corresponds to the 5th time format in the pop-up menu, specifically the AM/PM time. A number field with this format would display 25000 as:

6:56 AM

Exemples

The following table shows how different formats affect the display of numbers. The three columns — Positive, Negative, and Zero — each show how 1,234.50, -1,234.50, and 0 would be displayed.

Format Entered	Positive	Negative	Zero
###	<<<	<<<	
### #	1234	<<<<	
##### ##	1234	-1234	
####.##	1234.5	-1234.5	
### ##.00	1234.50	-1234.50	0.00
### ##0	1234	-1234	0
+### ##0;-### ##0;0	+1234	-1234	0
### ##0DB;### ##0CR;0	1234DB	1234CR	0
### ##0;(### ##0)	1234	(1234)	0
##,##0	1,234	-1,234	0
##,##0.00	1,234.50	-1,234.50	0.00
~~~~~	1234	-1234	
~~~~~^0	1234	-1234	0
~,^0	1,234	-1,234	0
~,^0.00	1,234.50	-1,234.50	0.00
*****	***1234	**-1234	*****
*****0	***1234	**-1234	*****0
, **0	**1,234	*-1,234	***0
*, **0.00	*1,234.50	-1,234.50	*****0.00
\$*, **0.00;-\$*, **0.00	\$1,234.50	-\$1,234.50	\$*****0.00
\$~~~~^0	\$ 1234	\$-1234	\$ 0
\$~~~^0;-\$~~~^0	\$1234	-\$1234	\$ 0
\$~~~^0 ;(\$~~~^0)	\$1234	(\$1234)	\$ 0
\$~,^0.00 ;(\$~,^0.00)	\$1,234.50	(\$1,234.50)	\$ 0.00
&2	1.2e+3	-1.2e+3	0.0e+0
&5	1.23450e+3	-1.23450e+3	0.00000
&xml	1234.5	-1234.5	0

Grammaire JSON

Nom	Type de données	Valeurs possibles
numberFormat	string	Numbers (including a decimal point or minus sign if necessary)

Objets pris en charge

[Combo Box](#) - [Drop-down List](#) - [Input](#) - [List Box Column](#) - [List Box Footer](#) - [Progress Indicators](#)

Picture Format

Picture formats control how pictures appear when displayed or printed. For data entry, the user always enters pictures by pasting them from the Clipboard or by drag and drop, regardless of the display format.

The truncation and scaling options do not affect the picture itself. The contents of a Picture field are always saved. Only the display on the particular form is affected by the picture display format.

Scaled to fit

JSON grammar: "scaled"

The Scaled to fit format causes 4D to resize the picture to fit the dimensions of the area.



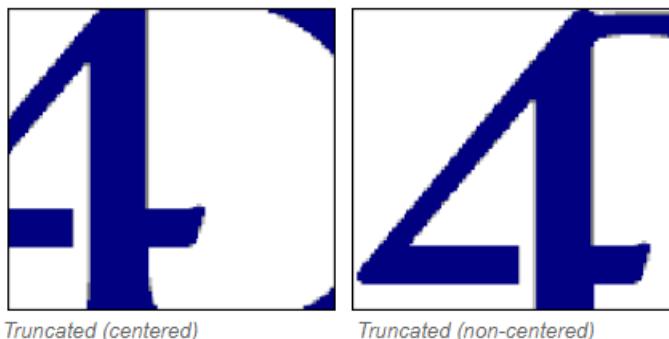
Truncated (centered and non-centered)

JSON grammar: "truncatedCenter" / "truncatedTopLeft"

The Truncated (centered) format causes 4D to center the picture in the area and crop any portion that does not fit within the area. 4D crops equally from each edge and from the top and bottom.

The Truncated (non-centered) format causes 4D to place the upper-left corner of the picture in the upper-left corner of the area and crop any portion that does not fit within the area. 4D crops from the right and bottom.

When the picture format is Truncated (non-centered), it is possible to add scroll bars to the input area.



Scaled to fit (proportional) and Scaled to fit centered (proportional)

JSON grammar: "proportionalTopLeft" / "proportionalCenter"

When you use Scaled to fit (proportional), the picture is reduced proportionally on all sides to fit the area created for the picture. The Scaled to fit centered (proportional) option does the same, but centers the picture in the picture area.

If the picture is smaller than the area set in the form, it will not be modified. If the picture is bigger than the area set in the form, it is proportionally reduced. Since it is proportionally reduced, the picture will not appear distorted.

If you have applied the Scaled to fit centered (proportional) format, the picture is also centered in the area:



Scaled to fit (proportional)

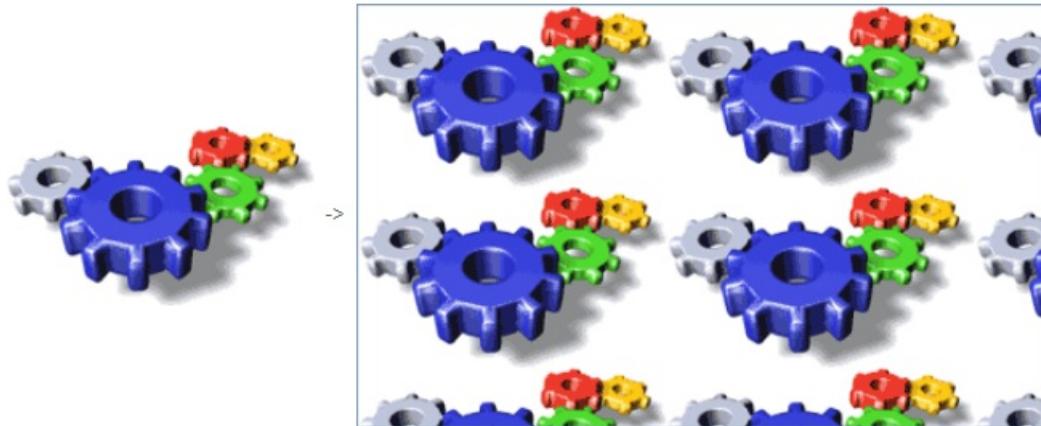


Scaled to fit centered (proportional)

Replicated

JSON grammar: "tiled"

When the area that contains a picture with the Replicated format is enlarged, the picture is not deformed but is replicated as many times as necessary in order to fill the area entirely.



If the field is reduced to a size smaller than that of the original picture, the picture is truncated (non-centered).

Grammaire JSON

Nom	Type de données	Valeurs possibles
pictureFormat	string	"truncatedTopLeft", "scaled", "truncatedCenter", "tiled", "proportionalTopLeft", "proportionalCenter"

Objets pris en charge

[Input](#) - [List Box Column](#) - [List Box Footer](#)

Time Format

Time formats control the way times appear when displayed or printed. For data entry, you enter times in the 24-hour HH:MM:SS format or the 12-hour HH:MM:SS AM/PM format, regardless of the display format you have chosen.

Unlike [Alpha](#) and [Number](#) formats, display formats for times must only be selected among the 4D built-in formats.

The table below shows the Time field display formats and gives examples:

Nom du format	Chaine JSON	Commentaires	Exemple pour 04:30:25
HH:MM:SS	hh_mm_ss		04:30:25
HH:MM	hh_mm		04:30
Hour Min Sec	HH_MM_SS		4 heures 30 minutes 25 secondes
Hour Min	HH_MM		4 heures 30 minutes
HH:MM AM/PM	hh_mm_am		4:30 a.m.
MM SS	mm_ss	Heure exprimée sous forme de durée à partir de 00:00:00	270:25
Min Sec	MM_SS	Heure exprimée sous forme de durée à partir de 00:00:00	270 Minutes 25 Secondes
ISO Date Time	iso8601	Corresponds to the XML standard for representing time-related data. It is mainly intended to be used when importing/exporting data in XML format	0000-00-00T04:30:25
System time short	- (default)	Standard time format defined in the system	04:30:25
System time long abbreviated	systemMedium	macOS only: Abbreviated time format defined in the system. Windows: this format is the same as the System time short format	4•30•25 AM
System time long	systemLong	macOS only: Long time format defined in the system. Windows: this format is the same as the System time short format	4:30:25 AM HNEC

Grammaire JSON

Nom	Type de données	Valeurs possibles
timeFormat	string	"systemShort", "systemMedium", "systemLong", "iso8601", "hh_mm_ss", "hh_mm", "hh_mm_am", "mm_ss", "HH_MM_SS", "HH_MM", "MM_SS", "blankIfNull" (can be combined with the other possible values)

Objets pris en charge

[Combo Box](#) - [Drop-down List](#) - [Input](#) - [List Box Column](#) - [List Box Footer](#)

Text when False/Text when True

When a [boolean expression](#) is displayed as:

- a text in an [input object](#)
- a "[popup](#)" in a [list box column](#),

... you can select the text to display for each value:

- Text when True - the text to be displayed when the value is "true"
- Text when False - the text to be displayed when the value is "false"

Grammaire JSON

Nom	Type de données	Valeurs possibles
booleanFormat	string	"<textWhenTrue>;<textWhenFalse>", e.g. "Assigned;Unassigned"

Objets pris en charge

[List Box Column - Input](#)

Type d'affichage

Used to associate a display format with the column data. The formats provided depends on the variable type (array type list box) or the data/field type (selection and collection type list boxes).

Boolean and number (numeric or integer) columns can be displayed as check boxes. In this case, the [Title](#) property can be defined.

Boolean columns can also be displayed as pop-up menus. In this case, the [Text when False](#) and [Text when True](#) properties must be defined.

Grammaire JSON

Nom	Type de données	Valeurs possibles
controlType	string	<ul style="list-style-type: none"> number columns: "automatic" (default) or "checkbox" boolean columns: "checkbox" (default) or "popup"

Objets pris en charge

[Colonne de list box](#)

Not rendered

When this property is enabled, the object is not drawn on the form, however it can still be activated.

In particular, this property allows implementing "invisible" buttons. Non-rendered buttons can be placed on top of graphic objects. They remain invisible and do not highlight when clicked, however their action is triggered when they are clicked.

Grammaire JSON

Nom	Type de données	Valeurs possibles
display	boolean	true, false

Objets pris en charge

[Button - Drop-down List](#)

Three-States

Allows a check box object to accept a third state. La variable associée à la case à cocher retourne la valeur 2 lorsque celle-ci se trouve dans le troisième état.

Three-states check boxes in list box columns

List box columns with a numeric [data type](#) can be displayed as three-states check boxes. If chosen, the following values

are displayed:

- 0 = unchecked box,
- 1 = checked box,
- 2 (or any value >0) = semi-checked box (third state). For data entry, this state returns the value 2.
- -1 = invisible check box,
- -2 = unchecked box, not enterable,
- -3 = checked box, not enterable,
- -4 = semi-checked box, not enterable

In this case as well, the [Title](#) property is also available so that the title of the check box can be entered.

Grammaire JSON

Nom	Type de données	Valeurs possibles
threeState	boolean	true, false

Objets pris en charge

[Check box - List Box Column](#)

Titre de menu

This property is available for a list box column if:

- the [column type](#) is boolean and its [display type](#) is "Check Box"
- the [column type](#) is number (numeric or integer) and its [display type](#) is "Three-states Checkbox".

In that cases, the title of the check box can be entered using this property.

Grammaire JSON

Nom	Type de données	Valeurs possibles
controlTitle	string	Any custom label for the check box

Objets pris en charge

[Colonne de list box](#)

Truncate with ellipsis

Controls the display of values when list box columns are too narrow to show their full contents.

This option is available for columns with any type of contents, except pictures and objects.

- When the property is enabled (default), if the contents of a list box cell exceed the width of the column, they are truncated and an ellipsis is displayed:

Cities	Popu...
Charlotte	792862
New York	8406...
Philadelp...	1553...
San Fran...	837442
San Jose	288054

The position of the ellipsis depends on the OS. In the above example (Windows), it is added on the right side of

the text. On macOS, the ellipsis is added in the middle of the text.

- When the property is disabled, if the contents of a cell exceed the width of the column, they are simply clipped with no ellipsis added:

Cities	Popu...
Charlotte	792862
New York	3406000
Philadelphia	1553000
San Francisco	837442
San Jose	288054

The Truncate with ellipsis option is enabled by default and can be specified with list boxes of the Array, Selection, or Collection type.

When applied to Text type columns, the Truncate with ellipsis option is available only if the [Wordwrap](#) option is not selected. When the Wordwrap property is selected, extra contents in cells are handled through the word-wrapping features so the Truncate with ellipsis property is not available.

The Truncate with ellipsis property can be applied to Boolean type columns; however, the result differs depending on the [cell format](#):

- For Pop-up type Boolean formats, labels are truncated with an ellipsis,
- For Check box type Boolean formats, labels are always clipped.

Grammaire JSON

Nom	Type de données	Valeurs possibles
truncateMode	string	"withEllipsis", "none"

Objets pris en charge

[List Box Column](#) - [List Box Header](#)

Visibilité

This property allows hiding the object in the Application environment.

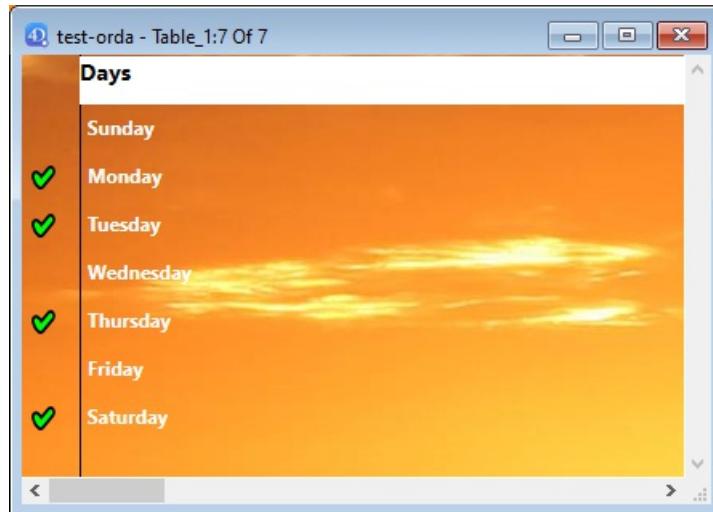
You can handle the Visibility property for most form objects. This property is mainly used to simplify dynamic interface development. In this context, it is often necessary to hide objects programatically during the `On load` event of the form then to display certain objects afterwards. In this context, it is often necessary to hide objects programatically during the `On load` event of the form then to display certain objects afterwards. The Visibility property allows inverting this logic by making certain objects invisible by default.

Automatic visibility in list forms

In the context of ["list" forms](#), the Visibility property supports two specific values:

- If record selected (JSON name: "selectedRows")
- If record not selected (JSON name: "unselectedRows")

This property is only used when drawing objects located in the body of a list form. It tells 4D whether or not to draw the object depending on whether the record being processed is selected/not selected. It allows you to represent a selection of records using visual attributes other than highlight colors:



4D does not take this property into account if the object was hidden using the `OBJECT SET VISIBLE` command; in this case, the object remains invisible regardless of whether or not the record is selected.

Grammaire JSON

Nom	Type de données	Valeurs possibles
visibility	string	"visible", "hidden", "selectedRows" (list form only), "unselectedRows" (list form only)

Objets pris en charge

4D View Pro area - 4D Write Pro area - Button - Button Grid - Check Box - Combo Box - Drop-down List - Group Box - Hierarchical List - List Box - List Box Column - List Box Footer - List Box Header - Picture Button - Picture Pop-up Menu - Plug-in Area - Progress indicator - Radio Button - Spinner - Splitter - Static Picture - Stepper - Subform - Tab control - Text Area - Web Area

Wordwrap

Pour les `input`, disponibles lorsque la propriété `Multiligne` est définie sur "oui".

Manages the display of contents when it exceeds the width of the object.

Checked for list box/Yes for input

JSON grammar: "normal"

When this option is selected, text automatically wraps to the next line whenever its width exceeds that of the column/area, if the column/area height permits it.

- In single-line columns/areas, only the last word that can be displayed entirely is displayed. 4D inserts line returns; it is possible to scroll the contents of the area by pressing the down arrow key.
- In multiline columns/areas, 4D carries out automatic line returns.

Name	A BLOB is loaded into memory in its entirety. A
Text	<p>A BLOB is loaded into memory in its entirety. A BLOB variable or BLOB array is held and exists in memory only. A BLOB field is loaded into memory from the disk, like the rest of the record to which it belongs.</p> <p>Like the other field types that can retain a large amount of data (such as the Picture field type), BLOB fields are not duplicated in memory when</p>

Unchecked for list box/No for input

JSON grammar: "none"

When this option is selected, 4D does not do any automatic line returns and the last word that can be displayed may be truncated. In text type areas, carriage returns are supported:

Name	A BLOB is loaded into memory in its entirety. A BLO
Text	A BLOB is loaded into memory in its entirety. A BLO Like the other field types that can retain a large amo

In list boxes, any text that is too long is truncated and displayed with an ellipse (...). In the following example, the Wordwrap option is checked for the left column and unchecked for the right column:

Header1	Header2
The vertical alignment will be applied as long as the full text fits in the cell. Otherwise, the text will be aligned to the top so as the beginning of the text will visible.	The vertical alignment will be ap...
You can make a field invisible in the Application environment and for the plug-ins by selecting the Invisible property for this field.	You can make a field invisible in ...

Note that regardless of the Wordwrap option's value, the row height is not changed. If the text with line breaks cannot be entirely displayed in the column, it is truncated (without an ellipse). In the case of list boxes displaying just a single row, only the first line of text is displayed:

Header1	Header2
The vertical alignment will be	The vertical alignment will be ap...
You can make a field invisible in	You can make a field invisible in ...

Automatic for input (default option)

JSON grammar: "automatic"

- In single-line areas, words located at the end of lines are truncated and there are no line returns.
- In multiline areas, 4D carries out automatic line returns.

Name	A BLOB is loaded into memory in its entirety. A BLO
Text	A BLOB is loaded into memory in its entirety. A BLOB variable or BLOB array is held and exists in memory only. A BLOB field is loaded into memory from the disk, like the rest of the record to which it belongs. Like the other field types that can retain a large amount of data (such as the Picture field type),

Grammaire JSON

Nom	Type de données	Valeurs possibles
wordwrap	string	"automatic" (à l'exception de list box), "normal", "none"

Objets pris en charge

[Input - List Box Column - List Box Footer](#)

Saisie

Correction orthographique

4D inclut des fonctionnalités de correction orthographique intégrées et personnalisables. Text type [inputs](#) can be checked, as well as [4D Write Pro](#) documents.

The Auto Spellcheck property activates the spell-check for each object. When used, a spell-check is automatically performed during data entry. You can also execute the `SPELL CHECKING` 4D language command for each object to be checked.

Grammaire JSON

Nom	Type de données	Valeurs possibles
spellcheck	boolean	true, false

Objets pris en charge

[Zone 4D Write Pro](#) - [Zone de saisie](#)

Context Menu

Allows the user access to a standard context menu in the object when the form is executed.

For a picture type [input](#), in addition to standard editing commands (Cut, Copy, Paste and Clear), the menu contains the Import... command, which can be used to import a picture stored in a file, as well as the Save as... command, which can be used to save the picture to disk. The menu can also be used to modify the display format of the picture: the Truncated non-centered, Scaled to fit and Scaled to fit centered prop. options are provided. The modification of the [display format](#) using this menu is temporary; it is not saved with the record.

For a [multi-style](#) text type [input](#), in addition to standard editing commands, the context menu provides the following commands:

- Fonts...: displays the font system dialog box
- Recent fonts: displays the names of recent fonts selected during the session. The list can store up to 10 fonts (beyond that, the last font used replaces the oldest). By default, this list is empty and the option is not displayed. You can manage this list using the `SET RECENT FONTS` and `FONT LIST` commands.
- commands for supported style modifications: font, size, style, color and background color. When the user modifies a style attribute via this pop-up menu, 4D generates the `On After Edit` form event.

For a [Web Area](#), the contents of the menu depend of the rendering engine of the platform. It is possible to control access to the context menu via the `WA SET PREFERENCE` command.

Grammaire JSON

Nom	Type de données	Valeurs possibles
contextMenu	string	"automatic" (used if missing), "none"

Objets pris en charge

[Input](#) - [Web Area](#) - [4D Write Pro areas](#)

Saisissable

The Enterable attribute indicates whether users can enter values into the object.

Objects are enterable by default. If you want to make a field or an object non-enterable for that form, you can disable the Enterable property for the object. A non-enterable object only displays data. You control the data by methods that use the field or variable name. You can still use the `On Clicked`, `On Double Clicked`, `On Drag Over`, `On Drop`, `On Getting Focus` and `On Losing Focus` form events with non-enterable objects. This makes it easier to manage custom context menus and lets you design interfaces where you can drag-and-drop and select non-enterable variables.

When this property is disabled, any pop-up menus associated with a list box column via a list are disabled.

Grammaire JSON

Nom	Type de données	Valeurs possibles
enterable	boolean	true, false

Objets pris en charge

[Zones 4D Write Pro](#) - [Check Box](#) - [Liste hiérarchique](#) - [Zone de saisie](#) - [Colonne List Box](#) - [Barre de progression](#) - [Règle](#) - [Stepper](#)

Filtres de saisie

Un filtre de saisie contrôle exactement ce que l'utilisateur peut taper au clavier pendant la saisie. Unlike [required lists](#) for example, entry filters operate on a character-by-character basis. Par exemple, si un numéro de composant est toujours constitué de trois lettres suivies de trois chiffres, vous pouvez contraindre la saisie à respecter cette forme. You can even control the particular letters and numbers.

Un filtre de saisie n'est effectif que pendant la saisie. Il n'a aucun effet sur l'affichage des données une fois que l'objet est désélectionné. En général, les filtres de saisie sont utilisés conjointement avec les [formats d'affichage](#). Le filtre agit pendant la saisie et le format d'affichage assure un affichage approprié de la valeur après sa saisie.

Pendant la saisie de données, un filtre de saisie évalue chaque caractère au moment où il est saisi. Si l'utilisateur tente de taper un caractère invalide (un chiffre à la place d'une lettre, par exemple), 4D refuse la saisie du caractère. The null character remains unchanged until the user types a valid character.

Les filtres de saisie peuvent aussi être utilisés pour afficher des caractères de formatage afin d'éviter à l'utilisateur de les taper. Par exemple, un numéro de téléphone français est constitué d'un chiffre de code opérateur suivi d'un chiffre de zone et d'un nombre à huit chiffres groupés par paires. Un format d'affichage peut être utilisé pour afficher le code opérateur entre parenthèses et pour afficher un tiret entre les paires de chiffres. Lorsqu'un tel format est utilisé, l'utilisateur n'a pas besoin de saisir les parenthèses ou le tiret.

Définition d'un filtre de saisie

La plupart du temps, les [filtres intégrés](#) de 4D répondront à vos besoins. Toutefois, vous pouvez créer des filtres personnalisés:

- you can directly enter a filter definition string
- or you can enter the name of an entry filter created in the Filters editor in the Toolbox. The names of custom filters you create begin with a vertical bar (!).

For information about creating entry filters, see [Filter and format codes](#).

Filtres par défaut

Ce tableau décrit les filtres de saisie du menu de sélection :

Filtres de saisie	Description
~A	Permet la saisie de toute lettre, mais les transforme en caractères majuscules.
&9	Permet tout chiffre.
&A	Ne permet que la saisie de lettres majuscules.
&a	Ne permet que la saisie de lettres (minuscules et majuscules).
&@	Ne permet que la saisie de caractères alphanumériques. Pas de caractères spéciaux.
~a##	State name abbreviation (e.g., CA). Permet la saisie de deux lettres, mais les transforme en caractères majuscules.
!0&9##/#/#/#	Filtre standard de saisie des dates. Affiche des zéros aux emplacements de saisie. Permet la saisie de tout chiffre.
!0&9 Day: ## Month: ## Year: ##	Filtre personnalisé de saisie de date. Affiche des zéros aux emplacements de saisie. Permet la saisie de tout chiffre. Limited to hours and minutes.
!0&9##:##	Filtre de saisie d'heure. Limited to hours and minutes. Affiche des zéros aux emplacements de saisie. Allow any four numbers, separated by a colon.
!0&9## Hrs ## Mins ## Secs	Filtre de saisie d'heure. Affiche des zéros aux emplacements de saisie. Allow any two numbers before each word.
!0&9Hrs: ## Mins: ## Secs: ##	Filtre de saisie d'heure. Affiche des zéros aux emplacements de saisie. Allow any two numbers after each word.
!0&9##-##-##-##	Local telephone number format. Affiche des zéros aux emplacements de saisie. Allow any number. Three entries, hyphen, four entries.
!_&9(###)!0###-####	Long distance telephone number. Display underscores in first three entry spaces, zeros in remainder.
!0&9##-##-##-##	Long distance telephone number. Affiche des zéros aux emplacements de saisie. Allow any number. Three entries, hyphen, three entries, hyphen, four entries.
!0&9##-##-##-##	Social Security number. Affiche des zéros aux emplacements de saisie. Permet la saisie de tout chiffre.
~"A-Z;0-9; ;;.;;"	Uppercase letters and punctuation. Allow only capital letters, numbers, spaces, commas, periods, and hyphens.
&"a-z;0-9; ;;.;;"	Upper and lowercase letters and punctuation. Allow lowercase letters, numbers, spaces, commas, periods, and hyphens.
&"0-9;.;;"	Numbers. Allow only numbers, decimal points, and hyphens (minus sign).

Grammaire JSON

Nom	Type de données	Valeurs possibles
entryFilter	string	<ul style="list-style-type: none"> Entry filter code or Entry filter name (filter names start with)

Objets pris en charge

Check Box - Combo Box - Liste hiérarchique - Zone de saisie - Colonne List Box

Focusable

When the Focusable property is enabled for an object, the object can have the focus (and can thus be activated by the keyboard for instance). It is outlined by a gray dotted line when it is selected — except when the Hide focus rectangle

option has also been selected.

An [input object](#) is always focusable if it has the [Enterable](#) property.

- **Current Member**
Check box shows focus when selected

- **Current Member**
Check box is selected but cannot show focus]

When the Focusable property is selected for a non-enterable object, the user can select, copy or even drag-and-drop the contents of the area.

Grammaire JSON

Nom	Type de données	Valeurs possibles
focusable	boolean	true, false

Objets pris en charge

[4D Write Pro areas](#) - [Button](#) - [Check Box](#) - [Drop-down List](#) - [Hierarchical List](#) - [Input](#) - [List Box](#) - [Plug-in Area](#) - [Radio Button](#) - [Subform](#)

Keyboard Layout

This property associates a specific keyboard layout to an [input object](#). For example, in an international application, if a form contains a field whose contents must be entered in Greek characters, you can associate the "Greek" keyboard layout with this field. This way, during data entry, the keyboard configuration is automatically changed when this field has the focus.

By default, the object uses the current keyboard layout.

You can also set and get the keyboard dynamically using the `OBJECT SET KEYBOARD LAYOUT` and `OBJECT Get keyboard layout` commands.

Grammaire JSON

Nom	Type de données	Valeurs possibles
keyboardDialect	Texte	Language code, for example "ar-ma" or "cs". See RFC3066, ISO639 and ISO3166

Objets pris en charge

[4D Write Pro areas](#) - [Input](#)

Multilignes

Cette propriété est disponible pour les [objets de zone de saisie](#) contenant les expressions de type texte et les champs de type alpha et texte. Elle peut prendre trois valeurs : Oui, Non, Automatique (par défaut).

Automatique

- Dans les zones mono-lignes, les mots situés en fin de ligne sont tronqués et il n'y a pas de retours à la ligne.

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

Name	A BLOB is loaded into memory in its entirety. A BLO
Text	A BLOB is loaded into memory in its entirety. A BLOB variable or BLOB array is held and exists in memory only. A BLOB field is loaded into memory from the disk, like the rest of the record to which it belongs. Like the other field types that can retain a large amount of data (such as the Picture field type), BLOB fields are not duplicated in memory when

Non

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

Name	A BLOB is loaded into memory in its entirety. A BLO
Text	A BLOB is loaded into memory in its entirety. A BLO

Oui

Lorsque cette valeur est sélectionnée, la propriété est gérée par l'option [Retour à la ligne](#).

Grammaire JSON

Nom	Type de données	Valeurs possibles
multiline	Texte	"yes", "no", "automatic" (par défaut si non défini)

Objets pris en charge

[Input](#)

Placeholder

4D can display placeholder text in the fields of your forms.

Placeholder text appears as watermark text in a field, supplying a help tip, indication or example for the data to be entered. This text disappears as soon as the user enters a character in the area:

Product Number	<input type="text" value="Enter 12-digit product code"/>
Product Number	<input type="text" value="1"/>

The placeholder text is displayed again if the contents of the field is erased.

A placeholder can be displayed for the following types of data:

- string (text or alpha)
- date and time when the Blank if null property is enabled.

You can use an XLIFF reference in the ":xliiff:resname" form as a placeholder, for example:

```
:xliiff:PH_Lastname
```

You only pass the reference in the "Placeholder" field; it is not possible to combine a reference with static text.

You can also set and get the placeholder text by programming using the [OBJECT SET PLACEHOLDER](#) and [OBJECT Get placeholder](#) commands.

Grammaire JSON

Nom	Type de données	Valeurs possibles
placeholder	string	Texte à afficher (grisé) lorsque l'objet ne contient aucune valeur

Objets pris en charge

[Combo Box - Zone de saisie](#)

Voir aussi

[Message d'aide](#)

Selection always visible

This property keeps the selection visible within the object after it has lost the focus. This makes it easier to implement interfaces that allow the text style to be modified (see [Multi-style](#)).

Grammaire JSON

Nom	Type de données	Valeurs possibles
showSelection	boolean	true, false

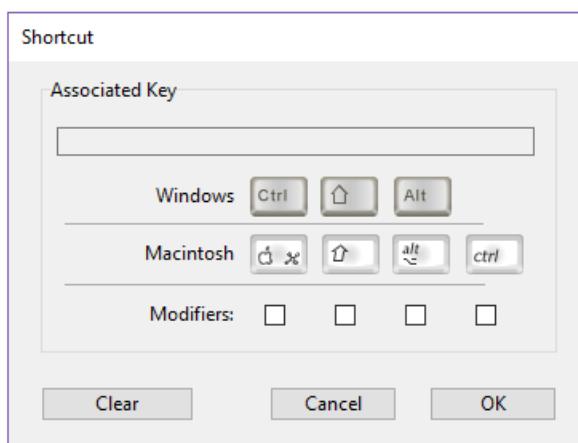
Objets pris en charge

[4D Write Pro areas - Input](#)

Shortcut

This property allows setting special meaning keys (keyboard shortcuts) for [buttons](#), [radio buttons](#), and [checkboxes](#). They allow the user to use the control using the keyboard instead of having to use the mouse.

You can configure this option by clicking the [...] button in the Shortcuts property in the Property List.



You can also assign a shortcut to a custom menu command. If there is a conflict between two shortcuts, the

active object has priority. For more information about associating shortcuts with menus, refer to [Setting menu properties](#).

To view a list of all the shortcuts used in the 4D Design environment, see the [Shortcuts Page](#) in the Preferences dialog box.

Grammaire JSON

Nom	Type de données	Valeurs possibles
shortcutAccel	boolean	true, false (Ctrl Windows/Command macOS)
shortcutAlt	boolean	true, false
shortcutCommand	boolean	true, false
shortcutControl	boolean	true, false (macOS Control)
shortcutShift	boolean	true, false
shortcutKey	string	<ul style="list-style-type: none">any character key: "a", "b"...[F1]" -> "[F15]", "[Return]", "[Enter]", "[Backspace]", "[Tab]", "[Esc]", "[Del]", "[Home]", "[End]", "[Help]", "[Page up]", "[Page down]", "[left arrow]", "[right arrow]", "[up arrow]", "[down arrow]"

Objets pris en charge

[Button](#) - [Check Box](#) - [Picture Button](#) - [Radio Button](#)

Saisie sur clic unique

Enables direct passage to edit mode in list boxes.

When this option is enabled, list box cells switch to edit mode after a single user click, regardless of whether or not this area of the list box was selected beforehand. Note that this option allows cells to be edited even when the list box [selection mode](#) is set to "None".

When this option is not enabled, users must first select the cell row and then click on a cell in order to edit its contents.

Grammaire JSON

Nom	Type de données	Valeurs possibles
singleClickEdit	boolean	true, false

Objets pris en charge

[List Box](#)

Pieds

Afficher pieds

Cette propriété est utilisée pour afficher ou masquer [les pieds de de colonne listbox](#). Il existe un pied par colonne; chaque pied est configuré séparément.

Grammaire JSON

Nom	Type de données	Valeurs possibles
showFooters	boolean	true, false

Objets pris en charge

[List Box](#)

Hauteur

Cette propriété sert à définir la hauteur de ligne d'un pied de list box en pixels ou en lignes de texte (lorsqu'elle est affichée). Les deux types d'unités peuvent être utilisés dans la même list box :

- *Pixel* - la valeur de hauteur est appliquée directement à la ligne concernée, quelle que soit la taille de la police contenue dans les colonnes. Si une police est trop grande, le texte est tronqué. De plus, les images sont tronquées ou redimensionnées selon leur format.
- *Ligne* - la hauteur est calculée en tenant compte de la taille de police de la ligne concernée.
 - Si plus d'une taille est définie, 4D utilise la plus grande. Par exemple, si une ligne contient «Verdana 18», «Geneva 12» et «Arial 9», 4D utilise «Verdana 18» pour déterminer la hauteur de ligne (par exemple, 25 pixels). Cette hauteur est ensuite multipliée par le nombre de lignes définies.
 - Ce calcul ne prend pas en compte la taille des images ni les styles appliqués aux polices.
 - Sous macOS, la hauteur de ligne peut être incorrecte si l'utilisateur saisit des caractères qui ne sont pas disponibles dans la police sélectionnée. Lorsque cela se produit, une police de remplacement est utilisée, ce qui peut entraîner des variations de taille.

Cette propriété peut également être définie dynamiquement à l'aide de la commande [LISTBOX SET FOOTERS HEIGHT](#).

Conversion d'unités : lorsque vous passez d'une unité à l'autre, 4D les convertit automatiquement et affiche le résultat dans la liste des propriétés. Par exemple, si la police utilisée est "Lucida grande 24", une hauteur de "1 ligne" est convertie en "30 pixels" et une hauteur de "60 pixels" est convertie en "2 lignes".

A noter que la conversion en va-et-vient peut conduire à un résultat final différent de la valeur de départ en raison des calculs automatiques effectués par 4D. Ceci est illustré dans les séquences suivantes :

(police Arial 18): 52 pixels -> 2 lignes -> 40 pixels (police Arial 12): 3 pixels -> 0.4 ligne arrondie à 1 ligne -> 19 pixels

Exemple JSON :

```
"List Box": {  
    "type": "listbox",  
    "showFooters": true,  
    "footerHeight": "44px",  
    ...  
}
```

Grammaire JSON

Nom	Type de données	Valeurs possibles
footerHeight	string	décimales positives +px em

Objets pris en charge

[List Box](#)

Voir aussi

[En-têtes - Pieds List box](#)

Quadrillage

Couleur lignes horizontales

Définit la couleur des lignes horizontales dans une list box (gris par défaut).

Grammaire JSON

Nom	Type de données	Valeurs possibles
horizontalLineStroke	color	une valeur css; "transparent"; "automatic"

Objets pris en charge

[List Box](#)

Couleur lignes verticales

Définit la couleur des lignes verticales d'une list box (gris par défaut).

Grammaire JSON

Nom	Type de données	Valeurs possibles
verticalLineStroke	color	une valeur css; "transparent"; "automatic"

Objets pris en charge

[List Box](#)

En-têtes

Afficher en-têtes

Cette propriété est utilisée pour afficher ou masquer [les en-têtes de colonne listbox](#). Il existe un en-tête par colonne; chaque en-tête est configuré séparément.

Grammaire JSON

Nom	Type de données	Valeurs possibles
showHeaders	boolean	true, false

Objets pris en charge

[List Box](#)

Hauteur

Cette propriété sert à définir la hauteur de ligne d'un en-tête de list box en pixels ou en lignes de texte (lorsqu'elle est affichée). Les deux types d'unités peuvent être utilisés dans la même list box :

- *Pixel* - la valeur de hauteur est appliquée directement à la ligne concernée, quelle que soit la taille de la police contenue dans les colonnes. Si une police est trop grande, le texte est tronqué. De plus, les images sont tronquées ou redimensionnées selon leur format.
- *Ligne* - la hauteur est calculée en tenant compte de la taille de police de la ligne concernée.
 - Si plus d'une taille est définie, 4D utilise la plus grande. Par exemple, si une ligne contient «Verdana 18», «Geneva 12» et «Arial 9», 4D utilise «Verdana 18» pour déterminer la hauteur de ligne (par exemple, 25 pixels). Cette hauteur est ensuite multipliée par le nombre de lignes définies.
 - Ce calcul ne prend pas en compte la taille des images ni les styles appliqués aux polices.
 - Sous macOS, la hauteur de ligne peut être incorrecte si l'utilisateur saisit des caractères qui ne sont pas disponibles dans la police sélectionnée. Lorsque cela se produit, une police de remplacement est utilisée, ce qui peut entraîner des variations de taille.

Cette propriété peut également être définie dynamiquement à l'aide de la commande [LISTBOX SET HEADERS HEIGHT](#).

Conversion d'unités : lorsque vous passez d'une unité à l'autre, 4D les convertit automatiquement et affiche le résultat dans la liste des propriétés. Par exemple, si la police utilisée est "Lucida grande 24", une hauteur de "1 ligne" est convertie en "30 pixels" et une hauteur de "60 pixels" est convertie en "2 lignes".

A noter que la conversion en va-et-vient peut conduire à un résultat final différent de la valeur de départ en raison des calculs automatiques effectués par 4D. Ceci est illustré dans les séquences suivantes :

(police Arial 18): 52 pixels -> 2 lignes -> 40 pixels (police Arial 12): 3 pixels -> 0.4 ligne arrondie à 1 ligne -> 19 pixels

Exemple JSON :

```
"List Box": {  
    "type": "listbox",  
    "showHeaders": true,  
    "headerHeight": "22px",  
    ...  
}
```

Grammaire JSON

Nom	Type de données	Valeurs possibles
headerHeight	string	décimales positives +px em)

Objets pris en charge

[List Box](#)

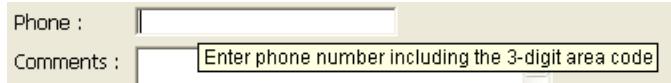
Voir aussi

[Pieds - En-têtes List box](#)

Aide

Infobulle

Cette propriété permet d'associer les messages d'aide aux objets actifs de vos formulaires. Ils peuvent être affichés au moment de l'exécution :



- Le délai d'affichage et la durée maximale des messages d'aide peuvent être définis à l'aide des sélecteurs de délai et de durée des conseils de la commande [SET DATABASE PARAMETER](#).
- Ces infobulles peuvent être désactivées ou activées pour l'application à l'aide du sélecteur de la commande [SET DATABASE PARAMETER](#).

Vous pouvez :

- désigner une info-bulle existante, préalablement spécifiée dans l'éditeur de [messages d'aide](#) de 4D.
- ou saisir directement le message d'aide sous forme de chaîne. Cela vous permet de profiter de l'architecture XLIFF. Vous pouvez saisir ici une référence XLIFF afin d'afficher un message dans le langage de l'application (pour plus d'informations sur XLIFF, reportez-vous à [l'Annexe B: Architecture XLIFF](#). Vous pouvez également utiliser des références 4D (voir [Utilisation de références en texte statique](#)).

Sous macOS, l'affichage des messages d'aide n'est pas pris en charge dans les fenêtres de type pop-up.

Grammaire JSON

Nom	Type de données	Valeurs possibles
tooltip	Texte	informations supplémentaires destinées à aider l'utilisateur

Objets pris en charge

[Bouton](#) - [Grille de boutons](#) - [Check Box](#) - [Liste déroulante](#) - [Combo Box](#) - [Liste hiérarchique](#) - [En-tête List Box](#) - [Pied List Box](#) - [Bouton image](#) - [Pop-up menu image](#) - [Bouton Radio](#)

Autres fonctionnalités d'aide

Vous pouvez aussi associer des messages d'aides aux objets formulaire de deux autres façons :

- au niveau de la structure de la base (champs uniquement). Dans ce cas, le message d'aide du champ apparaîtra sur les autres formulaires. Pour plus d'informations, référez-vous à la section [Propriétés des champs](#).
- en utilisant la commande [OBJECT SET HELP TIP](#), pour le process courant.

Lorsque différentes astuces sont associées au même objet à plusieurs emplacements, l'ordre de priorité suivant est appliqué :

1. structure (priorité la plus faible)
2. éditeur de formulaire
3. Commande [OBJECT SET HELP TIP](#) (haute priorité)

Voir aussi

[Placeholder](#)

Hiérarchie

List box hiérarchique

List box de type tableau

This property specifies that the list box must be displayed in hierarchical form. In the JSON form, this feature is triggered [when the `dataSource` property value is an array](#), i.e. a collection.

Additional options (Variable 1...10) are available when the *Hierarchical List Box* option is selected, corresponding to each `dataSource` array to use as break column. Each time a value is entered in a field, a new row is added. Up to 10 variables can be specified. These variables set the hierarchical levels to be displayed in the first column.

Voir [List box hiérarchiques](#)

Grammaire JSON

Nom	Type de données	Valeurs possibles
datasource	tableau chaîne	Collection de noms de tableaux définissant la hiérarchie

Objets pris en charge

[List Box](#)

List Box

Columns

Collection de colonnes de la list box.

Grammaire JSON

Nom	Type de données	Valeurs possibles
columns	collection d'objets colonne	Contient les propriétés des colonnes de list box

For a list of properties supported by column objects, please refer to the [Column Specific Properties](#) section.

Objets pris en charge

[List Box](#)

Nom formulaire détaillé

`Liste box sélection`

Indique le formulaire à utiliser pour modifier ou afficher les enregistrements individuels de la list box.

The specified form is displayed:

- when using `Add Subrecord` and `Edit Subrecord` standard actions applied to the list box (see [Using standard actions](#)),
- when a row is double-clicked and the `Double-click on Row` property is set to "Edit Record" or "Display Record".

Grammaire JSON

Nom	Type de données	Valeurs possibles
detailForm	string	<ul style="list-style-type: none">• Name (string) of table or project form• POSIX path (string) to a .json file describing the form• Object describing the form

Objets pris en charge

[List Box](#)

Double-clic sur ligne

`Liste box sélection`

Sets the action to be performed when a user double-clicks on a row in the list box. The available options are:

- Do nothing (default): Double-clicking a row does not trigger any automatic action.
- Edit Record: Double-clicking a row displays the corresponding record in the detail form defined [for the list box](#). The record is opened in read-write mode so it can be modified.
- Display Record: Identical to the previous action, except that the record is opened in read-only mode so it cannot be

modified.

Double-clicking an empty row is ignored in list boxes.

Regardless of the action selected/chosen, the `On Double clicked` form event is generated.

For the last two actions, the `On Open Detail` form event is also generated. The `On Close Detail` is then generated when a record displayed in the detail form associated with the list box is about to be closed (regardless of whether or not the record was modified).

Grammaire JSON

Nom	Type de données	Valeurs possibles
doubleClickInRowAction	string	"editSubrecord", "displaySubrecord"

Objets pris en charge

[List Box](#)

Ensemble surlignage

Liste box sélection

This property is used to specify the set to be used to manage highlighted records in the list box (when the `Arrays` data source is selected, a Boolean array with the same name as the list box is used).

4D creates a default set named `ListBoxSetN` where *N* starts at 0 and is incremented according to the number of list boxes in the form. If necessary, you can modify the default set. It can be a local, process or interprocess set (we recommend using a local set, for example `$LBSet`, in order to limit network traffic). It is then maintained automatically by 4D. If the user selects one or more rows in the list box, the set is updated immediately. If you want to select one or more rows by programming, you can apply the commands of the "Sets" theme to this set.

- The highlighted status of the list box rows and the highlighted status of the table records are completely independent.
- If the "Highlight Set" property does not contain a name, it will not be possible to make selections in the list box.

Grammaire JSON

Nom	Type de données	Valeurs possibles
highlightSet	string	Name of the set

Objets pris en charge

[List Box](#)

Locked columns and static columns

Locked columns and static columns are two separate and independent functionalities in list boxes:

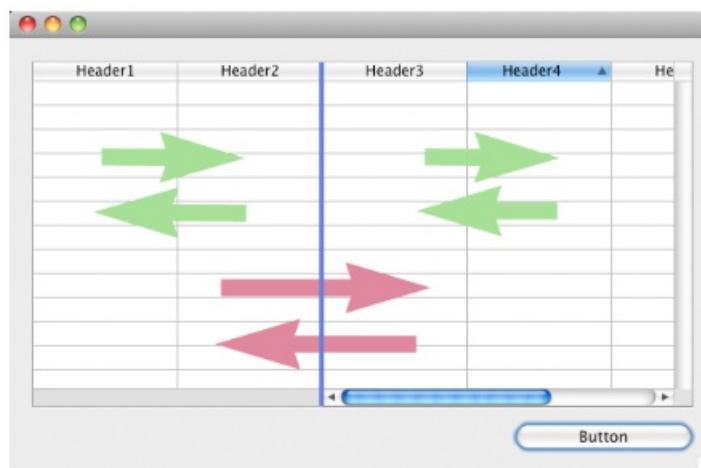
- Locked columns always stay displayed to the left of the list box; they do not scroll horizontally.

- Static columns cannot be moved by drag and drop within the list box.

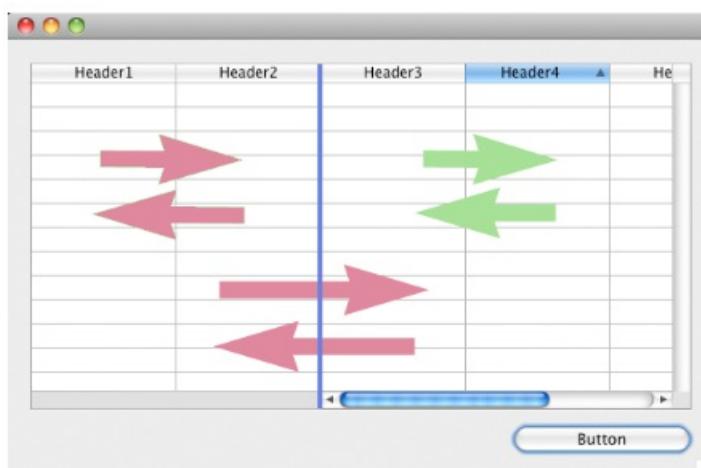
You can set static and locked columns by programming, refer to [List Box](#) in the [4D Language Reference](#) manual.

These properties interact as follows:

- If you set columns that are only static, they cannot be moved.
- If you set columns that are locked but not static, you can still change their position freely within the locked area. However, a locked column cannot be moved outside of this locked area.



- If you set all of the columns in the locked area as static, you cannot move these columns within the locked area.



- You can set a combination of locked and static columns according to your needs. For example, if you set three locked columns and one static column, the user can swap the two right-most columns within the locked area (since only the first column is static).

Nombre de colonnes verrouillées

Number of columns that must stay permanently displayed in the left part of the list box, even when the user scrolls through the columns horizontally.

Grammaire JSON

Nom	Type de données	Valeurs possibles
lockedColumnCount	entier	minimum : 0

Objets pris en charge

List Box

Nombre de colonnes statiques

Number of columns that cannot be moved during execution.

Grammaire JSON

Nom	Type de données	Valeurs possibles
staticColumnCount	entier	minimum : 0

Objets pris en charge

List Box

Nombre de colonnes

Sets the number of columns of the list box.

You can add or remove columns dynamically by programming, using commands such as [LISTBOX INSERT COLUMN](#) or [LISTBOX DELETE COLUMN](#).

Grammaire JSON

Nom	Type de données	Valeurs possibles
columnCount	entier	minimum: 1

Objets pris en charge

List Box

Tableau de contrôle des lignes

Array type list box

A 4D array controlling the display of list box rows.

You can set the "hidden", "disabled" and "selectable" interface properties for each row in an array-based list box using this array. It can also be designated using the [LISTBOX SET ARRAY](#) command.

The row control array must be of the Longint type and include the same number of rows as the list box. Each element of the *Row Control Array* defines the interface status of its corresponding row in the list box. Three interface properties are available using constants in the "List Box" constant theme:

Constante	Valeur	Commentaire
lk row is disabled	2	The corresponding row is disabled. The text and controls such as check boxes are dimmed or grayed out. Enterable text input areas are no longer enterable. Default value: Enabled
lk row is hidden	1	The corresponding row is hidden. Hiding rows only affects the display of the list box. The hidden rows are still present in the arrays and can be managed by programming. The language commands, more particularly LISTBOX Get number of rows or LISTBOX GET CELL POSITION, do not take the displayed/hidden status of rows into account. For example, in a list box with 10 rows where the first 9 rows are hidden, LISTBOX Get number of rows returns 10. From the user's point of view, the presence of hidden rows in a list box is not visibly discernible. Only visible rows can be selected (for example using the Select All command). Default value: Visible
lk row is not selectable	4	The corresponding row is not selectable (highlighting is not possible). Enterable text input areas are no longer enterable unless the Single-Click Edit option is enabled. Controls such as check boxes and lists are still functional however. This setting is ignored if the list box selection mode is "None". Default value: Selectable

To change the status for a row, you just need to set the appropriate constant(s) to the corresponding array element. For example, if you do not want row #10 to be selectable, you can write:

```
aLControlArr{10}:=lk row is not selectable
```

RowNum	Countries	Population	Landlocked
1	Luxembourg	0 502 202	<input checked="" type="checkbox"/>
2	Latvia	1 973 700	<input type="checkbox"/>
3	Kuwait	4 044 500	<input type="checkbox"/>
4	Croatia	4 284 889	<input type="checkbox"/>
5	Denmark	5 699 220	<input type="checkbox"/>
6	Nicaragua	6 071 045	<input type="checkbox"/>
7	Czech Republic	10 674 947	<input checked="" type="checkbox"/>
8	Serbia	7 306 677	<input checked="" type="checkbox"/>
9	Honduras	8 249 574	<input type="checkbox"/>
10	Austria	8 572 895	<input checked="" type="checkbox"/>
11	Hungary	10 005 000	<input checked="" type="checkbox"/>
12	Greece	10 815 197	<input type="checkbox"/>
13	Benin	10 879 829	<input type="checkbox"/>

You can define several interface properties at once:

```
aLControlArr{8}:=lk row is not selectable + lk row is disabled
```

RowNum	Countries	Population	Landlocked
1	Luxembourg	0 502 202	<input checked="" type="checkbox"/>
2	Latvia	1 973 700	<input type="checkbox"/>
3	Kuwait	4 044 500	<input type="checkbox"/>
4	Croatia	4 284 889	<input type="checkbox"/>
5	Denmark	5 699 220	<input type="checkbox"/>
6	Nicaragua	6 071 045	<input type="checkbox"/>
7	Czech Republic	10 674 947	<input checked="" type="checkbox"/>
8	Serbia	7 306 677	<input checked="" type="checkbox"/>
9	Honduras	8 249 574	<input type="checkbox"/>
10	Austria	8 572 895	<input checked="" type="checkbox"/>
11	Hungary	10 005 000	<input checked="" type="checkbox"/>
12	Greece	10 815 197	<input type="checkbox"/>
13	Benin	10 879 829	<input type="checkbox"/>

Note that setting properties for an element overrides any other values for this element (if not reset). Par exemple :

```

aLControlArr{6}:=lk row is disabled + lk row is not selectable
//sets row 6 as disabled AND not selectable
aLControlArr{6}:=lk row is disabled
//sets row 6 as disabled but selectable again

```

Grammaire JSON

Nom	Type de données	Valeurs possibles
rowControlSource	string	Row control array name

Objets pris en charge

[List Box](#)

Mode de sélection

Designates the option for allowing users to select rows:

- None: Rows cannot be selected if this mode is chosen. Clicking on the list will have no effect unless the [Single-Click Edit](#) option is enabled. The navigation keys only cause the list to scroll; the [On Selection Change](#) form event is not generated.
- Single: One row at a time can be selected in this mode. Clicking on a row will select it. A [Ctrl+click](#) (Windows) or [Command+click](#) (macOS) on a row toggles its state (between selected or not). The Up and Down arrow keys select the previous/next row in the list. The other navigation keys scroll the list. The [On Selection Change](#) form event is generated every time the current row is changed.
- Multiple: Several rows can be selected simultaneously in this mode.

Grammaire JSON

Nom	Type de données	Valeurs possibles
selectionMode	string	"multiple", "single", "none"

Objets pris en charge

[List Box](#)

Objets

Type

PARAMETRAGE OBLIGATOIRE

This property designates the type of the [active or inactive form object](#).

Grammaire JSON

Nom	Type de données	Valeurs possibles
type	string	"button", "buttonGrid", "checkbox", "combo", "dropdown", "groupBox", "input", "line", "list", "listbox", "oval", "picture", "pictureButton", "picturePopup", "plugin", "progress", "radio", "rectangle", "ruler", "spinner", "splitter", "stepper", "subform", "tab", "text", "view", "webArea", "write"

Objets pris en charge

4D View Pro area - 4D Write Pro area - Button - Button Grid - Check Box - Combo Box - Drop-down List - Group Box - Hierarchical List - List Box - List Box Column - List Box Footer - List Box Header - Picture Button - Picture Pop-up Menu - Plug-in Area - Progress indicator - Radio Button - Spinner - Splitter - Static Picture - Stepper - Subform - Tab control - Text Area - Web Area

Nom d'objet

Each active form object is associated with an object name. Each object name must be unique.

Object names are limited to a size of 255 bytes.

When using 4D's language, you can refer to an active form object by its object name (for more information about this, refer to [Object Properties](#) in the 4D Language Reference manual).

For more information about naming rules for form objects, refer to [Identifiers](#) section.

Grammaire JSON

Nom	Type de données	Valeurs possibles
name	string	Any allowed name which does not belong to an already existing object

Objets pris en charge

4D View Pro area - 4D Write Pro area - Button - Button Grid - Check Box - Combo Box - Drop-down List - Group Box - Hierarchical List - List Box - List Box Column - List Box Footer - List Box Header - Picture Button - Picture Pop-up Menu - Plug-in Area - Progress indicator - Spinner - Splitter - Static Picture - Stepper - Radio Button - Subform - Tab control - Text Area - Web Area

Save value

This property is available when the [Save Geometry](#) option is checked for the form.

This feature is only supported for objects that contribute to the overall geometry of the form. For example, this option is available for check boxes because their value can be used to hide or display additional areas in the window.

Here is the list of objects whose value can be saved:

Object	Saved value
Case à cocher	Value of associated variable (0, 1, 2)
Liste déroulante	Number of selected row
Bouton radio	Value of associated variable (1, 0, True or False for buttons according to their type)
Tab control	Number of selected tab

Grammaire JSON

Nom	Type de données	Valeurs possibles
memorizeValue	boolean	true, false

Objets pris en charge

[Check Box](#) - [Drop-down List](#) - [Radio Button](#) - [Tab control](#)

Variable ou expression

Voir également [Expression](#) pour les colonnes de list box de type sélection et collection.

This property specifies the source of the data. Each active form object is associated with an object name and a variable name. The variable name can be different from the object's name. In the same form, you can use the same variable several times while each [object name](#) must be unique.

Variable name size is limited to 31 bytes. See [Identifiers](#) section for more information about naming rules.

The form object variables allow you to control and monitor the objects. For example, when a button is clicked, its variable is set to 1; at all other times, it is 0. The expression associated with a progress indicator lets you read and change the current setting.

Variables or expressions can be enterable or non-enterable and can receive data of the Text, Integer, Numeric, Date, Time, Picture, Boolean, or Object type.

Grammaire JSON

Nom	Type de données	Valeurs possibles
dataSource	string, or string array	<ul style="list-style-type: none">• Variable, nom de champ, ou toute expression 4D.• Empty string for dynamic variables.• String array (collection of array names) for a hierarchical listbox column]

Expressions

You can use an [expression](#) as the data source for an object. Any valid 4D expression is allowed: simple expression, object property, formula, 4D function, project method name or field name using standard syntax [Table]Field . The expression is evaluated when the form is executed and reevaluated for each form event. Note that expressions can be [assignable](#) or [non-assignable](#).

If the value entered corresponds to both a variable name and a method name, 4D considers that you are indicating the method.

Dynamic variables

You can leave it up to 4D to create variables associated with your form objects (buttons, enterable variables, check boxes, etc.) dynamically and according to your needs. To do this, simply leave the "Variable or Expression" property (or `dataSource` JSON field) blank.

When a variable is not named, when the form is loaded, 4D creates a new variable for the object, with a calculated name that is unique in the space of the process variables of the interpreter (which means that this mechanism can be used even in compiled mode). This temporary variable will be destroyed when the form is closed. In order for this principle to work in compiled mode, it is imperative that dynamic variables are explicitly typed. There are two ways to do this:

- You can set the type using the [Expression type](#) property.
- You can use a specific initialization code when the form is loaded that uses, for example, the `VARIABLE TO VARIABLE` command:

```
If(Form event code=On Load)
  var $init : Text
  $Ptr_object:=OBJECT Get pointer(Object named;"comments")
  $init:=""
  VARIABLE TO VARIABLE(Current process;$Ptr_object->:$init)
End if
```

In the 4D code, dynamic variables can be accessed using a pointer obtained with the `OBJECT Get pointer` command.
Par exemple :

```
// assign the time 12:00:00 to the variable for the "tstart" object
$p :=OBJECT Get pointer(Object named;"tstart")
$p->:=?12:00:00?
```

There are two advantages with this mechanism:

- On the one hand, it allows the development of "subform" type components that can be used several times in the same host form. Let us take as an example the case of a datepicker subform that is inserted twice in a host form to set a start date and an end date. This subform will use objects for choosing the date of the month and the year. It will be necessary for these objects to work with different variables for the start date and the end date. Letting 4D create their variable with a unique name is a way of resolving this difficulty.
- On the other hand, it can be used to limit memory usage. In fact, form objects only work with process or inter-process variables. However, in compiled mode, an instance of each process variable is created in all the processes, including the server processes. This instance takes up memory, even when the form is not used during the session. Therefore, letting 4D create variables dynamically when loading the forms can save memory.

Array List Box

For an array list box, the Variable or Expression property usually holds the name of the array variable defined for the list box, and for each column. However, you can use a string array (containing arrays names) as `dataSource` value for a list box column to define a [hierarchical list box](#).

Objets pris en charge

[Zone 4D View Pro](#) - [Zone 4D Write Pro](#) - [Bouton](#) - [Grille de boutons](#) - [Case à cocher](#) - [Combo Box](#) - [Liste déroulante](#) - [Liste hiérarchique](#) - [List Box](#) - [Colonne List Box](#) - [En-tête list box](#) - [Pied List Box](#) - [Pop-up Menu Image](#) - [Plug-in Area](#) - [Indicateur de progression](#) - [Bouton Radio](#) - [Spinner](#) - [Splitter](#) - [Stepper](#) - [Sous-formulaire](#) - [Onglet](#) - [Zone Web](#)

Expression Type

Cette propriété est intitulée [Data Type](#) dans la Liste de Propriétés pour les colonnes de listbox de type [selection](#) et [collection](#) et pour les [Listes déroulantes](#) associées à un [objet](#) ou à un [tableau](#).

Specify the data type for the expression or variable associated to the object. Note that main purpose of this setting is to configure options (such as display formats) available for the data type. It does not actually type the variable itself. En vue d'une compilation de projet, vous devez [déclarer la variable](#).

However, this property has a typing function in the following specific cases:

- [Dynamic variables](#): you can use this property to declare the type of dynamic variables.
- [List Box Columns](#): this property is used to associate a display format with the column data. The formats provided will depend on the variable type (array type list box) or the data/field type (selection and collection type list boxes). The standard 4D formats that can be used are: Alpha, Numeric, Date, Time, Picture and Boolean. The Text type does not have specific display formats. Any existing custom formats are also available.
- [Picture variables](#): you can use this menu to declare the variables before loading the form in interpreted mode. Specific native mechanisms govern the display of picture variables in forms. These mechanisms require greater precision when configuring variables: from now on, they must have already been declared before loading the form — i.e., even before the `On Load` form event — unlike other types of variables. To do this, you need either for the statement `C_PICTURE(varName)` to have been executed before loading the form (typically, in the method calling the `DIALOG` command), or for the variable to have been typed at the form level using the expression type property. Otherwise, the picture variable will not be displayed correctly (only in interpreted mode).

Grammaire JSON

Nom	Type de données	Valeurs possibles
dataSourceTypeHint	string	<ul style="list-style-type: none">standard objects: "integer", "boolean", "number", "picture", "text", "date", "time", "arrayText", "arrayDate", "arrayTime", "arrayNumber", "collection", "object", "undefined"colonnes de list box : "boolean", "number", "picture", "text", "date", "time". <i>Listbox tableau/selection uniquement</i> : "integer", "object"

Objets pris en charge

[Check Box](#) - [Combo Box](#) - [Drop-down List](#) - [Input](#) - [List Box Column](#) - [List Box Footer](#) - [Plug-in Area](#) - [Progress indicator](#) - [Radio Button](#) - [Ruler](#) - [Spinner](#) - [Stepper](#) - [Subform](#) - [Tab Control](#)

CSS Class

A list of space-separated words used as class selectors in [css files](#).

Grammaire JSON

Nom	Type de données	Valeurs possibles
class	string	One string with CSS name(s) separated by space characters

Objets pris en charge

[4D View Pro area](#) - [4D Write Pro area](#) - [Button](#) - [Button Grid](#) - [Check Box](#) - [Combo Box](#) - [Drop-down List](#) - [Group Box](#) - [Hierarchical List](#) - [List Box](#) - [Picture Button](#) - [Picture Pop-up Menu](#) - [Plug-in Area](#) - [Radio Button](#) - [Static Picture](#) - [Subform](#) - [Text Area](#) - [Web Area](#)

Collection ou entity selection

To use collection elements or entities to define the row contents of the list box.

Enter an expression that returns either a collection or an entity selection. Usually, you will enter the name of a variable, a collection element or a property that contain a collection or an entity selection.

The collection or the entity selection must be available to the form when it is loaded. Each element of the collection or each entity of the entity selection will be associated to a list box row and will be available as an object through the [This](#) command:

- if you used a collection of objects, you can call This in the datasource expression to access each property value, for example This.<propertyPath>.
- if you used an entity selection, you can call This in the datasource expression to access each attribute value, for example This.<attributePath>.

If you used a collection of scalar values (and not objects), 4D allows you to display each value by calling This.value in the datasource expression. However in this case you will not be able to modify values or to access the current item object (see below) Note: For information about entity selections, please refer to the [ORDA](#) chapter.

Grammaire JSON

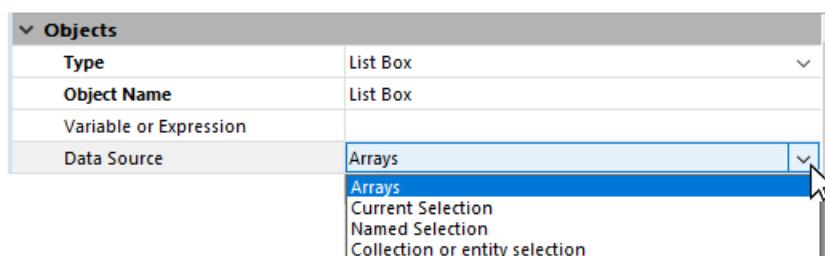
Nom	Type de données	Valeurs possibles
dataSource	string	Expression that returns a collection or an entity selection.

Objets pris en charge

List Box

Source de données

Specify the type of list box.



- Arrays(default): use array elements as the rows of the list box.
- Current Selection: use expressions, fields or methods whose values will be evaluated for each record of the current selection of a table.
- Named Selection: use expressions, fields or methods whose values will be evaluated for each record of a named selection.
- Collection or Entity Selection: use collection elements or entities to define the row contents of the list box. Note that with this list box type, you need to define the [Collection or Entity Selection](#) property.

Grammaire JSON

Nom	Type de données	Valeurs possibles
listboxType	string	"array", "currentSelection", "namedSelection", "collection"

Objets pris en charge

[List Box](#)

Plug-in Kind

Name of the [plug-in external area](#) associated to the object. Plug-in external area names are published in the manifest.json file of the plug-in.

Grammaire JSON

Nom	Type de données	Valeurs possibles
pluginAreaKind	string	Name of the plug-in external area (starts with a % character)

Objets pris en charge

[Zone de plug-in](#)

Radio Group

Enables radio buttons to be used in coordinated sets: only one button at a time can be selected in the set.

Grammaire JSON

Nom	Type de données	Valeurs possibles
radioGroup	string	Radio group name

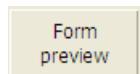
Objets pris en charge

[Bouton radio](#)

Titre de menu

Allows inserting a label on an object. The font and the style of this label can be specified.

You can force a carriage return in the label by using the ¥ character (backslash).



To insert a ¥ in the label, enter "¥¥".

By default, the label is placed in the center of the object. When the object also contains an icon, you can modify the relative location of these two elements using the [Title/Picture Position](#) property.

For application translation purposes, you can enter an XLIFF reference in the title area of a button (see [Appendix B: XLIFF architecture](#)).

Grammaire JSON

Nom	Type de données	Valeurs possibles
Texte	string	any text

Objets pris en charge

[Button](#) - [Check Box](#) - [List Box Header](#) - [Radio Button](#) - [Text Area](#)

Variable Calculation

This property sets the type of calculation to be done in a [column footer](#) area.

The calculation for footers can also be set using the [LISTBOX SET FOOTER CALCULATION](#) 4D command.

There are several types of calculations available. The following table shows which calculations can be used according to the type of data found in each column and indicates the type automatically affected by 4D to the footer variable (if it is not typed by the code):

Calculation	Num	Text	Date	Heure	Bool	Pict	footer var type
Minimum	X	X	X	X	X		Same as column type
Maximum	X	X	X	X	X		Same as column type
Sum	X			X	X		Same as column type
Count	X	X	X	X	X	X	Longint
Average	X			X			Réel
Standard deviation(*)	X			X			Réel
Variance(*)	X			X			Réel
Sum squares(*)	X			X			Réel
Custom ("none")	X	X	X	X	X	X	Any

(*) Only for array type list boxes.

Seules les [variables](#) déclarées ou dynamiques peuvent être utilisées pour afficher les calculs des pieds de listbox.
Les autres types d'[expressions](#) telles que [Form.sortValue](#) ne sont pas pris en charge.

Automatic calculations ignore the shown/hidden state of list box rows. If you want to restrict a calculation to only visible rows, you must use a custom calculation.

Null values are not taken into account for any calculations.

If the column contains different types of values (collection-based column for example):

- Average and Sum only take numerical elements into account (other element types are ignored).
- Minimum and Maximum return a result according to the usual type list order as defined in the [collection.sort\(\)](#) function.

Using automatic calculations in footers of columns based upon expressions has the following limitations:

- it is supported with all list box types when the expression is "simple" (such as [\[table\]field](#) or [this.attribute](#)),
- it is supported but not recommended for performance reasons with collection/entity selection list boxes when the expression is "complex" (other than [this.attribute](#)) and the list box contains a large number of rows,
- it is not supported with current selection/named selection list boxes when the expression is "complex". You need to use custom calculations.

When Custom ("none" in JSON) is set, no automatic calculations are performed by 4D and you must assign the value of

the variable in this area by programming.

Grammaire JSON

Nom	Type de données	Valeurs possibles
variableCalculation	string	"none", "minimum", "maximum", "sum", "count", "average", "standardDeviation", "variance", "sumSquare"

Objets pris en charge

[List Box Footer](#)

Image

Chemin d'accès

Chemin d'une image source statique pour un [bouton image](#), [un menu pop-up](#) ou une [image ou une image statique](#). Vous devez utiliser la syntaxe POSIX.

Les emplacements suivants peuvent être utilisés pour le chemin d'images statiques :

- in the Resources folder of the project. Appropriate when you want to share static pictures between several forms in the project. Dans ce cas, le chemin d'accès se trouve dans "/RESOURCES/<picture path>".
- dans un dossier d'images (nommé Images par exemple) dans le dossier du formulaire. Convient lorsque les images statiques sont utilisées uniquement dans le formulaire et/ou lorsque vous souhaitez pouvoir déplacer ou dupliquer le formulaire entier dans un ou plusieurs projets. Dans ce cas, le chemin d'accès est "<picture path>" et est déterminé à la racine du dossier du formulaire.
- dans une variable image 4D. L'image doit être chargée en mémoire lors de l'exécution du formulaire. Dans ce cas, le nom de chemin est "var:<variableName>".

Grammaire JSON

Nom	Type de données	Valeurs possibles
picture	Texte	Chemin relatif ou chemin du système de fichiers dans la syntaxe POSIX, ou "var:<variableName>" pour la variable image

Objets pris en charge

[Bouton image](#) - [Pop-up Menu image](#) - [Image statique](#)

Affichage

Scaled to fit

JSON grammar: "scaled"

The Scaled to fit format causes 4D to resize the picture to fit the dimensions of the area.

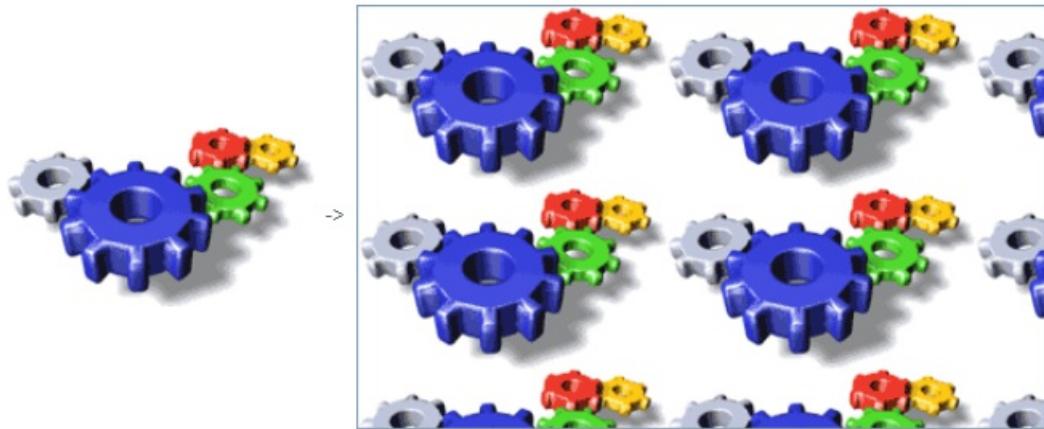


Replicated

JSON grammar: "tiled"

When the area that contains a picture with the Replicated format is enlarged, the picture is not deformed but is

replicated as many times as necessary in order to fill the area entirely.



If the field is reduced to a size smaller than that of the original picture, the picture is truncated (non-centered).

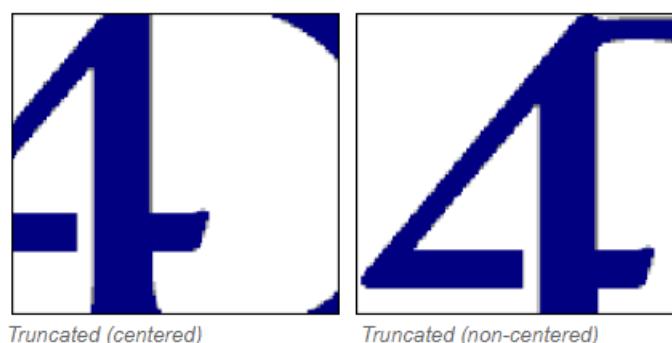
Centre / Tronquée (non centrée)

JSON grammar: "truncatedCenter" / "truncatedTopLeft"

Le format Centre permet à 4D de centrer l'image dans la zone et de rogner toute partie qui ne rentre pas dans la zone. 4D crops equally from each edge and from the top and bottom.

The Truncated (non-centered) format causes 4D to place the upper-left corner of the picture in the upper-left corner of the area and crop any portion that does not fit within the area. 4D crops from the right and bottom.

When the picture format is Truncated (non-centered), it is possible to add scroll bars to the input area.



Grammaire JSON

Nom	Type de données	Valeurs possibles
pictureFormat	string	"scaled", "tiled", "truncatedCenter", "truncatedTopLeft"

Objets pris en charge

[Image statique](#)

Plug-ins

Propriétés avancées

Si des options avancées sont fournies par l'auteur du plug-in, un bouton Propriétés avancées peut être activé dans la liste de propriétés. Dans ce cas, vous pouvez cliquer sur ce bouton pour définir ces options, généralement via une boîte de dialogue personnalisée.

Étant donné que la fonction Propriétés avancées est sous le contrôle de l'auteur du plug-in, les informations sur ces options avancées relèvent de la responsabilité du distributeur du plug-in.

Grammaire JSON

Nom	Type de données	Valeurs possibles
customProperties	Texte	Propriétés spécifiques du plug-in, passées au plug-in sous forme de chaîne JSON s'il s'agit d'un objet ou sous forme de tampon binaire s'il s'agit d'une chaîne encodée en base64

Objets pris en charge

[Zone de plug-in](#)

Imprimer

Impression cadre

Cette propriété permet de gérer le mode d'impression des objets dont la taille peut varier d'un enregistrement à l'autre en fonction de leur contenu. Ces objets peuvent être imprimés sur une hauteur de taille fixe ou variable. Un cadre de taille fixe provoque l'impression de l'objet dans les limites définies lors de la création de l'objet dans le formulaire. Un cadre de taille variable s'étend si nécessaire lors de l'impression afin d'imprimer l'intégralité de l'objet. A noter que la largeur des objets imprimés en taille variable n'est pas affectée par cette option; seule la hauteur varie automatiquement en fonction du contenu de l'objet.

Vous ne pouvez pas placer deux objets (ou plus) avec une taille variable côté à côté dans un formulaire. Vous pouvez placer des objets de taille fixe à côté d'un objet qui sera imprimé avec une taille variable si l'objet de taille variable est plus long d'au moins une ligne que l'objet placé à son côté et que leurs limites supérieures sont alignées. Si cette condition n'est pas respectée, le contenu des autres champs sera répété pour toute tranche horizontale de l'objet de taille variable.

Les commandes `Print object` et `Print form` ne sont pas compatibles avec cette option.

Les options d'impression sont les suivantes :

- L'option Variable / Impression taille variable cochée : 4D agrandit ou réduit la zone de l'objet du formulaire afin d'imprimer tous les sous-enregistrements.
- L'option Fixe (Tronqué) / Impression taille variable non cochée : 4D imprime uniquement le contenu qui apparaît dans la zone de l'objet. Le formulaire n'est imprimé qu'une seule fois et le contenu non imprimé est ignoré.
- Fixe (Enregistrements multiples) (sous-formulaires uniquement) : la taille initiale de la zone de sous-formulaire est conservée mais 4D imprime le formulaire plusieurs fois afin d'imprimer tous les enregistrements.

Cette propriété peut être définie par programmation à l'aide de la commande `OBJECT SET PRINT VARIABLE FRAME`.

Grammaire JSON

Nom	Type de données	Valeurs possibles
<code>printFrame</code>	string	"fixed", "variable", (sous-formulaire uniquement) "fixedMultiple"

Objets pris en charge

[Zone de saisie](#) - [Sous-formulaires](#) (sous-formulaires liste uniquement) - [Zones 4D Write Pro](#)

Plage de valeurs

Valeur par défaut

Vous pouvez attribuer une valeur par défaut à saisir dans un objet Zone de saisie. Cette propriété est utile par exemple lorsque la [source de données](#) de la zone de saisie est un champ : la valeur par défaut est saisie lors du premier affichage d'un nouvel enregistrement. Vous pouvez modifier la valeur, sauf si la zone de saisie a été définie comme [non saisissable](#).

La valeur par défaut ne peut être utilisée que si le [type de source de données](#) est :

- text/string
- number/integer
- date
- time
- boolean

4D provides stamps for generating default values for the date, time, and sequence number. The date and time are taken from the system date and time. 4D automatically generates any sequence numbers needed. The table below shows the stamp to use to generate default values automatically:

Stamp	Meaning
#D	Current date
#H	Current time
#N	Sequence number

You can use a sequence number to create a unique number for each record in the table for the current data file. A sequence number is a longint that is generated for each new record. The numbers start at one (1) and increase incrementally by one (1). A sequence number is never repeated even if the record it is assigned to is deleted from the table. Each table has its own internal counter of sequence numbers. For more information, refer to the [Autoincrement](#) paragraph.

Do not make confusion between this property and the "[default values](#)" property that allows to fill a list box column with static values.

Grammaire JSON

Nom	Type de données	Valeurs possibles
defaultValue	string, number, date, time, boolean	Any value and/or a stamp: "#D", "#H", "#N"

Objets pris en charge

[Input](#)

Excluded List

Allows setting a list whose values cannot be entered in the object. Si une valeur exclue est saisie, elle n'est pas acceptée et un message d'erreur s'affiche.

If a specified list is hierarchical, only the items of the first level are taken into account.

Grammaire JSON

Nom	Type de données	Valeurs possibles
excludedList	liste	Une liste de valeurs à exclure.

Objets pris en charge

[Combo Box](#) - [List Box Column](#) - [Input](#)

Required List

Limite les entrées valides aux éléments de la liste. Par exemple, si vous souhaitez utiliser une liste pour les titres de postes, afin que les entrées valides soient limitées aux titres qui ont été approuvés par la direction.

La création d'une liste obligatoire n'affiche pas automatiquement la liste lorsque le champ est sélectionné. Si vous souhaitez afficher la liste requise, assignez la même liste à la propriété [Choice List](#). Cependant, contrairement à la propriété [Choice List](#), lorsqu'une liste obligatoire est définie, la saisie au clavier n'est plus possible, seule la sélection d'une valeur de liste à l'aide du pop-up menu est autorisée. If different lists are defined using the [Choice List](#) and Required List properties, the Required List property has priority.

If a specified list is hierarchical, only the items of the first level are taken into account.

Grammaire JSON

Nom	Type de données	Valeurs possibles
requiredList	liste	Une liste de valeurs obligatoires.

Objets pris en charge

[Combo Box](#) - [List Box Column](#) - [Input](#)

Options de redimensionnement

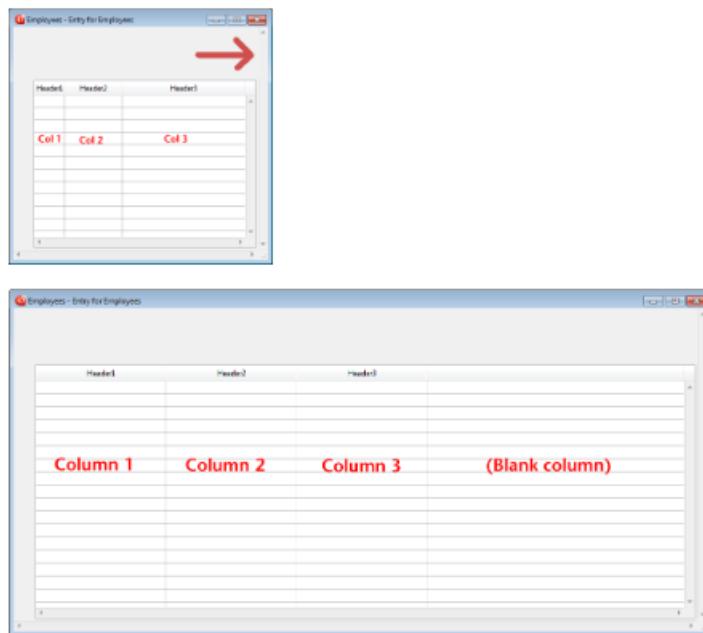
Redimensionnement colonnes auto

Lorsque cette propriété est activée (valeur `rightToLeft` dans JSON), les colonnes de listbox sont automatiquement redimensionnées en même temps que la listbox, dans les limites des largeurs `minimum` et `maximum` définies.

Lorsque cette propriété est désactivée (valeur `legacy` dans JSON), seule la colonne la plus à droite de la listbox est redimensionnée, même si sa largeur dépasse la valeur maximale définie.

Le redimensionnement automatique des colonnes

- À mesure que la largeur de la listbox augmente, ses colonnes sont agrandies une par une, en partant de la droite vers la gauche, jusqu'à ce que chacune atteigne sa [largeur maximale](#). Seules les colonnes dont la propriété [Resizable](#) est sélectionnée sont redimensionnées.
- The same procedure applies when the list box width decreases, but in reverse order (*i.e.*, columns are resized starting from left to right). When each column has reached its [minimum width](#), the horizontal scroll bar becomes active again.
- Columns are resized only when the horizontal scroll bar is not "active"; *i.e.*, all columns are fully visible in the list box at its current size. Note: If the horizontal scroll bar is hidden, this does not alter its state: a scroll bar may still be active, even though it is not visible.
- After all columns reach their maximum size, they are no longer enlarged and instead a blank (fake) column is added on the right to fill the extra space. If a fake (blank) column is present, when the list box width decreases, this is the first area to be reduced.



About the fake (blank) column

The appearance of the fake column matches that of the existing columns; it will have a fake header and/or footer if these elements are present in the existing list box columns and it will have the same background color(s) applied.

The fake header and/or footer can be clicked but this does not have any effect on the other columns (e.g.: no sort is performed); nevertheless, the `On Clicked`, `On Header Click` and `On Footer Click` events are generated accordingly.

If a cell in the fake column is clicked, the [LISTBOX GET CELL POSITION](#) command returns "X+1" for its column number (where X is the number of existing columns).

Grammaire JSON

Nom	Type de données	Valeurs possibles
resizingMode	string	"rightToLeft", "legacy"

Objets pris en charge

[List Box](#)

Dimensionnement horizontal

Cette propriété indique si la taille horizontale d'un objet doit être déplacée ou redimensionnée lorsqu'un utilisateur redimensionne le formulaire. Elle peut également être définie dynamiquement par la commande de langage [OBJECT SET RESIZING OPTIONS](#).

Trois options sont disponibles :

Option	Valeur JSON	Résultat
Agrandir	"grow"	Le même pourcentage est appliqué à la largeur de l'objet lorsque l'utilisateur redimensionne la largeur de la fenêtre,
Déplacer	"move"	L'objet est déplacé vers la gauche ou vers la droite selon l'augmentation de la largeur lorsque l'utilisateur redimensionne la largeur de la fenêtre,
Aucun	"fixed"	L'objet reste stationnaire lorsque le formulaire est redimensionné

Cette propriété fonctionne avec la propriété [Dimensionnement vertical](#).

Grammaire JSON

Nom	Type de données	Valeurs possibles
sizingX	string	"grow", "move", "fixed"

Objets pris en charge

[Zone 4D View Pro](#) - [Zone 4D Write Pro](#) - [Bouton](#) - [Grille de boutons](#) - [Case à cocher](#) - [Combo Box](#) - [Liste déroulante](#) - [Group Box](#) - [Liste hiérarchique](#) - [Zone de saisie](#) - [List Box](#) - [Ligne](#) - [Colonne List Box](#) - [Ovale](#) - [Bouton image](#) - [Pop up menu image](#) - [Zone de plug-in](#) - [Indicateur de progression](#) - [Bouton radio](#) - [Règle](#) - [Rectangle](#) - [Spinner](#) - [Splitter](#) - [Image statique Stepper](#) - [Sous-formulaire](#) - [Onglet](#) - [Zone de texte](#)

Dimensionnement vertical

Cette propriété indique si la taille verticale d'un objet doit être déplacée ou redimensionnée lorsqu'un utilisateur redimensionne le formulaire. Elle peut également être définie dynamiquement par la commande de langage [OBJECT SET RESIZING OPTIONS](#).

Trois options sont disponibles :

Option	Valeur JSON	Résultat
Agrandir	"grow"	Le même pourcentage est appliqué à la hauteur de l'objet lorsque l'utilisateur redimensionne la largeur de la fenêtre,
Déplacer	"move"	L'objet est déplacé vers le haut ou vers le bas selon l'augmentation de la hauteur lorsque l'utilisateur redimensionne la largeur de la fenêtre,
Aucun	"fixed"	L'objet reste stationnaire lorsque le formulaire est redimensionné

Cette propriété fonctionne avec la propriété [Dimensionnement horizontal](#).

Grammaire JSON

Nom	Type de données	Valeurs possibles
sizingY	string	"grow", "move", "fixed"

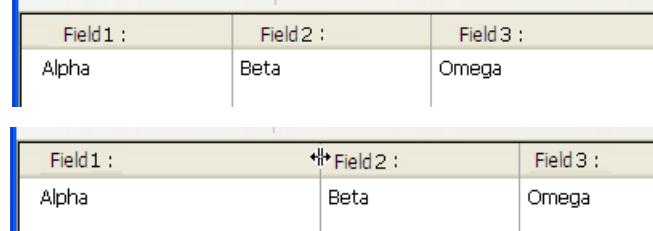
Objets pris en charge

[Zone 4D View Pro](#) - [Zone 4D Write Pro](#) - [Bouton](#) - [Grille de boutons](#) - [Case à cocher](#) - [Combo Box](#) - [Liste déroulante](#) - [Group Box](#) - [Liste hiérarchique](#) - [Zone de saisie](#) - [List Box](#) - [Ligne](#) - [Colonne List Box](#) - [Ovale](#) - [Bouton image](#) - [Pop up menu image](#) - [Zone de plug-in](#) - [Indicateur de progression](#) - [Bouton radio](#) - [Règle](#) - [Rectangle](#) - [Spinner](#) - [Splitter](#) - [Image statique Stepper](#) - [Sous-formulaire](#) - [Onglet](#) - [Zone de texte](#)

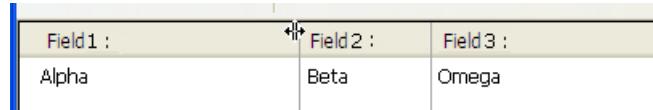
Pousseur

Lorsqu'un objet splitter a cette propriété, les autres objets à sa droite (splitter vertical) ou en dessous (splitter horizontal) sont poussés en même temps que le splitter, sans arrêt.

Voici le résultat du déplacement d'un splitter «pousseur» :



Lorsque cette propriété n'est pas appliquée au splitter, le résultat est le suivant :



Grammaire JSON

Nom	Type de données	Valeurs possibles
splitterMode	string	"move" (pusher), "resize" (standard)

Objets pris en charge

[Séparateur](#)

Redimensionnable

Indique si la taille de la colonne peut être modifiée par l'utilisateur.

Grammaire JSON

Nom	Type de données	Valeurs possibles
resizable	boolean	"true", "false"

Objets pris en charge

[Colonne de list box](#)

Echelle

Barber shop

Active la variante «barber shop» pour le thermomètre.

Grammaire JSON

Nom	Type de données	Valeurs possibles
max	number	NOT passed = enabled; passed = disabled (basic thermometer)

Objets pris en charge

[Barber shop](#)

Afficher graduation

Affiche/masque les graduations à côté des étiquettes.

Grammaire JSON

Nom	Type de données	Valeurs possibles
showGraduations	boolean	"true", "false"

Objets pris en charge

[Thermomètre - Règle](#)

Graduation step

Mesure de l'affichage de l'échelle.

Grammaire JSON

Nom	Type de données	Valeurs possibles
graduationStep	entier	minimum : 0

Objets pris en charge

[Thermomètre - Règle](#)

Label Location

Indique l'emplacement du texte d'un objet.

- Aucun - aucun libellé n'est affiché
- Haut - Affiche les libellés à gauche ou au-dessus d'un indicateur

- Bas - Affiche les libellés à droite ou en dessous d'un indicateur

Grammaire JSON

Nom	Type de données	Valeurs possibles
labelsPlacement	string	"none", "top", "bottom", "left", "right"

Objets pris en charge

[Thermomètre - Règle](#)

Maximum

Valeur maximale d'un indicateur.

- Pour les steppers numériques, cette propriété représente les secondes lorsque l'objet est associé à une valeur de type heure, et représente les jours lorsqu'il est associé à une valeur de type date.
- Pour activer les [thermomètres du barber shop](#), cette propriété doit être omise.

Grammaire JSON

Nom	Type de données	Valeurs possibles
max	chaîne / numérique	minimum: 0 (pour les types de données numériques)

Objets pris en charge

[Thermomètre - Règle - Stepper](#)

Minimum

Valeur minimale d'un indicateur. Pour les steppers numériques, cette propriété représente les secondes lorsque l'objet est associé à une valeur de type heure, et représente les jours lorsqu'il est associé à une valeur de type date.

Grammaire JSON

Nom	Type de données	Valeurs possibles
min	chaîne / numérique	minimum: 0 (pour les types de données numériques)

Objets pris en charge

[Thermomètre - Règle - Stepper](#)

Step

Intervalle minimum accepté entre les valeurs pendant l'utilisation. Pour les steppers numériques, cette propriété représente les secondes lorsque l'objet est associé à une valeur de type heure et représente les jours lorsqu'il est associé à une valeur de type date.

Grammaire JSON

Nom	Type de données	Valeurs possibles
step	entier	minimum: 1

Objets pris en charge

[Thermomètre](#) - [Règle](#) - [Stepper](#)

Sous-formulaire

Autoriser la suppression

Indique si l'utilisateur peut supprimer des sous-enregistrements dans un sous-formulaire liste.

Grammaire JSON

Nom	Type de données	Valeurs possibles
deletableInList	boolean	true, false (par défaut : true)

Objets pris en charge

[Sous-formulaire](#)

Formulaire détaillé

Cette option permet de désigner un formulaire détaillé à utiliser dans un sous-formulaire. Il peut être :

- un widget, c'est-à-dire un sous-formulaire de type page doté de fonctions spécifiques. Dans ce cas, [le sous-formulaire de liste](#) et les propriétés [Source](#) doivent être vides ou non présents.
Vous pouvez sélectionner un nom de formulaire de composant lorsqu'il est publié dans le composant.

Pour cela, il vous suffit de cliquer deux fois sur le champ à modifier afin de le passer en mode édition (veillez à laisser suffisamment de temps entre les deux clics pour ne pas générer de double-clic).

- le formulaire détaillé à associer au [sous-formulaire de liste](#). Le formulaire détaillé peut être utilisé pour saisir ou afficher des sous-enregistrements. Il contient généralement plus d'informations que le sous-formulaire liste. Naturellement, le formulaire détaillé doit appartenir à la même table que le sous-formulaire. Vous utilisez normalement un formulaire de sortie comme formulaire liste et un formulaire d'entrée comme formulaire détaillé. Si vous n'indiquez pas le formulaire à utiliser pour la saisie pleine page, 4D utilise automatiquement le format d'entrée par défaut de la table.

Grammaire JSON

Nom	Type de données	Valeurs possibles
detailForm	string	Name (string) of table or project form, a POSIX path (string) to a .json file describing the form, or an object describing the form

Objets pris en charge

[Sous-formulaire](#)

Double-clic sur ligne vide

Action to perform in case of a double-click on an empty line of a list subform. Les options suivantes sont disponibles :

- Ne rien faire : ignore le double-clic.

- Ajouter un enregistrement : crée un nouvel enregistrement dans le sous-formulaire et passe en mode édition. L'enregistrement sera créé directement dans la liste si la propriété [Saisissable dans la liste] est activée. Sinon, il sera créé en mode page, dans le [formulaire détaillé](#) associé au sous-formulaire.

Grammaire JSON

Nom	Type de données	Valeurs possibles
doubleClickInEmptyAreaAction	string	"addSubrecord" ou "" to do nothing

Objets pris en charge

[Sous-formulaire](#)

Voir aussi

[Double-clic sur ligne](#)

Double-clic sur ligne

Sous-formulaires liste

Définit l'action à réaliser lorsqu'un utilisateur double-clique sur une ligne dans un sous-formulaire liste. The available options are:

- Do nothing (default): Double-clicking a row does not trigger any automatic action.
- Modifier enregistrement : Un double-clic sur une ligne permet d'afficher l'enregistrement correspondant dans le [formulaire détaillé défini pour le sous-formulaire liste](#). The record is opened in read-write mode so it can be modified.
- Display Record: Identical to the previous action, except that the record is opened in read-only mode so it cannot be modified.

Regardless of the action selected/chosen, the `On Double clicked` form event is generated.

For the last two actions, the `On Open Detail` form event is also generated. The `On Close Detail` is then generated when a record displayed in the detail form associated with the list box is about to be closed (regardless of whether or not the record was modified).

Grammaire JSON

Nom	Type de données	Valeurs possibles
doubleClickInRowAction	string	"editSubrecord", "displaySubrecord"

Objets pris en charge

[Sous-formulaire](#)

Voir aussi

[Double-clic sur ligne vide](#)

Saisissable en liste

Lorsque cette propriété est activée pour un sous-formulaire de liste, l'utilisateur peut modifier les données de l'enregistrement directement dans la liste, sans avoir à utiliser le [formulaire détaillé associé](#).

Pour cela, il vous suffit de cliquer deux fois sur le champ à modifier afin de le passer en mode édition (veillez à

laisser suffisamment de temps entre les deux clics pour ne pas générer de double-clic).

Grammaire JSON

Nom	Type de données	Valeurs possibles
enterableInList	boolean	true, false

Objets pris en charge

[Sous-formulaire](#)

List Form

Cette option permet de désigner un formulaire liste à utiliser dans un sous-formulaire. Un sous-formulaire en liste vous permet de saisir, visualiser et modifier des données dans d'autres tables.

Les sous-formulaires de liste peuvent être utilisés pour la saisie de données de deux manières : l'utilisateur peut saisir des données directement dans le sous-formulaire ou les saisir dans un [formulaire de saisie](#). Dans cette configuration, le formulaire utilisé comme sous-formulaire est appelé formulaire Liste. Le formulaire de saisie est appelé le formulaire détaillé.

Grammaire JSON

Nom	Type de données	Valeurs possibles
listForm	string	Name (string) of table or project form, a POSIX path (string) to a .json file describing the form, or an object describing the form

Objets pris en charge

[Sous-formulaire](#)

Source

Spécifie la table à laquelle appartient le sous-formulaire Liste (le cas échéant).

Grammaire JSON

Nom	Type de données	Valeurs possibles
table	string	Nom de la table 4D, ou "" s'il n'existe aucune table.

Objets pris en charge

[Sous-formulaire](#)

Mode de sélection

Designates the option for allowing users to select rows:

- None: Rows cannot be selected if this mode is chosen. Cliquer sur la liste n'aura aucun effet à moins que l'option [Saisissable en liste](#) soit activée. The navigation keys only cause the list to scroll; the `On Selection Change` form event is not generated.

- Single: One row at a time can be selected in this mode. Clicking on a row will select it. A `Ctrl+click` (Windows) or `Command+click` (macOS) on a row toggles its state (between selected or not).
The Up and Down arrow keys select the previous/next row in the list. The other navigation keys scroll the list. The `On Selection Change` form event is generated every time the current row is changed.
- Multiple: Several rows can be selected simultaneously in this mode.
 - Les sous-enregistrements sélectionnés sont retournés par la commande `GET HIGHLIGHTED RECORDS`.
 - Cliquer sur l'enregistrement permettra de le sélectionner, mais ne modifiera pas l'enregistrement courant.
 - Si vous pressez `Ctrl+clic` (Windows) ou `Commande+clic` (macOS) sur un enregistrement, cela fera basculer son état (entre sélectionné ou non). Les touches fléchées Haut et Bas sélectionnent l'enregistrement précédent/suivant dans la liste. The other navigation keys scroll the list. L'événement formulaire `On Selection Change` est généré chaque fois que l'enregistrement sélectionné est modifié.

Grammaire JSON

Nom	Type de données	Valeurs possibles
<code>selectionMode</code>	<code>string</code>	<code>"multiple", "single", "none"</code>

Objets pris en charge

[Sous-formulaire](#)

Text

Autoriser sélecteur police/couleur

Lorsque cette propriété est activée, les commandes [OPEN FONT PICKER](#) et [OPEN COLOR PICKER](#) peuvent être appelées pour afficher les fenêtres de sélecteur de la police système et de couleurs. A l'aide de ces fenêtres, les utilisateurs peuvent modifier la police ou la couleur d'un objet formulaire dont le focus est accessible directement au clic. Lorsque cette propriété est désactivée (par défaut), les commandes d'ouverture du sélecteur ne produisent aucun effet.

Grammaire JSON

Propriété	Type de données	Valeurs possibles
allowFontColorPicker	boolean	false (par défaut), true

Objets pris en charge

[Input](#)

Gras

Le texte sélectionné est plus foncé et plus épais.

Vous pouvez également définir cette propriété à l'aide de la commande [OBJECT SET FONT STYLE](#).

Ceci est un texte normal.
Ceci est un texte en gras.

Grammaire JSON

Propriété	Type de données	Valeurs possibles
fontWeight	Texte	"normal", "bold"

Objets pris en charge

[Bouton](#) - [Case à cocher](#) - [Combo Box](#) - [Liste déroulante](#) - [Group Box](#) - [Liste hiérarchique](#) - [Zone de saisie](#) - [List Box](#) - [Colonne List Box](#) - [Pied List Box](#) - [En-tête List Box](#) - [Bouton Radio](#) - [Zone de texte](#)

Italique

Le texte sélectionné est légèrement penché vers la droite.

Vous pouvez également définir cette propriété à l'aide de la commande [OBJECT SET FONT STYLE](#).

Ceci est un texte normal.
Ceci est un texte en italique.

Grammaire JSON

Nom	Type de données	Valeurs possibles
fontStyle	string	"normal", "italic"

Objets pris en charge

Bouton - Case à cocher - Combo Box - Liste déroulante - Group Box - Liste hiérarchique - Zone de saisie - List Box - Colonne List Box - Pied List Box - En-tête List Box - Bouton Radio - Zone de texte

Souligné

Une ligne est placée sous le texte.

Ceci est un texte normal.
Ceci est un texte souligné..

Grammaire JSON

Nom	Type de données	Valeurs possibles
textDecoration	string	"normal", "underline"

Objets pris en charge

Bouton - Case à cocher - Combo Box - Liste déroulante - Group Box - Liste hiérarchique - Zone de saisie - List Box - Colonne List Box - Pied List Box - En-tête List Box - Bouton Radio - Zone de texte

Police

Cette propriété vous permet d'indiquer le thème de la police ou la famille de police utilisé(e) dans l'objet.

Les propriétés du thème et de la famille de police sont mutuellement exclusives. Un thème de police prend en charge les attributs de police, y compris la taille. Une famille de polices vous permet de définir le nom de la police, sa taille et sa couleur.

Thème de police

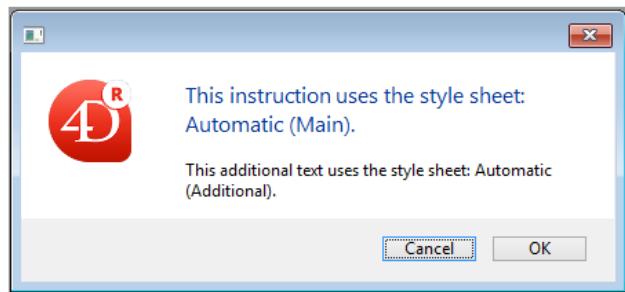
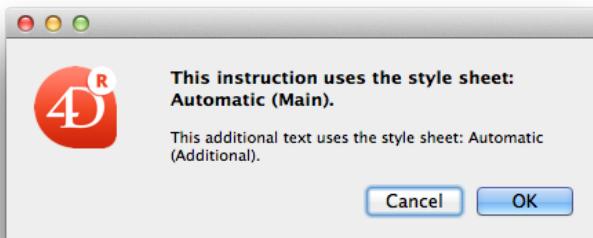
La propriété de thème de police désigne un nom de style automatique. Les styles automatiques déterminent de manière dynamique la famille de police, la taille et la couleur de police à utiliser pour l'objet, en fonction des paramètres système. Ces paramètres dépendent de :

- la plateforme,
- la langue du système,
- et le type d'objet de formulaire.

Avec le thème de police, vous avez la garantie que les titres s'affichent toujours conformément aux normes de l'interface du système. Cependant, leur taille peut varier d'une machine à l'autre.

Trois thèmes de polices sont disponibles :

- normal : style automatique, appliqué par défaut à tout nouvel objet créé dans l'éditeur de formulaires.
- Les thèmes de polices principaux et supplémentaires ne sont pris en charge uniquement par les zones de texte et les zones de saisie. Ces thèmes sont principalement destinés à la conception de boîtes de dialogue. Ils font référence aux styles de police utilisés respectivement pour le texte principal et les informations supplémentaires dans vos fenêtres d'interface. Voici les boîtes de dialogue typiques (macOS et Windows) utilisant ces thèmes de polices :



Les thèmes de polices gèrent la police ainsi que sa taille et sa couleur. Vous pouvez appliquer des propriétés de style personnalisées (Gras, Italique ou Souligné) sans modifier son fonctionnement.

Grammaire JSON

Nom	Type de données	Valeurs possibles
fontTheme	string	"normal", "main", "additional"

Objets pris en charge

Bouton - Case à cocher - Combo Box - Liste déroulante - Group Box - Liste hiérarchique - Zone de saisie - List Box - Colonne List Box - Pied List Box - En-tête List Box - Bouton Radio - Zone de texte

Famille de police

Il existe deux types de noms de familles de polices :

- *family-name* : Le nom d'une famille de polices, comme "times", "courier", "arial", etc.
- *generic-family *: Le nom d'une famille générique, comme "serif", "sans-serif", "cursive", "fantasy", "monospace".

Vous pouvez la définir à l'aide de la commande [OBJECT SET FONT](#) .

This is Times New Roman font.

This is Calibri font.

This is Papyrus font.

Grammaire JSON

Nom	Type de données	Valeurs possibles
fontFamily	string	Nom d'une famille de police CSS

4D recommande d'utiliser uniquement les polices de [sécurité Web](#).

Objets pris en charge

Bouton - Case à cocher - Combo Box - Liste déroulante - Group Box - Liste hiérarchique - Zone de saisie - List Box - Colonne List Box - Pied List Box - En-tête List Box - Bouton Radio - Zone de texte

Taille

Permet de définir en points la taille de police de l'objet.

Grammaire JSON

Nom	Type de données	Valeurs possibles
fontSize	entier	Taille de la police en points. Valeur minimale : 0

Objets pris en charge

Bouton - Case à cocher - Combo Box - Liste déroulante - Group Box - Liste hiérarchique - Zone de saisie - List Box - Colonne List Box - Pied List Box - En-tête List Box - Bouton Radio - Zone de texte

Couleur de la police

Désigne la couleur de la police.

Cette propriété définit également la couleur de **bordure** (le cas échéant) de l'objet lorsque le style "plein" ou "pointillé" est utilisé.

La couleur peut être spécifiée par :

- un nom de couleur - comme "red"
- une valeur HEX - comme "# ff0000"
- une valeur RVB - comme "rgb (255,0,0)"

Vous pouvez également définir cette propriété à l'aide de la commande [OBJECT SET RGB COLORS](#).

Grammaire JSON

Nom	Type de données	Valeurs possibles
stroke	string	une valeur css; "transparent"; "automatic"

Objets pris en charge

Bouton - Case à cocher - Combo Box - Liste déroulante - Group Box - Liste hiérarchique - Zone de saisie - List Box - Colonne List Box - Pied List Box - En-tête List Box - Indicateurs de progression - Règle - Bouton Radio - Zone de texte

Expression couleur police

List box de type collection/sélection d'entité

Utilisée pour appliquer une couleur de police personnalisée à chaque ligne de la list box. Vous devez utiliser des valeurs de couleur RVB. Pour plus d'informations à ce sujet, reportez-vous à la description de la commande [OBJECT SET RGB COLORS](#) dans le manuel Langage 4D.

Vous devez saisir une expression ou une variable (les variables de type tableau ne peuvent pas être utilisées). L'expression ou la variable sera évaluée pour chaque ligne affichée. Vous pouvez utiliser les constantes du thème [SET RGB COLORS](#).

Vous pouvez également définir cette propriété à l'aide de la commande [LISTBOX SET PROPERTY](#) avec la constante `lk font color expression`.

Cette propriété peut également être définie à l'aide d'une [expression Meta Info](#).

L'exemple suivant utilise un nom de variable : entrez `CompanyColor` pour l'expression couleur police et, dans la méthode formulaire, entrez le code suivant :

```
CompanyColor:=Choose([Companies]ID;Background color;Light shadow color;  
Foreground color;Dark shadow color)
```

Grammaire JSON

Nom	Type de données	Valeurs possibles
rowStrokeSource	string	Expression couleur police

Objets pris en charge

List Box

Expression style

`List box de type collection/sélection d'entité`

Utilisé pour appliquer un style de police personnalisé à chaque ligne de list box ou de chaque cellule de la colonne.

Vous devez saisir une expression ou une variable (les variables de type tableau ne peuvent pas être utilisées). L'expression ou variable sera évaluée pour chaque ligne affichée (si elle s'applique à la list box) ou chaque cellule affichée (si elle s'applique à la list box). Vous pouvez utiliser les constantes du thème [Styles de caractères](#).

Exemple :

```
Choose([Companies]ID;Bold;Plain;Italic;Underline)
```

Vous pouvez également définir cette propriété à l'aide de la commande `LISTBOX SET PROPERTY` avec la constante `lk font style expression`.

Cette propriété peut également être définie à l'aide d'une [expression Meta Info](#).

Grammaire JSON

Nom	Type de données	Valeurs possibles
rowStyleSource	string	Expression de style à évaluer pour chaque ligne/cellule.

Objets pris en charge

List Box - Colonne List Box

Alignement horizontal

Emplacement horizontal du texte dans la zone où il apparaît.

Grammaire JSON

Nom	Type de données	Valeurs possibles
textAlign	string	"automatic", "right", "center", "justify", "left"

Objets pris en charge

Alignment vertical

Emplacement vertical du texte dans la zone où il apparaît.

L'option Default (valeur JSON `automatique`) définit l'alignement en fonction du type de données identifiées dans chaque colonne :

- `bas` pour toutes les données (sauf les images) et
- `haut` pour les données de type image.

Cette propriété peut également être gérée par les commandes [OBJECT Get vertical alignment](#) et [OBJECT SET VERTICAL ALIGNMENT](#).

Grammaire JSON

Nom	Type de données	Valeurs possibles
verticalAlign	string	"automatic", "top", "middle", "bottom"

Objets pris en charge

[List Box](#) - [Colonne List Box](#) - [Pied List Box](#) - [En-tête List Box](#)

Meta Info expression

`List box de type collection ou entity selection (sélection d'entité)`

Indique une expression ou une variable qui sera évaluée pour chaque ligne affichée. Elle permet de définir un ensemble d'attributs texte des lignes. Vous devez passer une variable objet ou une expression qui retourne un objet. Les propriétés suivantes sont prises en charge :

Nom de propriété	Type	Description
stroke	string	Couleur de la police. Toute couleur CSS (ex : "#FF00FF"), "automatic", "transparent"
border-style	string	Couleur de fond. Toute couleur CSS (ex : "#FF00FF"), "automatic", "transparent"
fontStyle	string	"normal", "italic"
fontWeight	string	"normal", "bold"
textDecoration	string	"normal", "underline"
unselectable	boolean	Désigne la ligne correspondante comme n'étant pas sélectionnable (c'est-à-dire que le surlignage n'est pas possible). Les zones saisissables ne sont plus saisissables si cette option est activée, à moins que l'option «Single-click Edit» ne soit également activée. Les contrôles tels que les cases à cocher et les listes restent fonctionnels. This setting is ignored if the list box selection mode is "None". Valeurs par défaut : False.
disabled	boolean	Désactive la ligne correspondante. Les zones saisissables ne sont plus saisissables si cette option est activée. Le texte et les contrôles (cases à cocher, listes, etc.) sont grisés. Valeurs par défaut : False.
cell. <code><columnName></code>	object	Permet d'appliquer la propriété à une seule colonne. Passez dans <columnName> le nom d'objet de la colonne de list box. Note : les propriétés "unselectable" et "disabled" ne peuvent être définies qu'au niveau de la ligne. Elles sont ignorées si elles sont passées dans l'objet "cell"

Les paramètres de style définis avec cette propriété sont ignorés si d'autres paramètres de style sont déjà définis via des expressions (par exemple, [Style Expression](#), [Font Color Expression](#), [Background Color Expression](#)).

Exemple

Dans la méthode projet *Color*, entrez le code suivant :

```
//Méthode Color
//Définit la couleur de police pour certaines lignes et la couleur de fond pour une colonne spécifique :
C_OBJECT($0)
Form.meta:=New object
If(This.ID>5) //ID est un attribut d'objets/entités d'une collection
    Form.meta.stroke:="purple"
    Form.meta.cell:=New object("Column2";New object("fill";"black"))
Else
    Form.meta.stroke:="orange"
End if
$0:=Form.meta
```

Bonnes pratiques : Pour des raisons d'optimisation, il serait recommandé dans ce cas de créer l'objet `meta.cell` une fois contenu dans la méthode formulaire :

```
//méthode formulaire
Case of
    :(Form event code=On Load)
        Form.colStyle:=New object("Column2";New object("fill";"black"))
End case
```

La méthode *Color* contiendrait alors :

```
//méthode Color
...
If(This.ID>5)
    Form.meta.stroke:="purple"
    Form.meta.cell:=Form.colStyle //réutiliser le même objet pour de meilleures performances
...
```

Voir également la commande [This](#).

Grammaire JSON

Nom	Type de données	Valeurs possibles
metaSource	string	Expression de l'objet à évaluer pour chaque ligne/cellule.

Objets pris en charge

List Box

Multistyle

Cette propriété permet d'utiliser des styles spécifiques dans la zone sélectionnée. When this option is checked, 4D interprets any `` HTML tags found in the area.

Par défaut, cette option n'est pas activée.

Grammaire JSON

Nom	Type de données	Valeurs possibles
styledText	boolean	true, false

Objets pris en charge

[List Box Column - Input](#)

Orientation

Modifie l'orientation (rotation) d'une zone de texte. Les zones de texte peuvent être pivotées par incrément de 90°. Chaque valeur d'orientation est appliquée tout en conservant le même point de départ inférieur gauche pour l'objet :

Valeur d'orientation	Résultat
0 (par défaut)	A yellow rectangular text box containing the word "Chicago". The text is oriented horizontally. The box has blue control points at its corners and midpoints on all four sides. A grey "OK" button is at the bottom right.
90	A yellow rectangular text box containing the word "Chicago". The text is rotated 90 degrees counter-clockwise. The box has blue control points at its corners and midpoints on all four sides. A grey "OK" button is at the bottom right.
180	A yellow rectangular text box containing the word "Chicago". The text is rotated 180 degrees counter-clockwise. The box has blue control points at its corners and midpoints on all four sides. A grey "OK" button is at the bottom right.
270	A yellow rectangular text box containing the word "Chicago". The text is rotated 270 degrees counter-clockwise. The box has blue control points at its corners and midpoints on all four sides. A grey "OK" button is at the bottom right.

En plus des [zones de texte statiques](#), les objets de texte des [zones de saisie](#) peuvent être pivotés lorsqu'ils ne sont pas [saisissables](#). Lorsqu'une propriété de rotation est appliquée à un objet de saisie, la propriété saisissable est supprimée (le cas échéant). Cet objet est alors exclu de l'ordre de saisie.

Grammaire JSON

Nom	Type de données	Valeurs possibles
textAngle	number	0, 90, 180, 270

Objets pris en charge

[Zone de saisie](#) (non saisissable) - [Zone de texte](#)

Tableau couleurs de police

List box de type tableau

Permet de définir un style de police personnalisé à chaque ligne de list box ou de chaque cellule de la colonne.

Le nom d'un tableau Entier Long doit être utilisé. Chaque élément de ce tableau correspond à une ligne de la zone de list box (si elle est appliquée à la liste box) ou à une cellule de la colonne (si elle est appliquée à une colonne), le tableau doit donc avoir la même taille que le tableau associé à la colonne. Vous pouvez utiliser les constantes du thème [SET RGB COLORS](#). Si vous souhaitez que la cellule hérite de la couleur d'arrière-plan définie au niveau supérieur, passez la valeur -255 à l'élément de tableau correspondant.

Grammaire JSON

Nom	Type de données	Valeurs possibles
rowStrokeSource	string	Nom d'un tableau entier long

Objets pris en charge

[List Box](#) - [Colonne List Box](#)

Tableau de styles

List box de type tableau

[List Box](#) - [Colonne List Box](#)

Le nom d'un tableau Entier Long doit être utilisé. Chaque élément de ce tableau correspond à une ligne de la zone de list box (si elle est appliquée à la liste box) ou à une cellule de la colonne (si elle est appliquée à une colonne), le tableau doit donc avoir la même taille que le tableau associé à la colonne. Pour remplir le tableau (à l'aide d'une méthode), utilisez les constantes du thème [Styles de caractères](#). Vous pouvez ajouter des constantes ensemble pour combiner plusieurs styles. Si vous souhaitez que la cellule hérite du style défini au niveau supérieur, passez la valeur -255 à l'élément de tableau correspondant.

Grammaire JSON

Nom	Type de données	Valeurs possibles
rowStyleSource	string	Nom d'un tableau entier long.

Objets pris en charge

[List Box](#) - [Colonne List Box](#)

Stocker les balises par défaut

Cette propriété n'est disponible que pour une zone de saisie [multi-styles](#). Lorsque cette propriété est activée, la zone stockera les balises de style avec le texte, même si aucune modification n'a été apportée. Dans ce cas, les balises correspondent au style par défaut. Lorsque cette propriété est désactivée, seules les balises de style modifiées sont stockées.

Par exemple, voici un texte qui inclut une modification de style :

```
What a beautiful day
```

Lorsque la propriété est désactivée, la zone ne stocke que la modification qui a été apportée. Les contenus stockés sont donc :

```
Quelle <SPAN STYLE="font-size:13.5pt">belle</SPAN> journée !
```

Lorsque la propriété est activée, la zone stocke toutes les informations de mise en forme. La première balise générique décrit le style par défaut puis chaque variation fait l'objet d'une paire de balises imbriquées. Les contenus stockés dans la zone sont donc :

```
<SPAN STYLE="font-family:'Arial';font-size:9pt;text-align:left;font-weight:normal;font-style:normal;text-decoration:none">
```

Grammaire JSON

Nom	Type de données	Valeurs possibles
storeDefaultStyle	boolean	true, false (par défaut).

Objets pris en charge

[Input](#)

Texte et Image

Chemin d'accès arrière-plan

Définit le chemin d'accès de l'image qui sera dessinée en arrière-plan de l'objet. Si l'objet utilise une [icône avec différents états](#), l'image de fond prendra automatiquement en charge le même nombre d'états.

Le chemin d'accès à saisir est identique à celui de [la propriété Chemin d'accès pour les images statiques](#).

Grammaire JSON

Nom	Type de données	Valeurs possibles
customBackgroundPicture	string	Chemin relatif en syntaxe POSIX. Doit être utilisé avec l'option "Personnalisé" de la propriété "Style".

Objets pris en charge

[Bouton personnalisé](#) - [Case à cocher personnalisée](#) - [Bouton radio personnalisé](#)

Styles de bouton

Aspect général du bouton. Le style du bouton joue également un rôle dans la disponibilité de certaines options.

Grammaire JSON

Nom	Type de données	Valeurs possibles
style	Texte	"regular", "flat", "toolbar", "bevel", "roundedBevel", "gradientBevel", "texturedBevel", "office", "help", "circular", "disclosure", "roundedDisclosure", "custom"

Objets pris en charge

[Bouton](#) - [Bouton radio](#) - [Case à cocher](#) - [Bouton radio](#)

Marge horizontale

Cette propriété permet de définir la taille (en pixels) des marges horizontales du bouton. Cette marge délimite la zone que l'icône et le titre du bouton ne doivent pas dépasser.

Ce paramètre est utile, par exemple, lorsque l'image de fond contient des bordures :

Avec / Sans	Exemple
Sans marge	
Avec marge 13 pixels	

Cette propriété fonctionne avec la propriété [Marge verticale](#).

Grammaire JSON

Nom	Type de données	Valeurs possibles
customBorderX	number	A utiliser avec le style "personnalisé". Minimum : 0

Objets pris en charge

[Bouton personnalisé](#) - [Case à cocher personnalisée](#) - [Bouton radio personnalisé](#)

Emplacement de l'icône

Désigne l'emplacement d'une icône par rapport à l'objet formulaire.

Grammaire JSON

Nom	Type de données	Valeurs possibles
iconPlacement	string	"aucun", "gauche", "droite"

Objets pris en charge

[Entête de List Box](#)

Décalage icône

Définit une valeur de décalage personnalisée en pixels, qui sera utilisée lorsque le bouton est cliqué

Le titre du bouton sera décalé vers la droite et vers le bas pour le nombre de pixels saisis. Cela permet d'appliquer un effet 3D personnalisé lorsque le bouton est cliqué.

Grammaire JSON

Nom	Type de données	Valeurs possibles
customOffset	number	minimum : 0

Objets pris en charge

[Bouton personnalisé](#) - [Case à cocher personnalisée](#) - [Bouton radio personnalisé](#)

Nombre d'états

Cette propriété définit le nombre exact d'états présents dans l'image utilisée comme icône pour un [bouton avec icône](#), une [case à cocher](#) ou un [bouton radio](#) personnalisé. En général, une icône de bouton comprend quatre états : actif, cliqué, survolé et inactif.

Chaque état est représenté par une image différente. Dans l'image source, les états doivent être empilés verticalement :



Les états suivants sont représentés :

1. bouton non cliqué / case non cochée (valeur de la variable = 0)
2. bouton cliqué / case cochée (valeur de la variable = 1)
3. survolé
4. disabled

Grammaire JSON

Nom	Type de données	Valeurs possibles
iconFrames	number	minimum: 1

Objets pris en charge

[Bouton](#) (tous les styles sauf [Aide](#)) - [Case à cocher](#) - [Bouton radio](#)

Chemin d'accès de l'image

Définit le chemin d'accès de l'image qui sera utilisée comme icône de l'objet.

Le chemin d'accès à saisir est identique à celui de [la propriété Chemin d'accès pour les images statiques](#).

Lorsqu'elle est utilisée comme icône pour les objets actifs, l'image doit être conçue pour prendre en charge [un nombre d'états](#) variable.

Grammaire JSON

Nom	Type de données	Valeurs possibles
icon	picture	Chemin relatif ou filesystem en syntaxe POSIX.

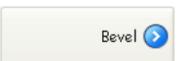
Objets pris en charge

[Bouton](#) (tous les styles sauf [Aide](#)) - [Case à cocher](#) - [En-tête List box](#) [Bouton radio](#)

Position Titre/Image

Cette propriété permet de modifier l'emplacement relatif du titre par rapport à l'icône associée. Cette propriété n'a pas d'effet lorsque le bouton contient uniquement un titre (pas d'image associée) ou une image (pas de titre). Par défaut, lorsqu'un bouton 3D contient un titre et une image, le texte est placé en-dessous de l'image.

Voici le résultat des différentes options de cette propriété :

Option	Description	Exemple
Gauche	Le texte est placé à gauche de l'icône. Le contenu du bouton est aligné à droite.	
Haut	Le texte est placé au-dessus de l'icône. Le contenu du bouton est centré.	
Droite	Le texte est placé à droite de l'icône. Le contenu du bouton est aligné à gauche.	
Bas	Le texte est placé en-dessous de l'icône. Le contenu du bouton est centré.	
Centre	Le texte de l'icône est centré verticalement et horizontalement dans le bouton. Ce paramétrage convient par exemple pour du texte inclus dans une icône.	

Grammaire JSON

Nom	Type de données	Valeurs possibles
textPlacement	string	"left", "top", "right", "bottom", "center"

Objets pris en charge

[Bouton](#) (tous les styles sauf [Aide](#)) - [Case à cocher](#) - [Bouton radio](#)

Marge verticale

Cette propriété permet de définir la taille (en pixels) des marges verticales du bouton. Cette marge délimite la zone que l'icône et le titre du bouton ne doivent pas dépasser.

Ce paramètre est utile, par exemple, lorsque l'image de fond contient des bordures.

Cette propriété fonctionne avec la propriété [Marge horizontale](#).

Grammaire JSON

Nom	Type de données	Valeurs possibles
customBorderY	number	A utiliser avec le style "personnalisé". Minimum : 0

Objets pris en charge

[Bouton personnalisé](#) - [Case à cocher personnalisée](#) - [Bouton radio personnalisé](#)

Avec pop-up menu

Cette propriété permet d'afficher un symbole en forme de triangle indiquant qu'un pop up menu lui est associé :



L'apparence et l'emplacement de ce symbole dépend du style de bouton et de la plate-forme courante.

Séparé et Lié

Pour associer un symbole de pop up menu à un bouton, vous disposez de deux options d'affichage :

Séparé	Et Lié
	

La disponibilité effective d'un mode "Séparé" dépend du style de bouton et de la plate-forme.

Chaque option précise la relation entre le bouton et le pop up menu associé :

- Lorsque le pop up menu est **séparé**, un clic sur la partie gauche du bouton exécute directement l'action courante du bouton ; cette action peut être modifiée via le pop up menu accessible dans la partie droite du bouton.
- Lorsque le pop up menu est **lié**, un simple clic sur le bouton ne déclenche aucune autre action que l'affichage du pop up menu. Seule la sélection de l'action dans le pop up menu provoque son déclenchement.

Gestion du pop up menu

Il est important de noter que la propriété "Avec pop up menu" gère uniquement l'aspect graphique du bouton. L'affichage du pop up menu et de ses valeurs doivent être entièrement gérés par le développeur, notamment à l'aide des événements formulaire et des commandes [Pop up menu dynamique](#) et [Pop up menu](#).

Grammaire JSON

Nom	Type de données	Valeurs possibles
popupPlacement	string	<ul style="list-style-type: none">◦ "none"◦ "linked"◦ "separated"

Objets pris en charge

[Bouton barre outils](#) - [Bouton Bevel](#) - [Bouton bevel circulaire](#) - [Bouton OS X Gradient](#) - [Bouton OS X Textured](#) - [Bouton Office XP](#) - [Bouton cercle](#) - [Personnalisé](#)

Zone Web

Accéder aux méthodes 4D

Il est possible d'appeler des méthodes 4D depuis le code JavaScript exécuté dans une zone Web et de recevoir des valeurs en retour. Pour pouvoir appeler des méthodes 4D depuis la zone Web, vous devez cocher l'option Accès méthodes 4D pour la zone dans la Liste des propriétés .

Cette option n'apparaît que si l'option [Utiliser le moteur de rendu Web intégré](#) est cochée.

Lorsque cette propriété est cochée, un objet JavaScript spécial `$4d` est instancié dans la zone Web et permet de [gérer les appels aux méthodes projet de 4D](#).

Grammaire JSON

Nom	Type de données	Valeurs possibles
methodsAccessibility	string	"none" (par défaut), "all"

Objets pris en charge

[Zone Web](#)

Variable Progression

La variable "Progression" est de type Entier long. Elle contient une valeur entre 0 et 100, représentant le pourcentage du chargement complet de la page affichée dans la zone Web. La variable est mise à jour automatiquement par 4D. Il n'est pas possible de la modifier manuellement.

À partir de 4D v19 R5, cette variable n'est plus mise à jour dans les zones Web utilisant le [moteur de rendu du système Windows](#).

Grammaire JSON

Nom	Type de données	Valeurs possibles
progressSource	string	Nom d'une variable Entier long

Objets pris en charge

[Zone Web](#)

Variable URL

La variable "URL" est de type chaîne. Elle contient l'URL chargé ou en cours de chargement par la zone Web associée. L'association entre la variable et la zone Web s'effectue dans les deux sens :

- Si l'utilisateur affecte un nouvel URL à la variable, l'URL est automatiquement chargé par la zone Web.
- Toute navigation effectuée à l'intérieur de la zone Web met automatiquement à jour le contenu de la variable.

Schématiquement, cette variable fonctionne comme la zone d'adresse d'un navigateur Web. Vous pouvez la représenter par une zone de texte située au-dessus de la zone Web.

Variable URL et commande WA OUVRIR URL

La variable URL produit les mêmes effets que la commande [WA OUVRIR URL](#). Les différences suivantes sont toutefois à noter :

- Pour les accès aux documents, la variable accepte uniquement des URLs conformes aux RFC ("file://c:/Mon%20Doc") et non les chemins d'accès système ("c:¥MonDoc"). La commande [WA OPEN URL](#) accepte les deux notations.
- Si la variable URL contient une chaîne vide, la zone Web ne tente pas de charger l'URL. La commande [WA OPEN URL](#) génère une erreur dans ce cas.
- Si la variable URL ne contient pas de protocole (http, mailto, file, etc.), la zone Web ajoute "http://", ce qui n'est pas le cas pour la commande [WA OPEN URL](#).
- Lorsque la zone Web n'est pas affichée dans le formulaire (lorsqu'elle se trouve sur une autre page du formulaire), l'exécution de la commande [WA OPEN URL](#) est sans effet tandis que la valorisation de la variable URL permet de mettre à jour l'URL courant.

Grammaire JSON

Nom	Type de données	Valeurs possibles
urlSource	string	Une URL.

Objets pris en charge

[Zone Web](#)

Utiliser le moteur de rendu Web intégré

Cette option vous permet de choisir entre deux moteurs de rendus pour la zone Web, en fonction des spécificités de votre application :

- non coché - `valeur JSON : system` (par défaut) : Dans ce cas, 4D utilise le "meilleur" moteur correspondant au système. Ce fonctionnement vous permet de bénéficier automatiquement des dernières avancées en matière de rendu Web, via HTML5 ou JavaScript. However, you may notice some rendering differences between platforms. On Windows, 4D uses Microsoft Edge WebView2. Sur macOS, 4D utilise la version courante du WebKit (Safari).

On Windows, if Microsoft Edge WebView2 is not installed, 4D uses the embedded engine as system rendering engine. To know if it is installed in your system, look for "Microsoft Edge WebView2 Runtime" in your applications panel.

- checked - `JSON value: embedded` : In this case, 4D uses the Chromium Embedded Framework (CEF). L'utilisation d'un moteur Web intégré vous permet d'avoir l'assurance que le rendu et le fonctionnement des zones Web de votre application seront quasiment identiques, quelle que soit la plate-forme d'exécution de 4D (de légères variations de pixels ou des différences liées à l'implémentation réseau pourront toutefois être constatées). L'utilisation d'un moteur Web intégré vous permet d'avoir l'assurance que le rendu et le fonctionnement des zones Web de votre application seront quasiment identiques, quelle que soit la plate-forme d'exécution de 4D (de légères variations de pixels ou des différences liées à l'implémentation réseau pourront toutefois être constatées).

The CEF engine has the following limitations:

- [WA SET PAGE CONTENT](#): using this command requires that at least one page is already loaded in the area (through a call to [WA OPEN URL](#) or an assignment to the URL variable associated to the area).
- When URL drops are enabled by the `WA enable URL drop` selector of the [WA SET PREFERENCE](#) command, the first drop must be preceded by at least one call to [WA OPEN URL](#) or one assignment to the URL variable associated to the area.

Grammaire JSON

Nom	Type de données	Valeurs possibles
webEngine	string	"embedded", "system"

Objets pris en charge

[Zone Web](#)

Sur activation

Code	Peut être appelé par	Définition
11	Formulaire	La fenêtre du formulaire devient la fenêtre de premier plan ou bien le conteneur du sous-formulaire obtient le focus

Description

Si la fenêtre d'un formulaire a été envoyée en arrière-plan, cet événement est appelé lorsque la fenêtre devient la fenêtre active.

Cet événement s'applique au formulaire dans son ensemble et non à un objet particulier. Par conséquent, si la propriété d'événement formulaire `On Activate` est sélectionnée, seul le formulaire aura sa méthode formulaire appelée.

Dans le cas d'un sous-formulaire, cet événement est passé au sous-formulaire lorsque le conteneur obtient le focus (s'il possède la propriété [focusable](#)).

Sur après modification

Code	Peut être appelé par	Définition
45	Zone 4D View Pro - Zone 4D Write Pro - Combo Box - Formulaire - Zone de saisie - Liste hiérarchique - List Box - Colonne List Box	Le contenu de l'objet saisissable qui a le focus vient d'être modifié

Description

Cas général

Cet événement peut être utilisé pour filtrer la saisie de données dans les objets saisissables au clavier au niveau le plus bas.

Lorsqu'il est utilisé, cet événement est généré après chaque modification apportée au contenu d'un objet saisissable, quelle que soit l'action qui a provoqué la modification, c'est-à-dire :

- Actions d'édition standard qui modifient le contenu comme les actions coller, couper, supprimer ou annuler;
- Déposer une valeur (action similaire à coller);
- Toute entrée au clavier effectuée par l'utilisateur; dans ce cas, l'événement `On After Edit` est généré après les événements `On Before Keystroke` et `On After Keystroke`, s'ils sont utilisés.
- Toute modification apportée à l'aide d'une commande de langage qui simule une action de l'utilisateur (c'est-à-dire `POST KEY`).

Dans le cadre de l'événement `On After Edit`, les données textuelles en cours de saisie sont retournées par la commande `Get edited text`.

4D View Pro

L'objet retourné par la commande `FORM Event` contient :

Propriété	Type	Description
code	entier long	Sur après modification
description	Texte	"On After Edit"
objectName	Texte	Nom de la zone 4D View Pro
sheetName	Texte	Nom de la feuille de l'événement
action	Texte	"editChange", "valueChanged", "DragDropBlock", "DragFillBlock", "formulaChanged", "clipboardPasted"

En fonction de la valeur de la propriété `action`, l'`objet event` contiendra des propriétés supplémentaires.

`action = editChange`

Propriété	Type	Description
range	object	Plage de cellule
editingText	variant	La valeur provenant de l'éditeur courant

`action = valueChanged`

Propriété	Type	Description
range	object	Plage de cellule
oldValue	variant	Valeur de la cellule avant la modification
newValue	variant	Valeur de la cellule après la modification

action = DragDropBlock

Propriété	Type	Description
fromRange	object	Plage de cellule source (qui est glissée)
toRange	object	Plage de cellule de destination (qui est déposée)
copy	boolean	Indique si la plage source est copiée ou non
insert	boolean	Indique si la plage source est insérée ou non

action = DragFillBlock

Propriété	Type	Description	fillDirection	longint	Direction du remplissage.
fillRange	object	Plage utilisée pour le remplissage			
autoFillType	longint	Valeur utilisée pour le remplissage. <ul style="list-style-type: none"> • 0 : les cellules contiennent toutes les données (valeurs, formatage et formules) • 1 : les cellules contiennent des données automatiquement séquentielles • 2 : les cellules contiennent uniquement le formatage • 3 : les cellules contiennent des valeurs mais pas de formatage • 4 : les valeurs des cellules sont supprimées • 5: Les cellules sont remplies automatiquement 			<ul style="list-style-type: none"> • 0 : les cellules à gauche sont remplies • 1 : les cellules à droite sont remplies • 2 : Les cellules ci-dessus sont remplies • 3 : Les cellules ci-dessous sont remplies

action = formulaChanged

Propriété	Type	Description
range	object	Plage de cellule
formula	Texte	La formule saisie

action = clipboardPasted

Propriété	Type	Description
range	object	Plage de cellule
pasteOption	entier long	Indique ce qui est collé à partir du presse-papiers : <ul style="list-style-type: none"> • 0 : tout est collé (valeurs, mise en forme et formules) • 1 : seules les valeurs sont collées • 2 : seul le formatage est collé • 3 : seules les formules sont collées • 4 : les valeurs et la mise en forme sont collées (pas les formules) • 5 : Les formules et la mise en forme sont collées (pas les valeurs)
pasteData	object	Les données du presse-papiers à coller <ul style="list-style-type: none"> • "text" (texte) : le texte du presse-papiers • "html" (texte) : le code HTML du presse-papiers

Exemple

Voici un exemple qui gère l'événement `On After Edit` :

```
If(FORM Event.code=On After Edit)
  If(FORM Event.action="valueChanged")
    ALERT("WARNING: You are currently changing the value\
      from "+String(FORM Event.oldValue)+\
      " to "+String(FORM Event.newValue)+"!")
  End if
End if
  End if
End if
  End if
End if
```

L'exemple ci-dessus pourrait générer un objet événement tel que celui-ci :

```
{
  "code":45,
  "description":"On After Edit",
  "objectName":"ViewProArea",
  "sheetname":"Sheet1",
  "action":"valueChanged",
  "range": {area:ViewProArea,ranges:[{column:1,row:2,sheet:1}]},
  "oldValue":"The quick brown fox",
  "newValue":"jumped over the lazy dog"
}
```

Sue après frappe clavier

Code	Peut être appelé par	Définition
28	4D Write Pro area - Combo Box - Form - Input - List Box - List Box Column	Un caractère est sur le point d'être saisi dans l'objet qui a le focus. <code>Get edited text</code> returns the object's text including this character.

► Historique

Description

L'événement `On After Keystroke` peut généralement être remplacé par l'événement `On After Edit` (voir ci-dessous).

Après avoir sélectionné les propriétés d'événement `On Before Keystroke` et `On After Keystroke` pour un objet, vous pouvez détecter et gérer les frappes au sein de l'objet, en utilisant la commande `FORM event` qui renverra `On Before Keystroke` puis `On After Keystroke` (pour plus d'informations, veuillez reportez-vous à la description de la commande `Get edited text`).

Ces événements sont également activés par des commandes de langage qui simulent une action utilisateur telle que `POST KEY`.

L'événement `On After Keystroke` n'est pas généré :

- dans la méthode [des colonnes de list box](#), sauf lorsqu'une cellule est en cours d'édition (cependant elle est générée dans tous les cas dans la méthode de [list box](#)),
- lorsque les modifications utilisateur ne sont pas effectuées à l'aide du clavier (coller, glisser-déposer, case à cocher, liste déroulante, combo box). Pour traiter ces événements, vous devez utiliser [On After Edit](#) .

Séquence de frappe

Lorsqu'une entrée nécessite une séquence de frappes clavier, les événements `On Before Keystroke` et [`On After Keystroke event`] sont générés uniquement lorsque l'entrée est entièrement validée par l'utilisateur. La commande `Keystroke` retourne le caractère validé. Ce cas se produit principalement :

- lors de l'utilisation de touches "mortes" telles que ^ ou ~: les événements ne sont générés que lorsque le caractère étendu est éventuellement saisi (par exemple "ê" ou ñ),
- lorsqu'un IME (Input method editor) affiche une boîte de dialogue intermédiaire où l'utilisateur peut saisir une combinaison de caractères : les événements sont générés uniquement lorsque la boîte de dialogue IME est validée.

Voir aussi

[On Before Keystroke](#).

Sur après tri

Code	Peut être appelé par	Définition
30	List Box - Colonne de List Box	Un tri standard vient d'être effectué dans une colonne de list box.

Description

Cet événement est généré juste après un tri standard (c'est-à-dire qu'il n'est PAS généré si \$0 retourne -1 dans l'événement [On Header Click](#)). Ce mécanisme est utile pour stocker les directions du dernier tri effectué par l'utilisateur. Dans ce cas, la commande `Self` retourne un pointeur vers la variable de l'en-tête de colonne triée.

Sur clic alternatif

Code	Peut être appelé par	Définition
38	Bouton - List Box - Colonne de List Box	<ul style="list-style-type: none">Boutons : la zone "flèche" d'un bouton est cliquéeList box : dans une colonne d'un tableau, un bouton de sélection (attribut "alternateButton") est cliqué

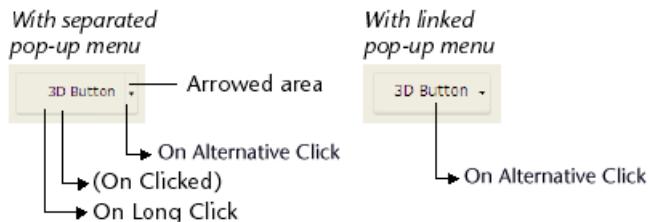
Description

Boutons

Certains styles de boutons peuvent être [liés à un menu contextuel](#) et afficher un triangle. En cliquant sur ce triangle, une fenêtre contextuelle de sélection apparaît et fournit un ensemble d'actions alternatives en relation avec l'action du bouton principal.

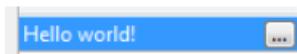
4D vous permet de gérer ce type de bouton à l'aide de l'événement `On Alternative Click`. Cet événement est généré lorsque l'utilisateur clique sur le triangle (dès que le bouton de la souris est maintenu enfoncé) :

- Si le menu pop-up est séparé, l'événement n'est généré que lorsqu'un clic se produit sur la partie du bouton avec la flèche.
- Si le pop-up menu est lié, l'événement est généré lorsqu'un clic se produit sur n'importe quelle partie du bouton. A noter que l'événement `On Long Click` ne peut pas être généré avec ce type de bouton.



List box

Cet événement est généré dans des colonnes de [list box de type tableau objets](#), lorsque l'utilisateur clique sur un bouton de sélection de widget (attribut "AlternateButton").



Voir la [description de l'attribut "alternateButton"](#).

Sur avant saisie

Code	Peut être appelé par	Définition
41	List Box - Colonne de List Box	Une cellule de list box est sur le point de passer en mode d'édition

Description

Cet événement est généré juste avant la modification d'une cellule de list box (avant l'affichage du curseur d'entrée). Cet événement permet par exemple au développeur d'afficher un texte différent selon le mode de l'utilisateur (mode affichage ou mode édition).

Lorsque le curseur arrive dans la cellule, l'événement `On Before Data Entry` est généré dans la list box ou la méthode de la colonne.

- Si, dans le contexte de cet événement, \$0 est défini sur -1, la cellule est considérée comme non saisissable. Si l'événement a été généré après avoir appuyé sur Tab ou Maj+Tab, le focus va respectivement à la cellule suivante ou à la précédente.
- Si la valeur de \$0 n'est pas -1 (par défaut \$0 est 0), la cellule est saisissable et passe en mode d'édition.

Voir également la section [Gestion des entrées](#).

Sur avant frappe clavier

Code	Peut être appelé par	Définition
17	4D Write Pro area - Combo Box - Form - Input - List Box - List Box Column	Un caractère est sur le point d'être saisi dans l'objet qui a le focus. <code>Get edited text</code> retourne le texte de l'objet, sans ce caractère.

► Historique

Description

Après avoir sélectionné les événements `On Before Keystroke` et `On After Keystroke event` pour un objet, vous pouvez détecter et gérer les frappes au sein de l'objet, en utilisant la commande `Form event` qui retournera `On Before Keystroke` puis `On After Keystroke` (pour plus d'informations, veuillez vous reporter à la description de la commande `Get edited text`). Dans l'événement `On Before Keystroke`, la commande `FILTER KEYSTROKE` peut être utilisée pour filtrer les caractères typés.

Ces événements sont également activés par des commandes de langage qui simulent une action utilisateur telle que `POST KEY`.

L'événement `On Before Keystroke` n'est pas généré :

- dans une méthode `colonnes de list box`, sauf lorsqu'une cellule est en cours d'édition (cependant elle est générée dans tous les cas dans la méthode de `list box`),
- lorsque les modifications utilisateur ne sont pas effectuées à l'aide du clavier (coller, glisser-déposer, case à cocher, liste déroulante, combo box). Pour traiter ces événements, vous devez utiliser `On After Edit`.

Objets non saisissables

L'événement `On Before Keystroke` peut être généré dans des objets non saisissables, par exemple dans une list box même si les cellules de la list box ne sont pas saisissables ou si les lignes ne peuvent pas être sélectionnées. Cela vous permet de créer des interfaces dans lesquelles l'utilisateur peut faire défiler dynamiquement jusqu'à une ligne spécifique dans une list box en saisissant les premières lettres d'une valeur. Dans le cas où les cellules de la list box sont saisissables, vous pouvez utiliser la commande `Is editing text` pour savoir si l'utilisateur saisit réellement du texte dans une cellule ou s'il utilise la fonction de saisie prédictive, puis exécutez le code approprié.

Séquence de frappe

Lorsqu'une entrée nécessite une séquence de frappes clavier, les événements ```On Before Keystroke` et [`On After Keystroke`] sont générés uniquement lorsque la saisie est entièrement validée par l'utilisateur. La commande `Keystroke` retourne le caractère validé. Ce cas se produit principalement :

- lors de l'utilisation de touches "mortes" telles que ^ ou ~: les événements ne sont générés que lorsque le caractère étendu est éventuellement saisi (par exemple "ê" ou ñ),
- lorsqu'un IME (Input method editor) affiche une boîte de dialogue intermédiaire où l'utilisateur peut saisir une combinaison de caractères : les événements sont générés uniquement lorsque la boîte de dialogue IME est validée.

Voir aussi

[On After Keystroke](#).

Sur début survol

Code	Peut être appelé par	Définition
17	Zone 4D WritePro - Bouton - Grille de boutons - Case à cocher - Liste déroulante - Formulaire - Liste hiérarchique - Zone de saisie - List Box - Colonne List Box - Bouton image - Pop up menu image - Zone de plug-in - Indicateur de progression - Bouton radio - Règle - Spinner - Splitter - Stepper - Onglet	Un objet est en cours de déplacement

Description

L'événement de formulaire `On Begin Drag Over` peut être sélectionné pour tous les objets formulaire pouvant être déplacés. Il est généré dans tous les cas où l'objet a la propriété **Draggable**. Il peut être appelé à partir de la méthode de l'objet source ou de la méthode formulaire de l'objet source.

Contrairement à l'événement de formulaire `On Drag Over`, `On Begin Drag Over` est appelé dans le contexte de l'objet source de l'action de glisser.

L'événement `On Begin Drag Over` est utile pour préparer l'action de glisser. Il peut être utilisé pour :

- Ajouter des données et des signatures au conteneur (via la commande `APPEND DATA TO PASTEBOARD`).
- Utiliser une icône personnalisée pendant l'action de glissement (via la commande `SET DRAG ICON`).
- Accepter ou refuser le glissement via \$0 dans la méthode de l'objet glissé.
 - Pour indiquer que les actions de glissement sont acceptées, la méthode de l'objet source doit retourner 0 (zéro); vous devez donc exécuter `$0:=0`.
 - Pour indiquer que les actions de glissement sont refusées, la méthode de l'objet source doit retourner -1 (moins un); vous devez donc exécuter `$0:=-1`.
 - Si aucun résultat n'est retourné, 4D considère que les actions de glissement sont acceptées.

Les données 4D sont placées dans le presse-papiers avant d'appeler l'événement. Par exemple, dans le cas d'un glissement sans l'action de glissement automatique, le texte glissé se trouve déjà dans le conteneur lorsque l'événement est appelé.

On Begin URL Loading

Code	Peut être appelé par	Définition
47	Zone Web	Une nouvelle URL est chargée dans la zone Web

Description

Cet événement est généré au début du chargement d'une nouvelle URL dans la zone Web. La variable `URL` associée à la zone Web peut être utilisée pour connaître l'URL en cours de chargement.

L'URL en cours de chargement est différente de [l'URL courante](#) (reportez-vous à la description de la commande [WA Get current URL](#)).

On Bound Variable Change

Code	Peut être appelé par	Définition
54	Formulaire	La variable liée à un sous-formulaire est modifiée

Description

Cet événement est généré dans le contexte de la méthode formulaire d'un [sous-formulaire](#) dès qu'une valeur est affectée à la variable liée au sous-formulaire du formulaire parent (même si la même valeur est réaffectée) et si le sous-formulaire appartient à la page formulaire courante ou à la page 0.

Pour plus d'informations, reportez-vous à la section [Gérer la variable liée](#).

Sur clic

Code	Peut être appelé par	Définition
4	Zone 4D View Pro - Zone 4D Write Pro - Bouton - Grille de boutons - Case à cocher - Combo Box - Liste déroulante - Formulaire - Liste hiérarchique - Zone de saisie - List Box - Colonne de List Box - Bouton image - Pop up menu image - Zone de plug-in - Indicateur de progression - Bouton radio - Règle - Spinner - Splitter - Stepper - Onglet	Un clic a été effectué sur un objet

Description

L'événement `On Clicked` est généré lorsque l'utilisateur clique sur un objet.

Certains objets de formulaire peuvent être activés avec le clavier. Par exemple, une fois qu'une case à cocher obtient le focus, elle peut être saisie à l'aide de la barre d'espace. Dans ce cas, l'événement `On Clicked` est toujours généré.

L'événement `On Clicked` se produit généralement une fois que le bouton de la souris est relâché. Cependant, il existe plusieurs exceptions :

- **Boutons invisibles**: l'événement `On Clicked` se produit dès que le clic est effectué et n'attend pas que le bouton de la souris soit relâché.
- **Règles** : si l'option de `méthode d'exécution d'objet` est définie sur true, l'événement `On Clicked` se produit dès que le clic est effectué.
- **Combo box** : l'événement `On Clicked` se produit uniquement si l'utilisateur sélectionne une autre valeur dans le menu associé. Une `combo box` doit être traitée comme une zone de texte saisissable dont la liste déroulante associée fournit des valeurs par défaut. Par conséquent, vous gérez la saisie de données dans une `combo box` via les événements `On Before Keystroke`, `On After Keystroke` et `On Data Change`.
- **Listes déroulantes** : l'événement `On Clicked` se produit uniquement si l'utilisateur sélectionne une autre valeur dans le menu. L'événement `On Data Change` vous permet de détecter l'activation de l'objet lorsqu'une valeur différente de la valeur courante est sélectionnée
- Lorsqu'une cellule d'entrée de list box est `en cours d'édition`, l'événement `On Clicked` est généré lorsque le bouton de la souris est enfoncé, permettant d'utiliser la commande `Contextual click` par exemple.

Dans le cas d'un événement `On Clicked`, vous pouvez tester le nombre de clics effectués par l'utilisateur à l'aide de la commande `Clickcount`.

On Clicked et On Double Clicked

Une fois que la propriété d'événement d'objet `On Clicked` ou `On Double Clicked` est sélectionnée pour un objet, vous pouvez détecter et gérer les clics dans ou sur l'objet, à l'aide de la commande `FORM event` qui retourne `On Clicked` ou `On Double Clicked`, selon le cas.

Si les deux événements sont sélectionnés pour un objet, les événements `On Clicked` puis `On Double Clicked` seront générés lorsque l'utilisateur double-clique sur l'objet.

4D View Pro

Cet événement est généré lorsque l'utilisateur clique n'importe où dans un document 4D View Pro. Dans ce contexte, l'`objet événement` retourné par la commande `FORM Event` contient :

Propriété	Type	Description
code	entier long	Sur clic
description	Texte	"On Clicked"
objectName	Texte	Nom de la zone 4D View Pro
sheetName	Texte	Nom de la feuille de l'événement
range	object	Plage de cellule

Exemple

```
If(FORM Event.code=On Clicked)
    VP SET CELL STYLE(FORM Event.range;New object("backColor";"green"))
End if
```

On Close Box

Code	Peut être appelé par	Définition
22	Formulaire	La case de fermeture de la fenêtre a été cliquée

Description

L'événement `On Close Box` est généré lorsque l'utilisateur clique sur la case fermeture de la fenêtre.

Exemple

Cet exemple illustre comment vous pouvez répondre à un événement de fermeture de fenêtre à l'aide d'un formulaire utilisé pour la saisie de données d'enregistrement :

```
//Méthode pour un formulaire d'entrée
$vpFormTable:=Current form table
Case of
//...
:(Form event code=On Close Box)
If(Modified record($vpFormTable->))
    CONFIRM("This record has been modified. Save Changes?")
    Save Changes?"")
    Save Changes?"")
    If(OK=1)
        ACCEPT
    Else
        CANCEL
    End if
Else
    CANCEL
End if
//...
//déclaration(s)
End case
```

Sur fermeture corps

Code	Peut être appelé par	Définition
26	Formulaire - List Box	Vous avez quitté le formulaire détaillé et vous retournez au formulaire de sortie

Description

L'événement `On Close Detail` peut être utilisé dans les contextes suivants :

- Formulaires de sortie : le formulaire détaillé est fermé et l'utilisateur retourne au formulaire liste. Cet événement ne peut pas être sélectionné pour les formulaires projet, il est uniquement disponible avec les formulaires table.
- List box [de type sélection](#) : Cet événement est généré lorsqu'un enregistrement affiché dans le [formulaire détaillé](#) associé à une list box de type sélection est sur le point d'être fermé (que l'enregistrement ait été modifié ou non).

Sur contracter

Code	Peut être appelé par	Définition
44	Liste hiérarchique - List Box	Un élément de la liste hiérarchique ou de la list box hiérarchique a été réduit à l'aide d'un clic ou d'une touche du clavier

Description

- [Liste hiérarchique](#) : Cet événement est généré chaque fois qu'un élément de la liste hiérarchique est réduit via un clic de souris ou une touche du clavier.
- [List box hiérarchiques](#) : Cet événement est généré lorsqu'une ligne de la list box hiérarchique est réduite.

Voir aussi

[Sur déployer](#)

Sur déplacement colonne

Code	Peut être appelé par	Définition
32	List Box - Colonne de List Box	Une colonne de list box est déplacée par l'utilisateur par glisser-déposer

Description

Cet événement est généré lorsqu'une colonne de list box est déplacée par l'utilisateur à l'aide du glisser-déposer ([s'il est autorisé](#)). Il n'est pas généré si la colonne est glissée puis déposée à son emplacement initial.

La commande `LISTBOX MOVED COLUMN NUMBER` retourne la nouvelle position de la colonne.

Sur redimensionnement colonne

Code	Peut être appelé par	Définition
33	Zone 4D View Pro - List Box - Colonne de List Box	La largeur d'une colonne est modifiée directement par l'utilisateur ou à la suite d'un redimensionnement de la fenêtre de formulaire

Description

List Box

Cet événement est généré lorsque la largeur d'une colonne dans la list box est modifiée par un utilisateur. L'événement est déclenché "en direct", c'est-à-dire envoyé en continu pendant l'événement, tant que la list box ou la colonne concernée est redimensionnée. Ce redimensionnement s'effectue manuellement par un utilisateur, ou peut se produire suite au redimensionnement de la list box et de ses colonnes avec la fenêtre de formulaire elle-même (que le formulaire soit redimensionné manuellement ou à l'aide de la commande `RESIZE FORM WINDOW`).

L'événement `On Column Resize` n'est pas déclenché lorsqu'une [fausse colonne](#) est redimensionnée.

4D View Pro

Cet événement est généré lorsque la largeur d'une colonne est modifiée par un utilisateur. Dans ce contexte, l' [objet événement](#) retourné par la commande `FORM Event` contient :

Propriété	Type	Description
code	entier long	Sur redimensionnement colonne
description	Texte	"On Column Resize"
objectName	Texte	Nom de la zone 4D View Pro
sheetName	Texte	Nom de la feuille de l'événement
range	object	Plage de cellules des colonnes dont les largeurs ont changé
header	boolean	"True" si la colonne d'en-tête de ligne (première colonne) est redimensionnée, sinon false

Exemple

```
If(FORM Event.code=On Column Resize)
    VP SET CELL STYLE(FORM Event.range;New object("hAlign";vk horizontal align right))
End if
```

Sur données modifiées

Code	Peut être appelé par	Définition
20	Zone 4D Write Pro - Liste déroulante - Formulaire - Liste hiérarchique - Zone de saisie - List Box - Colonne de List Box - Zone de Plug-in - Indicateur de progression - Règle - Spinner - Stepper - Sous-formulaire	Une donnée a été modifiée

Description

Lorsque la propriété d'événement `On Data Change` est sélectionnée pour un objet, vous pouvez détecter et gérer la modification de la valeur de la source de données à l'aide de la commande `FORM Event`.

L'événement est généré dès que la variable associée à l'objet est mise à jour en interne par 4D (c'est-à-dire, en général, lorsque la zone de saisie de l'objet perd le focus).

Avec les [sous-formulaires](#), l'événement `On Data Change` est déclenché lorsque la valeur de la variable de l'objet sous-formulaire a été modifiée.

On Deactivate

Code	Peut être appelé par	Définition
12	Formulaire	La fenêtre du formulaire cesse d'être la fenêtre active

Description

Si la fenêtre d'un formulaire était la fenêtre de premier plan, cet événement est appelé lorsque la fenêtre est envoyée en arrière-plan.

Cet événement s'applique au formulaire dans son ensemble et non à un objet particulier. Par conséquent, si la propriété de l'événement formulaire `On Deactivate` est sélectionnée, seul le formulaire verra sa méthode formulaire appelée.

Voir aussi

[Sur activation](#)

Sur action suppression

Code	Peut être appelé par	Définition
58	Liste hiérarchique - List Box	L'utilisateur tente de supprimer un élément

Description

Cet événement est généré chaque fois qu'un utilisateur tente de supprimer le ou les éléments sélectionnés en appuyant sur une touche de suppression (Supprimer ou Retour en arrière) ou en sélectionnant un élément de menu dont l'action standard associée est 'Effacer' (telle que la commande Effacer dans le menu Edition).

A noter que la génération de l'événement est la seule action réalisée par 4D : le programme ne supprime aucun élément. Il appartient au développeur de gérer la suppression et tous les messages d'avertissement précédents qui sont affichés.

Sur affichage corps

Code	Peut être appelé par	Définition
8	Formulaire - List Box	Un enregistrement est sur le point d'être affiché dans un formulaire liste ou bien une ligne est sur le point d'être affichée dans une list box.

Description

L'événement `On Display Detail` peut être utilisé dans les contextes suivants :

Formulaire de sortie

Un enregistrement est sur le point d'être affiché sous forme de liste affichée via `DISPLAY SELECTION` et `MODIFY SELECTION`.

Cet événement ne peut pas être sélectionné pour les formulaires projet, il est uniquement disponible avec les formulaires table.

Dans ce contexte, la séquence d'appels de méthodes et d'événements de formulaire suivante est déclenchée :

- Pour chaque enregistrement :
 - Pour chaque objet de la zone détaillée :
 - Méthode objet avec l'événement `On Display Detail`
 - Méthode formulaire avec l'événement `On Display Detail`

La zone d'en-tête est gérée à l'aide de l'événement `On Header`.

L'appel d'une commande 4D qui affiche une boîte de dialogue à partir de l'événement `On Display Detail` n'est pas autorisé et générera une erreur de syntaxe. Plus particulièrement, les commandes concernées sont : `ALERT`, `DIALOG`, `CONFIRM`, `Request`, `ADD RECORD`, `MODIFY RECORD`, `DISPLAY SELECTION`, et `MODIFY SELECTION`.

Liste box sélection

Cet événement est généré lorsqu'une ligne de list box [de type sélection](#) est affichée.

Numéro de ligne affiché

La commande 4D `Displayed line number` fonctionne avec l'événement formulaire `On Display Detail`. Elle retourne le numéro de la ligne en cours de traitement tandis qu'une liste d'enregistrements ou de lignes de list box s'affiche à l'écran.

Sur double clic

Code	Peut être appelé par	Définition
13	Zone 4D View Pro - Zone 4D Write Pro - Bouton - Grille de boutons - Case à cocher - Combo Box - Liste déroulante - Formulaire - Liste hiérarchique - Zone de saisie - List Box - Colonne de List Box - Bouton image - Pop up menu image - Zone de plug-in - Indicateur de progression - Bouton radio - Règle - Spinner - Splitter - Stepper - Onglet	Un double-clic a été effectué sur un objet

Description

L'événement `On Double Clicked` est généré lorsque l'utilisateur double-clique sur un objet. La durée maximale séparant un double-clic est définie dans les préférences système.

Si la propriété `On Clicked` ou `On Double Clicked` d'événement d'objet de `onDoubleClicked.md` est sélectionnée pour un objet, vous pouvez détecter et gérer les clics dans ou sur l'objet, à l'aide de la commande `FORM event` qui retourne `On Clicked` ou `On Double Clicked`, selon le cas.

Si les deux événements sont sélectionnés pour un objet, les événements `On Clicked` puis `On Double Clicked` seront générés lorsque l'utilisateur double-clique sur l'objet.

4D View Pro

Cet événement est généré lorsque l'utilisateur double-clique n'importe où dans un document 4D View Pro. Dans ce contexte, l'[objet événement](#) retourné par la commande `FORM Event` contient :

Propriété	Type	Description
code	entier long	13
description	Texte	"On Double Clicked"
objectName	Texte	Nom de la zone 4D View Pro
sheetName	Texte	Nom de la feuille de l'événement
range	object	Plage de cellule

Exemple

```
If(FORM Event.code=On Double Clicked)
    $value:=VP Get value(FORM Event.range)
End if
```

Sur glisser

Code	Peut être appelé par	Définition
21	Zone 4D Write Pro - Bouton - Grille de boutons - Case à cocher - Liste déroulante - Liste hiérarchique - Zone de saisie -List Box - Colonne de List Box - Bouton image - Pop up menu image - Zone de plug-in - Indicateur de progression - Bouton radio - Règle -Spinner - Splitter - Stepper - Onglet	Les données peuvent être déposées sur un objet

Description

L'événement `On Drag Over` est envoyé à plusieurs reprises à l'objet de destination lorsque le pointeur de la souris est déplacé sur l'objet. Généralement, en réponse à cet événement :

- Vous récupérez les données et les signatures présentes dans le conteneur (via la commande `GET PASTEBOARD DATA`).
- En fonction de la nature et du type de données dans le conteneur, vous acceptez ou refusez le glisser-déposer.

Pour accepter le glissement, la méthode de l'objet de destination doit retourner 0 (zéro), vous devez donc écrire `$0:=0`. Pour rejeter le glissement, la méthode de l'objet de destination doit retourner -1 (moins un), vous devez donc écrire `$0:=-1`. Lors d'un événement `On Drag Over`, 4D traite la méthode objet comme une fonction. Si aucun résultat n'est retourné, 4D suppose que le glissement est accepté.

Si vous acceptez le glissement, l'objet de destination est mis en surbrillance. Si vous refusez le glissement, la destination n'est pas mise en surbrillance. Accepter le glissement ne signifie pas que les données déplacées vont être insérées dans l'objet de destination. Cela signifie seulement que si le bouton de la souris était relâché à ce stade, l'objet de destination accepterait les données glissées et l'événement `On Drop` serait déclenché.

Si vous ne traitez pas l'événement `On Drag Over` pour un objet déposable, cet objet sera mis en surbrillance pour toutes les opérations de glissement, quels que soient la nature et le type des données déplacées.

L'événement `On Drag Over` est le moyen par lequel vous contrôlez la première phase d'une opération de glisser-déposer. Vous pouvez non seulement tester si les données déplacées sont d'un type compatible avec l'objet de destination, puis accepter ou rejeter le glissement; vous pouvez simultanément avertir l'utilisateur de ce fait, car 4D met en évidence (ou non) l'objet de destination, en fonction de votre décision.

Le code gérant un événement `On Drag Over` doit être court et s'exécuter rapidement, car cet événement est envoyé à plusieurs reprises à l'objet de destination courant, en raison des mouvements de la souris.

Voir aussi

[Sur début survol](#)

Sur déposer

Code	Peut être appelé par	Définition
16	Zone 4D WritePro - Bouton - Grille de boutons - Case à cocher - Liste déroulante - Formulaire - Liste hiérarchique - Zone de saisie - List Box - Colonne List Box - Bouton image - Pop up menu image - Zone de plug-in - Indicateur de progression - Bouton radio - Règle - Spinner - Splitter - Stepper - Onglet	Les données ont été déposées sur un objet

Description

L'événement `On Drop` est envoyé une fois à l'objet de destination lorsque le pointeur de la souris est relâché sur l'objet. Cet événement est la deuxième phase de l'opération de glisser-déposer, où l'opération que vous réalisez est en réponse à l'action de l'utilisateur.

Cet événement n'est pas envoyé à l'objet si le glissement n'a pas été accepté lors des événements `On Drag Over`. Si vous traitez l'événement `On Drag Over` pour un objet et rejetez un glissement, l'événement `On Drop` ne se produit pas. Ainsi, si lors de l'événement `On Drag Over` vous avez testé la compatibilité des types de données entre les objets source et destination, et si vous avez accepté un éventuel dépôt, vous n'avez pas besoin de re-tester les données pendant l'événement `On Drop`. Vous savez déjà que les données sont adaptées à l'objet de destination.

Voir aussi

[Sur début survol](#)

On End URL Loading

Code	Peut être appelé par	Définition
49	Zone Web	Toutes les ressources de l'URL ont été chargées

Description

Cet événement est généré une fois que le chargement de toutes les ressources de l'URL est terminé. Vous pouvez appeler la commande `WA Get current URL` afin d'obtenir l'URL chargée.

Sur déployer

Code	Peut être appelé par	Définition
44	Liste hiérarchique - List Box	Un élément de la liste hiérarchique ou de la list box hiérarchique a été développé à l'aide d'un clic ou d'une touche

Description

- [Liste hiérarchique](#) : Cet événement est généré chaque fois qu'un élément de la liste hiérarchique est étendu via un clic de souris ou une touche du clavier.
- [List box hiérarchiques](#) : Cet événement est généré lorsqu'une ligne de la list box hiérarchique est étendue.

Voir aussi

[Sur contracter](#)

Sur clic pied

Code	Peut être appelé par	Définition
57	List Box - Colonne de List Box	Un clic se produit dans le pied de page d'une colonne de list box

Description

Cet événement est disponible pour un objet list box ou colonne de list box. Il est généré lorsqu'un clic se produit dans le pied de page d'une colonne de list box. Dans ce contexte, la commande `OBJECT Get pointer` retourne un pointeur vers la variable du pied de page sur lequel l'utilisateur a cliqué. L'événement est généré pour les clics gauche et droit.

Vous pouvez tester le nombre de clics effectués par l'utilisateur à l'aide de la commande `Clickcount`.

On getting focus

Code	Peut être appelé par	Définition
15	Zone 4D View Pro - Zone 4D Write Pro - Bouton - Case à cocher - Combo Box - Formulaire - Liste hiérarchique - Zone de saisie - List Box - Colonne de List Box - Zone de Plug-in - Indicateur de progression - Bouton Radio - Règle - Spinner - Stepper - Sous-formulaire - Zone Web	Un objet formulaire reçoit le focus

Description

Les événements `On Getting Focus` et `On loss Focus` permettent de détecter et de gérer le changement de focus pour les objets [focalisables](#).

Avec les [objets sous-formulaire](#), cet événement est généré dans la méthode de l'objet sous-formulaire lorsqu'il est vérifié. Il est envoyé à la méthode formulaire du sous-formulaire, ce qui signifie, par exemple, que vous pouvez gérer l'affichage des boutons de navigation dans le sous-formulaire en fonction du focus. A noter que les objets de sous-formulaire peuvent eux-mêmes avoir le focus.

Sur entête

Code	Peut être appelé par	Définition
5	Zone 4D Write Pro - Bouton - Grille de boutons - Case à cocher - Liste déroulante - Formulaire (formulaire liste uniquement) - Liste hiérarchique - Zone de saisie - Bouton image - Pop up menu image - Zone de plug-in - Indicateur de progression - Bouton radio - Règle - Spinner - Splitter - Stepper - Onglet	La zone d'en-tête du formulaire est sur le point d'être imprimée ou affichée.

Description

L'événement `On Header` est appelé lorsqu'un enregistrement est sur le point d'être affiché dans un formulaire liste affiché via `DISPLAY SELECTION` et `MODIFY SELECTION`.

Cet événement ne peut pas être sélectionné pour les formulaires projet, il est uniquement disponible avec les formulaires table.

Dans ce contexte, la séquence d'appels de méthodes et d'événements de formulaire suivante est déclenchée :

- Pour chaque objet de la zone d'en-tête :
 - Méthode objet avec l'événement `On Header`
 - Méthode formulaire avec l'événement `On Header`

Les enregistrements imprimés sont gérés à l'aide de l'événement `On Display Detail`.

L'appel d'une commande 4D qui affiche une boîte de dialogue à partir de l'événement `On Header` n'est pas autorisé et générera une erreur de syntaxe. Plus particulièrement, les commandes concernées sont : `ALERT`, `DIALOG`, `CONFIRM`, `Request`, `ADD RECORD`, `MODIFY RECORD`, `DISPLAY SELECTION`, et `MODIFY SELECTION`.

Sur clic entête

Code	Peut être appelé par	Définition
42	Zone 4D View Pro - List Box - Colonne de List Box	Un clic se produit dans un en-tête de colonne

Description

List Box

Cet événement est généré lorsqu'un clic se produit sur l'en-tête d'une colonne de list box. Dans ce cas, la commande `Self` vous permet d'identifier l'en-tête de la colonne sur laquelle vous avez cliqué.

Si la propriété [Sortable](#) a été sélectionnée pour la list box, vous pouvez décider d'autoriser ou non un tri standard de la colonne en passant la valeur 0 ou -1 dans la variable `$0` :

- Si `$0` est égal à 0, un tri standard est effectué.
- Si `$0` est égal à -1, un tri standard n'est pas effectué et l'en-tête n'affiche pas la flèche de tri. Le développeur peut toujours générer un tri de colonne basé sur des critères de tri personnalisés à l'aide du langage 4D.

Si la propriété [Sortable](#) n'est pas sélectionnée pour la list box, la variable `$0` n'est pas utilisée.

4D View Pro

Cet événement est généré lorsque l'utilisateur clique sur un en-tête de colonne ou de ligne dans un document 4D View Pro. Dans ce contexte, l'[objet événement](#) retourné par la commande `FORM Event` contient :

Propriété	Type	Description
code	entier long	42
description	Texte	"On Header Click"
objectName	Texte	Nom de la zone 4D View Pro
sheetName	Texte	Nom de la feuille de l'événement
range	object	Plage de cellule
sheetArea	entier long	L'emplacement de la feuille où l'événement a eu lieu : <ul style="list-style-type: none">• 0 : la zone de croisement entre le numéro de colonne/les en-têtes de lettre (en haut à gauche de la feuille)• 1 : les en-têtes de colonne (zone indiquant les numéros/lettres de colonnes)• 2 : les en-têtes de ligne (zone indiquant les numéros de ligne)

Exemple

```
If(FORM Event.code=On Header Click)
Case of
:(FORM Event.sheetArea=1)
    $values:=VP Get values(FORM Event.range)
:(FORM Event.sheetArea=2)
    VP SET CELL STYLE(FORM Event.range;New object("backColor";"gray"))
:(FORM Event.sheetArea=0)
    VP SET CELL STYLE(FORM Event.range;New object("borderBottom";\
        New object("color";"#800080";"style";vk line style thick)))
End case
End if
```

On Load

Code	Peut être appelé par	Définition
1	Zone 4D View Pro - Zone 4D Write Pro - Bouton - Grille de boutons - Case à cocher - Combo Box - Liste déroulante - Formulaire - Liste hiérarchique - Zone de saisie -List Box - Colonne de List Box - Bouton image - Pop up menu image - Zone de plug-in - Indicateur de progression - Bouton radio - Règle -Spinner - Splitter - Stepper - Sous-formulaire - Onglet - Zone Web	Le formulaire est sur le point d'être affiché ou imprimé

Description

Cet événement est déclenché lorsque le formulaire est en cours de chargement ou d'impression.

Tous les objets du formulaire (de n'importe quelle page) dont la propriété d'événement `On Load` est sélectionnée verront leur méthode objet appelée. Ensuite, si la propriété d'événement formulaire `On Load` est sélectionnée, la méthode formulaire sera appelée.

Les événements `On Load` et `On Unload` sont générés pour les objets s'ils sont activés à la fois pour les objets et pour le formulaire auquel appartiennent les objets. Si les événements sont activés pour les objets uniquement, ils ne se produiront pas; ces deux événements doivent également être activés au niveau du formulaire.

Sous-formulaire

L'événement `On Load` est généré à l'ouverture du sous-formulaire (cet événement doit également avoir été activé au niveau du formulaire parent pour être pris en compte). L'événement est généré avant ceux du formulaire parent. A noter également que, conformément aux principes de fonctionnement des événements de formulaire, si le sous-formulaire est placé sur une page autre que la page 0 ou 1, cet événement ne sera généré que lorsque cette page sera affichée (et non lorsque le formulaire sera affiché).

Voir aussi

[On Unload](#)

Sur chargement ligne

Code	Peut être appelé par	Définition
40	Formulaire	Lors de la saisie de l'utilisateur dans la liste, un enregistrement est chargé et un champ est édité

Description

L'événement `On Load Record` ne peut être utilisé que dans le contexte d'un formulaire de sortie. Il est déclenché lors de la saisie des données dans la liste, après la mise en surbrillance d'un enregistrement et le passage d'un champ en mode d'édition.

Cet événement ne peut pas être sélectionné pour les formulaires projet, il est uniquement disponible avec les formulaires table.

Sur clic long

Code	Peut être appelé par	Définition
39	Bouton	Un bouton est cliqué et le bouton de la souris reste enfoncé pendant un certain temps

Description

Cet événement est généré lorsqu'un bouton reçoit un clic et que le bouton de la souris est maintenu pendant un certain temps. En théorie, la durée de génération de cet événement est égale à la durée maximale séparant un double-clic, telle que définie dans les préférences système.

Cet événement peut être généré pour les styles de boutons suivants :

- [Barre d'outils](#)
- [Bevel](#)
- [Bevel arrondi](#)
- [OS X Gradient](#)
- [OS X Texture](#)
- [Office XP](#)
- [Aide](#)
- [Rond](#)
- [Personnalisé](#)

Cet événement est généralement utilisé pour afficher des pop-up menus en cas de longs clics sur les boutons. Si l'événement `On Clicked` est activé, il est généré si l'utilisateur relâche le bouton de la souris avant la limite de temps du "long clic".

Voir aussi

[Sur clic alternatif](#)

Sur perte focus

Code	Peut être appelé par	Définition
14	Zone 4D View Pro - Zone 4D Write Pro - Bouton - Case à cocher - Combo Box - Formulaire - Liste hiérarchique - Zone de saisie - List Box - Colonne de List Box - Zone de Plug-in - Indicateur de progression - Bouton Radio - Règle - Spinner - Stepper - Sous-formulaire - Zone Web	Un objet formulaire perd le focus

Description

Les événements `On Losing Focus` et `On Getting Focus` permettent de détecter et de gérer le changement de focus pour les objets **focalisables**.

Avec les [objets sous-formulaire](#), cet événement est généré dans la méthode de l'objet sous-formulaire lorsqu'il est vérifié. Il est envoyé à la méthode formulaire du sous-formulaire, ce qui signifie, par exemple, que vous pouvez gérer l'affichage des boutons de navigation dans le sous-formulaire en fonction du focus. A noter que les objets de sous-formulaire peuvent eux-mêmes avoir le focus.

Sur menu sélectionné

Code	Peut être appelé par	Définition
18	Formulaire	Un élément de menu a été choisi dans la barre de menu associée

Description

L'événement `On Menu Selected` est envoyé à la méthode formulaire lorsqu'une commande d'une barre de menus associée au formulaire est sélectionnée. Vous pouvez ensuite appeler la commande `Menu selected` pour tester le menu sélectionné.

Vous pouvez associer une barre de menus à un formulaire dans les propriétés du formulaire. Les menus d'une barre de menus formulaire sont ajoutés à la barre de menus courante lorsque le formulaire est affiché en tant que formulaire de sortie dans l'environnement Application.

Sur début survol

Code	Peut être appelé par	Définition
35	Zone 4D Write Pro - Bouton - Grille de boutons - Case à cocher - Combo Box - Liste déroulante -Formulaire - Liste hiérarchique - Zone de saisie - List Box - Bouton image - Pop up menu image - Zone de plug-in - Indicateur de progression - Bouton radio - Règle -Spinner - Splitter - Stepper - Onglet	Le curseur de la souris entre dans la zone graphique d'un objet

Description

Cet événement est généré une fois, lorsque le curseur de la souris entre dans la zone graphique d'un objet du formulaire.

L'événement `On Mouse Enter` met à jour les variables système `MouseX` et `MouseY`.

Les objets rendus invisibles à l'aide de la commande `OBJECT SET VISIBLE` ou de la propriété `Visibility` ne génèrent pas cet événement.

Appeler la pile

Si l'événement `On Mouse Enter` a été coché pour le formulaire, il est généré pour chaque objet de formulaire. S'il est vérifié pour un objet, il n'est généré que pour cet objet. Lorsqu'il existe des objets superposés, l'événement est généré par le premier objet capable de le gérer qui se trouve en allant de haut en bas.

Voir aussi

- [Sur survol](#)
- [Sur fin survol](#)

Sur fin survol

Code	Peut être appelé par	Définition
36	Zone 4D Write Pro - Bouton - Grille de boutons - Case à cocher - Combo Box - Liste déroulante -Formulaire - Liste hiérarchique - Zone de saisie - List Box - Bouton image - Pop up menu image - Zone de plug-in - Indicateur de progression - Bouton radio - Règle -Spinner - Splitter - Stepper - Onglet	Le curseur de la souris quitte la zone graphique d'un objet

Description

Cet événement est généré une fois, lorsque le curseur de la souris quitte la zone graphique d'un objet.

L'événement `On Mouse Leave` met à jour les variables système `MouseX` et `MouseY`.

Les objets rendus invisibles à l'aide de la commande `OBJECT SET VISIBLE` ou de la propriété `Visibility` ne génèrent pas cet événement.

Appeler la pile

Si l'événement `On Mouse Leave` a été coché pour le formulaire, il est généré pour chaque objet de formulaire. S'il est vérifié pour un objet, il n'est généré que pour cet objet. Lorsqu'il existe des objets superposés, l'événement est généré par le premier objet capable de le gérer qui se trouve en allant de haut en bas.

Voir aussi

- [Sur survol](#)
- [Sur fin survol](#)

Sur survol

Code	Peut être appelé par	Définition
37	Zone 4D Write Pro - Bouton - Grille de boutons - Case à cocher - Combo Box - Liste déroulante -Formulaire - Liste hiérarchique - Zone de saisie - List Box - Bouton image - Pop up menu image - Zone de plug-in - Indicateur de progression - Bouton radio - Règle -Spinner - Splitter - Stepper - Onglet	Le curseur de la souris se déplace d'au moins un pixel OU une touche de modification (Shift, Alt/Option, Shift Lock) a été pressée

Description

Cet événement est généré :

- lorsque le curseur de la souris se déplace d'au moins un pixel
- OU lorsque l'on presse sur une touche de modification (Ctrl, Alt/Option, Verr Maj). Cela permet de gérer les opérations de glisser-déposer de type copier ou déplacer.

Si l'événement est coché pour un objet uniquement, il est généré uniquement lorsque le curseur se trouve dans la zone graphique de l'objet.

L'événement `On Mouse Move` met à jour les variables système `MouseX` et `MouseY`.

Les objets rendus invisibles à l'aide de la commande `OBJECT SET VISIBLE` ou de la propriété `Visibility` ne génèrent pas cet événement.

Appeler la pile

Si l'événement `On Mouse Move` a été coché pour le formulaire, il est généré pour chaque objet de formulaire. S'il est vérifié pour un objet, il n'est généré que pour cet objet. Lorsqu'il existe des objets superposés, l'événement est généré par le premier objet capable de le gérer qui se trouve en allant de haut en bas.

Voir aussi

- [Sur début survol](#)
- [Sur fin survol](#)

On Mouse Up

Code	Peut être appelé par	Définition
2	Zone de saisie de type image	L'utilisateur vient de relâcher le bouton gauche de la souris dans un objet Image

Description

L'événement `On Mouse Up` est généré lorsque l'utilisateur vient de relâcher le bouton gauche de la souris tout en faisant glisser une image. Cet événement est utile, par exemple, lorsque vous souhaitez que l'utilisateur puisse déplacer, redimensionner ou dessiner des objets dans une zone SVG.

Lorsque l'événement `On Mouse Up` est généré, vous pouvez obtenir les coordonnées locales où le bouton de la souris a été relâché. Ces coordonnées sont renvoyées dans les variables système `MouseX` et `MouseY`. Les coordonnées sont exprimées en pixels par rapport à l'angle supérieur gauche de l'image (0,0).

Lorsque vous utilisez cet événement, vous devez également utiliser la commande `Is waiting mouse up` pour gérer les cas où le "gestionnaire d'état" du formulaire est désynchronisé, c'est-à-dire lorsque l'événement `On Mouse Up` n'est pas reçu après un clic. C'est le cas par exemple lorsqu'une boîte de dialogue d'alerte s'affiche au-dessus du formulaire alors que le bouton de la souris n'a pas été relâché. Pour plus d'informations et pour voir un exemple d'utilisation de l'événement `On Mouse Up`, veuillez vous référer à la description de la commande `Is waiting mouse up`.

Si l'option `Draggable` est activée pour l'objet image, l'événement `On Mouse Up` n'est jamais généré.

Sur ouverture corps

Code	Peut être appelé par	Définition
25	Formulaire - List Box	Le formulaire détaillé associé au formulaire de sortie ou à la list box est sur le point d'être ouvert.

Description

L'événement `On Open Detail` peut être utilisé dans les contextes suivants :

- Formulaires de sortie : un enregistrement est sur le point d'être affiché dans le formulaire détaillé associé au formulaire de sortie. Cet événement ne peut pas être sélectionné pour les formulaires projet, il est uniquement disponible avec les formulaires table.
- List box [de type sélection](#) : Cet événement est généré lorsqu'un enregistrement est sur le point d'être affiché dans le formulaire détaillé associé à une list box de type sélection (et avant l'ouverture de ce formulaire).

Numéro de ligne affiché

La commande 4D `Displayed line number` fonctionne avec l'événement formulaire `On Open Detail`. Elle retourne le numéro de la ligne en cours de traitement tandis qu'une liste d'enregistrements ou de lignes de list box s'affiche à l'écran.

On Open External Link

Code	Peut être appelé par	Définition
52	Zone Web	Une URL externe a été ouverte dans le navigateur

Description

Cet événement est généré lorsque le chargement d'une URL a été bloqué par la zone Web et que l'URL a été ouverte avec le navigateur système actuel, en raison d'un filtre mis en place via la commande `WA SET EXTERNAL LINKS FILTERS`.

Vous pouvez identifier l'URL bloquée à l'aide de la commande `WA Get last filtered URL`.

Voir aussi

[On URL Filtering](#)

Sur appel extérieur

Code	Peut être appelé par	Définition
10	Formulaire	Le formulaire a reçu un appel <code>POST OUTSIDE CALL</code>

Description

Cet événement est appelé lorsque le formulaire est appelé à partir d'un autre processus via la commande `POST OUTSIDE CALL`.

L'événement `On Outside Call` modifie le contexte de saisie du formulaire. En particulier si un champ était en cours de modification, l'événement `On Data Change` est généré.

Sur changement page

Code	Peut être appelé par	Définition
56	Formulaire	La page courante du formulaire est modifiée

Description

Cet événement n'est disponible qu'au niveau du formulaire (il est appelé dans la méthode formulaire). Il est généré à chaque fois que la page courante du formulaire change (suite à un appel à la commande `FORM GOTO PAGE` ou à une action de navigation standard).

A noter qu'il est généré après le chargement complet de la page, c'est-à-dire une fois tous les objets qu'elle contient initialisés, y compris les [zones Web](#).

La seule exception concerne les zones 4D View Pro, pour lesquelles vous devez appeler l'événement spécifique [On VP Ready](#).

L'événement `On Page Change` est utile pour exécuter du code qui nécessite que tous les objets soient préalablement initialisés. Vous pouvez également l'utiliser pour optimiser l'application en exécutant du code (par exemple, une recherche) uniquement après l'affichage d'une page spécifique du formulaire et pas seulement dès que la page 1 est chargée. Si l'utilisateur ne va pas sur cette page, le code n'est pas exécuté.

Sur appel zone du plug in

Code	Peut être appelé par	Définition
19	Formulaire - Zone de Plug-in	Un objet externe a demandé que sa méthode objet soit exécutée

Description

L'événement est généré lorsqu'un plug-in a demandé à sa zone de formulaire d'exécuter la méthode objet associée.

On Printing Break

Code	Peut être appelé par	Définition
6	Zone 4D Write Pro - Bouton - Grille de boutons - Case à cocher - Combo Box - Liste déroulante - Formulaire - Liste hiérarchique - Zone de saisie - Bouton image - Pop up menu image - Zone de plug-in - Indicateur de progression - Bouton radio - Règle - Spinner - Splitter - Stepper - Onglet	L'une des zones de rupture du formulaire est sur le point d'être imprimée

Description

L'événement `On Printing Break` ne peut être utilisé que dans le contexte d'un formulaire de sortie. Il est déclenché chaque fois qu'une zone de rupture du formulaire de sortie est sur le point d'être imprimée, afin que vous puissiez évaluer les valeurs de rupture, par exemple.

Cet événement fait généralement suite à un appel à la commande `Subtotal`.

Cet événement ne peut pas être sélectionné pour les formulaires projet, il est uniquement disponible avec les formulaires table.

On Printing Detail

Code	Peut être appelé par	Définition
23	Zone 4D Write Pro - Bouton - Grille de boutons - Case à cocher - Combo Box - Liste déroulante -Formulaire - Liste hiérarchique - Zone de saisie - Bouton image - Pop up menu image - Zone de plug-in - Indicateur de progression - Bouton radio - Règle -Spinner - Splitter - Stepper - Onglet	La zone détaillée du formulaire est sur le point d'être imprimée

Description

L'événement `On Printing Detail` ne peut être utilisé que dans le contexte d'un formulaire de sortie. Il est déclenché lorsque la zone de détail du formulaire de sortie est sur le point d'être imprimée, par exemple suite à un appel à la commande `Print form`.

La commande `Print form` génère un seul événement `On Printing Detail` pour la méthode formulaire.

Cet événement ne peut pas être sélectionné pour les formulaires projet, il est uniquement disponible avec les formulaires table.

On Printing Footer

Code	Peut être appelé par	Définition
7	Zone 4D Write Pro - Bouton - Grille de boutons - Case à cocher - Combo Box - Liste déroulante -Formulaire - Liste hiérarchique - Zone de saisie - Bouton image - Pop up menu image - Zone de plug-in - Indicateur de progression - Bouton radio - Règle -Spinner - Splitter - Stepper - Onglet	La zone de pied du formulaire est sur le point d'être imprimée

Description

L'événement `On Printing Footer` ne peut être utilisé que dans le contexte d'un formulaire de sortie. Il est déclenché lorsqu'une zone de pied du formulaire de sortie est sur le point d'être imprimée, afin que vous puissiez évaluer les valeurs du pied.

Cet événement peut être déclenché dans le cadre d'une commande `PRINT SELECTION`.

Cet événement ne peut pas être sélectionné pour les formulaires projet, il est uniquement disponible avec les formulaires table.

Sur redimensionnement

Code	Peut être appelé par	Définition
29	Formulaire	La fenêtre du formulaire est redimensionnée ou l'objet sous-formulaire est redimensionné (dans ce cas, l'événement est généré dans la méthode formulaire du sous-formulaire)

Description

Cet événement est appelé :

- lorsque la fenêtre du formulaire est redimensionnée,
- dans le contexte de sous-formulaires, lorsque la taille de l'objet de sous-formulaire du formulaire parent a changé.
Dans ce cas, cet événement est envoyé à la méthode formulaire du sous-formulaire.

Sur déplacement ligne

Code	Peut être appelé par	Définition
34	List Box de type tableau - Colonne de List Box	Une ligne de list box est déplacée par l'utilisateur par glisser-déposer

Description

Cet événement est généré lorsqu'une ligne de list box (de [type tableau uniquement](#)) est déplacée par l'utilisateur à l'aide du glisser-déposer ([si autorisé](#)). Il n'est pas généré si la ligne est glissée puis déposée à son emplacement initial.

La commande `LISTBOX MOVED ROW NUMBER` retourne la nouvelle position de la ligne.

On Row Resize

Code	Peut être appelé par	Définition
60	Zone 4D View Pro	La hauteur d'une ligne est modifiée par un utilisateur avec la souris

Description

Cet événement est généré lorsque la hauteur d'une ligne est modifiée par un utilisateur dans un document 4D View Pro. Dans ce contexte, l'[objet événement](#) retourné par la commande `FORM Event` contient :

Propriété	Type	Description
code	entier long	60
description	Texte	"On Row Resize"
objectName	Texte	Nom de la zone 4D View Pro
sheetName	Texte	Nom de la feuille de l'événement
range	object	Plage de cellules des lignes dont les hauteurs ont changé
header	boolean	"True" si la ligne de la colonne d'en-tête (première ligne) est redimensionnée, sinon false

Exemple

```
If(FORM Event.code=On Row Resize)
    VP SET CELL STYLE(FORM Event.range;New object("vAlign";vk vertical align top))
End if
```

Sur défilement

Code	Peut être appelé par	Définition
59	Zone de saisie de type image - List Box	L'utilisateur fait défiler le contenu d'un objet image ou d'une list box à l'aide de la souris ou du clavier.

Description

Cet événement peut être généré dans le contexte d'une entrée d'image ou d'une list box.

Il est déclenché après tout autre événement utilisateur lié à l'action de défilement ([On Clicked](#), [On After Keystroke](#), etc.). L'événement est uniquement généré dans la méthode objet (pas dans la méthode formulaire).

L'événement est déclenché lorsque le défilement est le résultat d'une action de l'utilisateur : à l'aide des barres de défilement et/ou des curseurs, à l'aide de la molette de la souris ou du [clavier](#). Il n'est pas généré lors du défilement de l'objet en raison de l'exécution de la commande `OBJECT SET SCROLL POSITION`.

Entrée d'image

L'événement est généré dès qu'un utilisateur fait défiler une image dans l'entrée d'image (champ ou variable) qui la contient. Vous pouvez faire défiler le contenu d'une zone d'image lorsque la taille de la zone est plus petite que son contenu et que le [format d'affichage](#) est "Tronqué (non centré)".

List box

L'événement est généré dès qu'un utilisateur fait défiler les lignes ou les colonnes de la list box.

Sur nouvelle sélection

Code	Peut être appelé par	Définition
31	Zone 4D View Pro - Zone 4D Write Pro - Formulaire - Liste hiérarchique - Zone de saisie - List Box	La sélection faite dans l'objet est modifiée

Description

Cet événement peut être généré dans différents contextes.

4D View Pro

La sélection courante de lignes ou de colonnes est modifiée. Dans ce contexte, l' [objet événement](#) retourné par la commande `FORM Event` contient :

Propriété	Type	Description
code	entier long	31
description	Texte	"On Selection Change"
objectName	Texte	Nom de la zone 4D View Pro
sheetName	Texte	Nom de la feuille de l'événement
oldSelections	object	Plage de cellules avant changement
newSelections	object	Plage de cellules après changement

Exemple

```
If(FORM Event.code=0n Selection Change)
    VP SET CELL STYLE(FORM Event.oldSelections;New object("backColor";Null))
    VP SET CELL STYLE(FORM Event.newSelections;New object("backColor";"red"))
End if
```

Formulaire liste

L'enregistrement courant ou la sélection courante de lignes est modifié(e) sous dans un formulaire liste.

Liste hiérarchique

Cet événement est généré à chaque fois que la sélection faite dans la liste hiérarchique est modifiée après un clic de souris ou une frappe.

Zone de saisie et 4D Write Pro

La sélection de texte ou la position du curseur dans la zone est modifiée suite à un clic ou une frappe.

List box

Cet événement est généré chaque fois que la sélection courante de lignes ou de colonnes de la list box est modifiée.

Sur minuteur

Code	Peut être appelé par	Définition
27	Formulaire	Le nombre de graduations défini par la commande <code>SET TIMER</code> est passé

Description

Cet événement est généré uniquement si la méthode formulaire contient un appel à la commande `SET TIMER` réalisé antérieurement.

Lorsque la propriété d'événement formulaire `On Timer` est sélectionnée, seule la méthode formulaire recevra l'événement, aucune méthode objet ne sera appelée.

On Unload

Code	Peut être appelé par	Définition
24	Zone 4D View Pro - Zone 4D Write Pro - Bouton - Grille de boutons - Case à cocher - Combo Box - Liste déroulante - Formulaire - Liste hiérarchique - Zone de saisie -List Box - Colonne de List Box - Bouton image - Pop up menu image - Zone de plug-in - Indicateur de progression - Bouton radio - Règle -Spinner - Splitter - Stepper - Sous-formulaire - Onglet - Zone Web	Le formulaire est sur le point d'être quitté et généré

Description

Cet événement est déclenché lorsque le formulaire est généré.

Tous les objets du formulaire (de n'importe quelle page) dont la propriété d'événement `On Unload` est sélectionnée verront leur méthode objet appelée. Ensuite, si la propriété d'événement formulaire `On Unload` est sélectionnée, la méthode formulaire sera appelée.

Les événements `On Load` et [`On Unload`] sont générés pour les objets s'ils sont activés à la fois pour les objets et pour le formulaire auquel appartiennent les objets. Si les événements sont activés pour les objets uniquement, ils ne se produiront pas; ces deux événements doivent également être activés au niveau du formulaire.

Sous-formulaire

L'événement `On Unload` est généré à la fermeture du sous-formulaire (cet événement doit également avoir été activé au niveau du formulaire parent pour être pris en compte). L'événement est généré avant ceux du formulaire parent.
The event is generated before those of the parent form.

Voir aussi

[On Load](#)

On URL Filtering

Code	Peut être appelé par	Définition
51	Zone Web	Une URL a été bloquée par la zone Web

Description

Cet événement est généré lorsque le chargement d'une URL est bloqué par la zone Web en raison d'un filtre configuré à l'aide de la commande `WA SET URL FILTERS`.

Vous pouvez identifier l'URL bloquée à l'aide de la commande `WA Get last filtered URL`.

Voir aussi

[On Open External Link](#)

On URL Loading Error

Code	Peut être appelé par	Définition
50	Zone Web	Une erreur s'est produite lors du chargement de l'URL

Description

Cet événement est généré lorsqu'une erreur est détectée lors du chargement d'une URL.

Vous pouvez appeler la commande `WA GET LAST URL ERROR` afin d'obtenir des informations sur l'erreur.

Voir aussi

[On Open External Link](#)

On URL Resource Loading

Code	Peut être appelé par	Définition
48	Zone Web	Une nouvelle ressource est chargée dans la zone Web

Description

Cet événement est généré chaque fois qu'une nouvelle ressource (image, cadre, etc.) est chargée sur la page Web courante.

La variable [Progression](#) associée à la zone vous permet de connaître l'état du chargement.

Voir aussi

[On Open External Link](#)

Sur validation

Code	Peut être appelé par	Définition
3	Zone 4D Write Pro - Bouton - Grille de boutons - Case à cocher - Combo Box - Liste déroulante - Formulaire - Liste hiérarchique - Zone de saisie -List Box - Colonne de List Box - Bouton image - Pop up menu image - Zone de plug-in - Indicateur de progression - Bouton radio - Règle -Spinner - Splitter - Stepper - Sous-formulaire - Onglet	La saisie des données d'enregistrement a été validée

Description

Cet événement est déclenché lorsque la saisie des données d'enregistrement a été validée, par exemple après un appel de la commande `SAVE RECORD` ou après une `action standard accept`.

Sous-formulaire

L'événement `On Validate` est déclenché lorsque la saisie de données est validée dans le sous-formulaire.

On VP Range Changed

Code	Peut être appelé par	Définition
61	Zone 4D View Pro	La plage de cellules 4D View Pro a changé (ex : un calcul de formule, une valeur supprimée d'une cellule, etc.)

Description

Cet événement est généré lorsqu'un changement intervient dans une plage de cellules dans le document 4D View Pro.

L'objet retourné par la commande FORM Event contient :

Propriété	Type	Description
objectName	Texte	Nom de la zone 4D View Pro
code	entier long	On VP Range Changed
description	Texte	"On VP Range Changed"
sheetName	Texte	Nom de la feuille de l'événement
range	object	Plage de cellules liées au changement
changedCells	object	Plage contenant uniquement les cellules modifiées. Il peut s'agir d'une gamme combinée.
action	Texte	Le type d'opération générant l'événement : <ul style="list-style-type: none">• "clear" - A clear range value operation• "dragDrop" - Une opération de glisser-déposer• "dragFill" - Une opération de remplissage par glisser• "evaluFormula" - Définition d'une formule dans une plage de cellules spécifiée• "coller" - Une opération de collage• "setArrayFormula" - Définition d'une formule dans une plage de cellules spécifiée• "sort" - Tri d'une plage de cellules

Voir également [On After Edit](#).

On VP Ready

Code	Peut être appelé par	Définition
9	Zone 4D View Pro	Le chargement de la zone 4D View Pro est terminé

Description

Cet événement est généré lorsque le chargement de la zone 4D View Pro est terminé.

Vous devez utiliser cet événement pour écrire le code d'initialisation de la zone. Tout code d'initialisation de zone 4D View Pro, pour le chargement ou la lecture de valeurs issues de la zone ou contenues dans la zone, doit se trouver dans l'événement formulaire `On VP Ready` de la zone. Cet événement formulaire est déclenché une fois le chargement de la zone terminé. Tester cet événement vous garantit que le code sera exécuté dans un contexte valide. Une erreur est retournée si une commande 4D View Pro est appelée avant la génération de l'événement formulaire `On VP Ready`.

Les zones 4D View Pro sont chargées de manière asynchrone dans les formulaires 4D. Cela signifie que l'événement standard `On load` form ne peut pas être utilisé pour le code d'initialisation de 4D View Pro, car il pourrait être exécuté avant la fin du chargement de la zone. `On VP Ready` est toujours généré après `On load`.

On Window Opening Denied

Code	Peut être appelé par	Définition
53	Zone Web	Une fenêtre pop-up a été bloquée

► Historique

Description

Cet événement est généré lorsque l'ouverture d'une fenêtre pop-up est bloquée par la zone Web. Les zones Web de 4D ne permettent pas l'ouverture de fenêtres contextuelles.

Vous pouvez identifier l'URL bloquée à l'aide de la commande `WA Get last filtered URL`.

Cet événement est également déclenché après une opération de déposer dans la zone Web (avec des [moteurs](#) système Windows et intégrés) si l'option [Glisser Déposer](#) est également activée pour la zone. Vous pouvez accepter l'action de déposer en appelant :

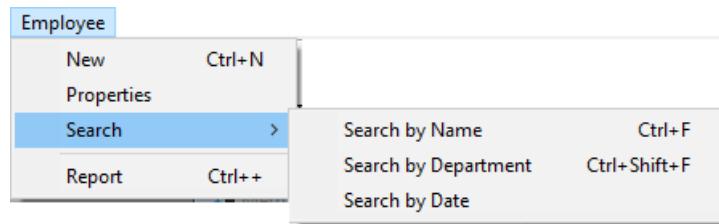
```
//méthode objet zone web
If (FORM Event.code=On Window Opening Denied)
    WA OPEN URL(*; "WebArea"; WA Get last filtered URL(*; "WebArea"))
    // ou UrlVariable:=WA Get last filtered URL(*; "WebArea")
    // où UrlVariable est la variable URL associé à la zone web
End if
```

Voir aussi

[On Open External Link](#)

Aperçu

4D vous permet de créer des barres de menus et des menus pour vos applications 4D. Etant donné que les menus sont une fonctionnalité standard de toute application, leur présence dans votre base facilite son utilisation.



Une barre de menus est un groupe de menus qui peuvent être affichés dans le même écran. Chaque menu d'une barre de menus peut posséder plusieurs commandes dont certaines peuvent faire appel à des sous-menus en cascade (on parle alors de sous-menus hiérarchiques). Lorsque l'utilisateur choisit une commande de menu ou de sous-menu, il appelle une méthode projet ou une action standard qui réalise une opération.

Vous pouvez disposer de différentes barres de menus dans la même application. Par exemple, vous pouvez utiliser une barre de menus pour les opérations standard de votre base de données et une autre qui n'est active que pour la réalisation d'états. Une barre de menus peut contenir des commandes destinées à la saisie d'enregistrements. La barre de menus qui apparaît avec le formulaire de saisie peut contenir le même menu, mais ses commandes peuvent être désactivées car inutiles pour la saisie dans le formulaire.

Vous pouvez utiliser le même menu dans plusieurs barres de menus et menus, ou ne pas l'attacher et le gérer uniquement par programmation (on parle dans ce cas de menu indépendant).

Lorsque vous créez des menus, il est utile de garder à l'esprit les deux règles suivantes :

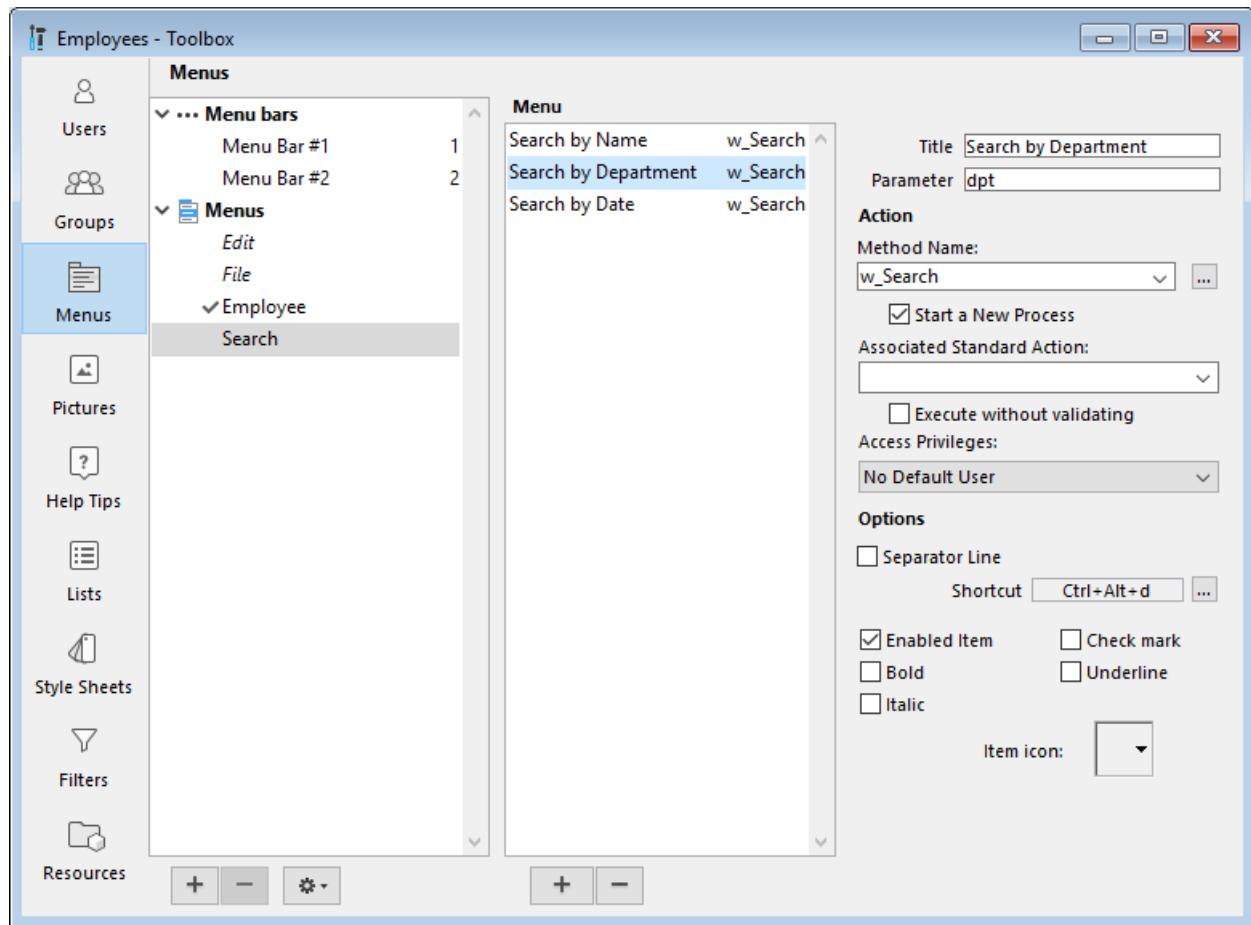
- N'utilisez les menus que pour des tâches qui leurs sont adaptées : Les commandes de menus doivent réaliser des tâches telles que l'ajout d'un enregistrement, les recherches ou les impressions.
- Groupez les commandes de menus par fonctions : l'utilisateur doit pouvoir être capable de s'orienter dans un ordre logique de menus. Par exemple, toutes les commandes de menu qui permettent de naviguer dans la base doivent être placées dans le même menu.

Pour créer des menus et des barres de menus, vous pouvez utiliser soit :

- l'éditeur de menus,
- les commandes du langage,
- un mélange des deux.

Éditeur de menus

L'éditeur de menus est accessible via le bouton Menus de la Boîte à outils.



Les barres de menus et les menus sont affichés sous forme de deux éléments d'une même liste hiérarchique, dans la partie gauche de la fenêtre. Chaque menu peut être attaché à une barre de menus ou à un autre menu. Dans le deuxième cas, le menu devient un sous-menu.

4D affecte des numéros de barre de menus séquentiellement -- Menu Bar #1 apparaît en premier. Vous pouvez renommer des barres de menu mais vous ne pouvez pas modifier leur numéro. Ces derniers sont utilisés par les commandes du langage.

Créer des menus et des barres de menus

Les barres de menus peuvent être définies :

- dans l'éditeur de menus de la fenêtre de Boîte à outils 4D. Dans ce cas, les menus et barres de menus sont stockés dans la structure de l'application.
- dynamiquement, à l'aide des commandes du langage depuis le thème "Menus". Dans ce cas, les menus et barres de menus ne sont pas stockés, ils existent uniquement dans la mémoire.

Vous pouvez combiner les deux fonctionnalités et utiliser les menus créés dans la structure comme templates pour définir des menus dans la mémoire.

Barre de menu par défaut

Une application personnalisée doit contenir au moins une barre de menu avec un menu. La barre de menus par défaut (Barre n°1) comporte des menus standard et une commande de retour au mode Développement. La barre de menus par défaut (Barre n°1) comporte des menus standard et une commande de retour au mode Développement.

This allows the user to access the Application environment as soon as the project is created. La barre de menus n°1 est automatiquement appelée lorsque la commande Tester l'application est sélectionnée dans le menu Exécution.

La barre de menus par défaut contient trois menus : Fichier, Edition et Mode.

- Fichier : ce menu comporte uniquement la commande Quitter. L'action automatique *Quitter* est associée à la commande, ce qui a pour effet de provoquer la fermeture de l'application.
- Edition : menu standard et entièrement modifiable. Editing functions such as copy, paste, etc. are defined using standard actions.
- Mode : par défaut, ce menu contient la commande Retour au mode Développement, permettant de sortir du mode Application.

Les libellés apparaissent *en caractères italiques* car il s'agit de références et non de textes en dur. Pour plus d'informations sur ce point, reportez-vous à la section [Utiliser des références dans les titres de menus](#).

Vous pouvez modifier cette barre de menus comme vous le souhaitez ou créer des barres de menus supplémentaires.

Créer des menus

A l'aide de l'éditeur de menus

1. Sélectionnez la ligne de menu que vous souhaitez créer et cliquez sur le bouton d'ajout sous la zone de liste des barres de menu. OU Choisissez la commande Créer une nouvelle barre de menus ou Créer un nouveau menu dans le menu contextuel de la liste ou dans le menu d'options situé sous la liste. Si vous avez créé une barre de menu, une nouvelle barre de menus apparaît dans la liste, contenant les menus par défaut (Fichier et Edition).
2. (Facultatif) Effectuez un double-clic sur le nom du menu/de la barre de menus afin de le rendre éditable et saisissez un nom personnalisé. OU Saisissez le nom personnalisé dans la zone "Titre". Les noms des barres de menu doivent être uniques. Ils peuvent comporter jusqu'à 31 caractères. You can enter the name as "hard coded" or enter a reference (see [information about the Title property](#)).

A l'aide du langage 4D

Utilisez la commande de `Create menu` pour créer une nouvelle barre de menu ou une référence de menu (*MenuRef*) en mémoire.

Lorsque les menus sont gérés par des références *MenuRef*, il n'y a pas de différence en soi entre un menu et une barre de menus. Dans les deux cas, il s'agit d'une liste d'éléments. Seul leur utilisation diffère. Dans le cas d'une barre de menus, chaque élément correspond à un menu lui-même composé d'éléments.

Créer un menu permet de créer des menus vides (à remplir à l'aide de l'option APPEND MENU ITEM ou INSERT MENU ITEM) ou des menus créés à partir de menus conçus dans l'éditeur de menus.

Ajouter des lignes

Pour chacun des menus, vous devez ajouter les commandes qui apparaissent lorsque le menu est déroulé. Vous pouvez insérer des lignes qui seront associées à des méthodes ou à des actions standard, ou rattacher d'autres menus (sous-menus).

A l'aide de l'éditeur de menus

Pour ajouter une ligne de menu :

1. Dans la liste des menus source, sélectionnez le menu auquel vous souhaitez ajouter une commande. Si le menu contient déjà des commandes, elles seront affichées dans la liste centrale. Si vous souhaitez insérer la nouvelle commande, sélectionnez celle que vous souhaitez voir apparaître ci-dessus. Il est toujours possible de réorganiser le menu ultérieurement par glisser-déposer.
2. Choisissez Add an item to menu "MenuName" dans le menu d'options de l'éditeur ou depuis le menu contextuel (clic droit dans la liste centrale). OU Cliquez sur le bouton Ajouter situé sous la liste centrale. 4D ajoute une nouvelle ligne avec le nom par défaut "Ligne X", où X représente le nombre de lignes déjà créées.
3. Double-cliquez sur le nom de la commande pour passer en mode édition et saisissez un nom personnalisé. OU Saisissez le nom personnalisé dans la zone "Titre". Il peut comporter jusqu'à 31 caractères. Vous pouvez saisir le nom comme "en dur" ou saisir une référence (voir ci-dessous).

A l'aide du langage 4D

Utilisez INSERT MENU ITEM ou APPEND MENU ITEM pour insérer ou ajouter des lignes de menu dans les références de menu existantes.

Supprimer des menus et des lignes de menus

A l'aide de l'éditeur de menus

Vous pouvez supprimer une barre de menus, un menu ou une ligne de menu à tout moment. A noter qu'il n'existe qu'une seule référence d'un menu ou barre de menus. Lorsqu'un menu est rattaché à différentes barres ou différents menus, toute modification ou suppression effectuée dans ce menu est immédiatement reportée dans toutes les instances de ce menu. Supprimer un menu supprimera uniquement une référence. Lorsque vous supprimez la dernière référence d'un menu, 4D affiche une alerte.

Pour supprimer une barre de menus, un menu ou une ligne de menu, vous disposez de deux possibilités :

- Sélectionner l'élément à supprimer et de cliquer sur le bouton de suppression situé sous la liste.
- ou, utiliser la commande Supprimer ... dans le menu contextuel ou le menu d'options de l'éditeur.

Il est impossible de supprimer Menu Bar #1.

A l'aide du langage 4D

Utilisez la commande SUPPRIMER LIGNE DE MENU pour supprimer une ligne de la barre de menus. Utilisez la commande EFFACER MENU pour ne pas charger le menu de la mémoire.

Rattacher des menus

Une fois que vous avez créé un menu, vous pouvez le rattacher à une ou plusieurs barres de menus ou à un ou plusieurs autres menus (sous-menus).

Les sous-menus permettent de regrouper des fonctions thématiques à l'intérieur d'un même menu. Les sous-menus et

leurs lignes peuvent disposer des mêmes attributs que les menus (actions, méthodes, raccourcis, icônes, etc.). Les lignes du sous-menu conservent leurs caractéristiques et leurs propriétés, le fonctionnement du sous-menu est identique à celui d'un menu standard.

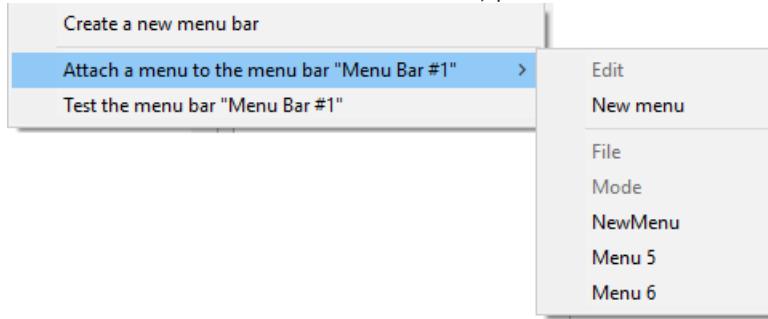
Vous pouvez créer des sous-menus de sous-menus sur une profondeur virtuellement illimitée. A noter toutefois que pour des raisons d'ergonomie d'interface, il n'est généralement pas conseillé de dépasser deux niveaux de sous-menus.

A l'exécution, si un menu rattaché est modifié par programmation, toute autre élément du menu reflétera ces modifications.

A l'aide de l'éditeur de menus

Un menu peut être attaché à une barre de menus ou à un autre menu.

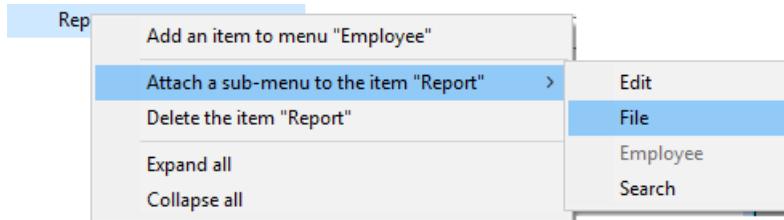
- Pour rattacher un menu à une barre de menus : faites un clic droit sur la barre de menus et sélectionnez `Attach a menu to the menu bar "nom de la barre"`, puis choisissez le menu à rattacher à la barre de menus :



Vous pouvez également sélectionner une barre

de menus puis cliquez sur le bouton des options situé sous la liste.

- Pour rattacher un menu à un autre menu : sélectionnez le menu dans la partie gauche puis faites un clic droit sur la ligne de menus et sélectionnez `Attach a sub-menu to the item "Report"`, puis choisissez le menu à utiliser comme sous-menu :



Vous pouvez également sélectionner une ligne

de menus puis cliquez sur le bouton des options situé sous la liste. Le menu que vous êtes en train de rattacher deviendra un sous-menu. Le titre de la ligne est maintenu (le nom initial du sous-menu est ignoré), mais il peut être modifié.

Détacher des menus

Vous pouvez à tout moment détacher un menu d'une barre ou un sous-menu d'un menu. Le menu détaché n'est alors plus disponible dans la barre ou le sous-menu, mais reste présent dans la liste des menus.

Pour détacher un menu, cliquez avec le bouton droit dans la liste centrale sur le menu ou le sous-menu à détacher puis choisissez la commande `Détacher le menu "nom menu"` de la barre `"nom barre"` ou `Détacher le sous-menu de la ligne "nom ligne"`

A l'aide du langage 4D

Etant donné qu'il n'y a pas de différence entre les menus et les barres de menus dans le langage de 4D, rattacher des menus ou des sous-menus se fait de la même manière : utilisez le paramètre `sous-menu` de la commande `APPEND MENU ITEM` pour rattacher un menu à une barre de menu ou à un autre menu.

Propriétés des menus

Vous pouvez définir plusieurs propriétés à partir des lignes de menu, telles que des actions, des styles de police, les lignes de séparation, des raccourcis clavier ou des icônes.

Titre de menu

La propriété Title contient le libellé d'un menu ou d'une ligne de menu, tel qu'il sera affiché dans l'interface de l'application.

Dans l'éditeur de menus, vous pouvez saisir directement le libellé "en dur". Vous pouvez également saisir une référence pour une variable ou un élément XLIFF, ce qui facilitera la mise à jour et la traduction des applications. Vous pouvez utiliser types de références suivants :

- Une référence à une ressource XLIFF, du type :xliiff:MonLibellé. Pour plus d'informations sur les références XLIFF, reportez-vous à la section *Annexe B : Architecture XLIFF du Mode Développement 4D*.
- Un nom de variable interprocess suivi d'un chiffre, par exemple :<>vlang,3. Il suffit de changer le contenu de cette variable pour modifier le libellé du menu lors de son affichage. Dans ce cas, le libellé fera appel à une ressource XLIFF. La valeur contenue dans la variable <>vlang correspond à l'attribut *id* de l'élément *group*. La seconde valeur (3 dans cet exemple) désigne l'attribut *id* de l'élément *trans-unit*.

En utilisant le langage 4D, vous définissez la propriété Title à l'aide du paramètre *itemText* des commandes APPEND MENU ITEM , INSERT MENU ITEM , et SET MENU ITEM .

Caractères de contrôle

Il est possible de définir les propriétés des lignes de menus en insérant des caractères de contrôle ("métacaractères") directement dans les libellés des commandes de menus. Cette possibilité est utile pour la création de menus par programmation. Par exemple, vous pouvez associer le raccourci-clavier Ctrl+G (ou Commande+G sous macOS) à une ligne de menu en plaçant les caractères "/G" dans son libellé.

Les caractères de contrôle n'apparaissent pas dans les libellés des commandes de menus. Vous devez donc les éviter afin de ne pas obtenir d'effets indésirables. Ces caractères sont les suivants :

Caractère	Description	Utilisation
(parenthèse ouvrante	Désactiver la ligne
<B	inférieur à B	Gras
<I	inférieur à I	Italique
<U	inférieur à U	Souligné
!+caractère	point d'exclamation+caractère	Caractère d'ajout sous forme de coche (macOS); ajouter une coche (Windows)
/+caractère	slash+caractère	Ajouter un caractère comme raccourci

Paramètres

Il est possible d'associer un paramètre personnalisé à chaque ligne de menu. Un paramètre de ligne de menu est une chaîne de caractères dont le contenu est libre. Il peut être défini dans l'éditeur de menu ou à l'aide de la commande SET MENU ITEM PARAMETER .

Les paramètres de lignes de menus sont utiles pour la gestion programmée des menus, en particulier lors de l'utilisation des commandes Dynamic pop up menu , Get menu item parameter et Get selected menu item parameter .

Action

Chaque commande de menu peut avoir une méthode projet ou une action standard qui lui est associée. Lorsqu'une ligne de menu est sélectionnée, 4D exécute l'action standard ou la méthode projet qui lui est associée. Par exemple, la ligne Etat peut appeler une méthode projet qui prépare un état mensuel à partir d'une table contenant des données financières. La ligne Couper peut appeler l'action standard `Couper` pour placer la sélection dans le Presse-papiers et l'effacer de la fenêtre de premier plan.

Si aucune méthode ou action standard n'est affectée à une commande de menu, le choix de cette commande provoquera la fermeture du mode Application et l'ouverture du mode Développement. Si seul le mode Application est disponible, ce qui signifie un retour au Desktop.

Les actions standard peuvent être utilisées pour effectuer des opérations associées aux fonctions système (copier, quitter, etc.) ou aux fonctions de la base (nouvel enregistrement, tout sélectionner, etc.).

Vous pouvez associer à la fois une action standard et une méthode projet à une commande de menu. Dans ce cas, l'action standard n'est jamais exécutée ; toutefois, 4D utilise cette action pour activer/inactiver la commande de menu en fonction du contexte et pour associer une opération spécifique en fonction de la plateforme. Lorsqu'une commande de menu est inactivée, la méthode projet associée ne peut être exécutée.

Vous choisissez d'associer une action standard ou une méthode projet à la commande de menu en fonction du type de résultat attendu. En principe, il est préférable de choisir une action standard lorsque cela est possible car les actions standard mettent en oeuvre des mécanismes optimisés, notamment l'activation/inactivation de la ligne en fonction du contexte.

Associer une méthode projet ou une action standard

Pour associer une méthode projet et/ou une action standard à une commande de menu sélectionnée dans l'éditeur de menu :

- Nom de la méthode : sélectionnez une méthode projet existante dans la combo box. Si la méthode projet n'existe pas, saisissez son nom dans la combo box "Nom de la méthode" puis cliquez sur le bouton [...]. 4D affiche la boîte de dialogue de création de méthode projet, vous permettant d'accéder à l'éditeur de méthodes.
- Action standard associée : Choisissez ou saisissez le nom de l'action que vous souhaitez associer dans la combo box "Action standard associée". Vous pouvez saisir toute action prise en charge et (optionnellement) tout paramètre dans la zone. Pour la liste complète des actions standard, veuillez vous reporter à la section Actions standard dans le *Mode Développement*. Note macOS : Sous macOS, les commandes de menus créés associées à l'action `Quitter` sont automatiquement placées dans le menu de l'application, conformément aux normes d'interface de cette plate-forme.

A l'aide du langage 4D, vous pouvez associer une méthode projet via la commande `SET MENU ITEM METHOD` et une action standard via la commande `SET MENU ITEM PROPERTY`.

Démarrer un process

L'option Démarrer un nouveau process est disponible pour les commandes de menu associées à des méthodes. Elle peut être définie via une case à cocher dans l'éditeur de menus, ou via le paramètre *property* de la commande `SET MENU ITEM PROPERTY`.

Lorsque l'option Démarrer un nouveau process est activée, un nouveau process est créé lorsque la commande de menu est choisie. Normalement, une méthode associée à une commande de menu est exécutée dans le process courant, à moins que vous n'appeliez explicitement un autre process dans votre code. La case à cocher Démarrer un nouveau process facilite le lancement d'un nouveau process. Si vous la sélectionnez, 4D créera un nouveau process lorsque la commande de menu sera sélectionnée.

Dans la liste des process, 4D affecte au nouveau process un nom par défaut "ML_NumerProcess". Les noms des process lancés à partir d'une ligne de menu sont créés en combinant le préfixe "ML_" avec le numéro de process.

Exécuter sans valider

L'option Exécuter sans valider est disponible pour les commandes de menu associées à des actions standard uniquement dans l'éditeur de menus.

Lorsque cette option est cochée, 4D ne provoquera pas la "validation" du champ dans lequel se trouve le curseur avant d'exécuter l'action associée. Cette option est principalement destinée aux commandes du menu Edition. Par défaut, 4D traite et "valide" le contenu d'un champ avant d'exécuter une action standard (via une commande de menu ou un raccourci-clavier), ce qui a pour effet de générer un événement formulaire `Sur données modifiées`. Ce principe peut gêner le fonctionnement des commandes du type copier ou coller, car au moment de leur appel, l'événement `Sur données modifiées` est généré de manière inopinée. Dans ce cas, il est utile de cocher l'option `Exécuter sans valider`.

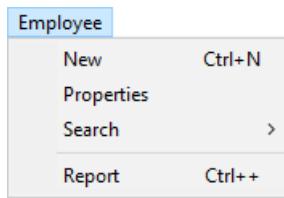
Privilèges d'accès à distance

Cette option de l'éditeur de menus permet de définir un groupe pour une commande de menu afin que seuls les utilisateurs de ce groupe puissent utiliser la commande de menu depuis un 4D distant (voir Utilisateurs et groupes).

Options

Ligne de séparation

Les groupes de commandes de menus peuvent être séparés par un filet. Cette fonctionnalité est utile pour rassembler les commandes de menus par fonction.



Vous ajoutez un filet de séparation en créant une commande de menu.

Dans l'éditeur de menus, au lieu de saisir le nom de la commande de menu, il suffit de cocher l'option `Ligne de séparation`. La ligne apparaît alors dans la zone de la commande courante. Lorsque cette option est cochée, les autres propriétés sont sans effet. Note : Sous macOS, il est possible de procéder en faisant commencer le nom de la commande par un tiret "-". Cette commande sera alors affichée comme une ligne de séparation.

Dans le langage 4D, vous insérez une ligne de séparation en saisissant `-` ou `(-` comme `itemText` pour les commandes `APPEND MENU ITEM`, `INSERT MENU ITEM`, ou `SET MENU ITEM`.

Raccourcis clavier

Vous pouvez affecter des raccourcis clavier à toute commande de menu. Lorsqu'une commande de menu se voit affecter un raccourci clavier, il s'affiche en face de son libellé. Par exemple, "Ctrl+C" (Windows) ou "Commande+C" (macOS) apparaît en face de la commande de menu Copier dans le menu Edition.

Vous pouvez également ajouter les touches Majuscule ainsi que Alt (Windows) ou Option (macOS) au raccourci clavier associé à une commande. Cette possibilité multiplie le nombre de raccourcis clavier utilisables dans les barres de menus. Les raccourcis clavier définis peuvent donc être de différents types :

- Sous Windows :
 - Ctrl+lettre
 - Ctrl+Maj+lettre
 - Ctrl+Alt+lettre
 - Ctrl+Maj+Alt+lettre
- Sous macOS :
 - Commande+lettre
 - Commande+Maj+lettre
 - Commande+Option+lettre
 - Commande+Maj+Option+lettre

Lorsque vous utilisez des actions standard, il est conseillé de conserver les raccourcis clavier qui leur sont

associés par défaut.

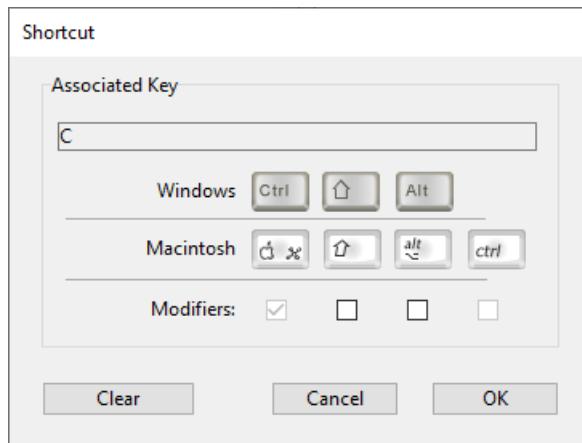
Vous pouvez utiliser toute touche alphanumérique comme raccourci clavier, hormis celles qui sont utilisées par les commandes de menus standard qui apparaissent dans les menus Editer et Fichier, et les clés réservées aux commandes de menu 4D.

Les combinaisons réservées sont décrites dans le tableau suivant :

Clé (Windows)	Clé (macOS)	Opération
Ctrl+C	Commande+C	Copier
Ctrl+Q	Commande+Q	Quitter
Ctrl+V	Commande+V	Coller
Ctrl+X	Commande+X	Couper
Ctrl+Z	Commande+Z	Annuler
Ctrl+. (point)	Commande+. (point)	Arrêter action en cours

Pour affecter un raccourci clavier dans l'éditeur de menus :

Sélectionnez la ligne de menu à laquelle vous souhaitez affecter un raccourci clavier. Cliquez sur le bouton [...] à droite de la zone "Raccourci clavier". La fenêtre suivante apparaît :



Saisissez le caractère à utiliser puis (facultatif) cochez les options Majuscule et/ou Alt (Option) en fonction de la combinaison que vous souhaitez obtenir. Vous pouvez également appuyer sur les touches constituant la combinaison (n'appuyez pas sur la touche Ctrl/Commande).

Il n'est pas possible de désélectionner la touche Ctrl/Commande, elle est obligatoire dans les raccourcis clavier des menus. Pour recommencer, cliquez sur la touche Effacer. Cliquez sur OK pour valider la modification. Le raccourci défini est représenté dans la zone "Raccourci clavier".

Pour affecter un raccourci clavier à l'aide du langage 4D, utilisez la commande `SET ITEM SHORTCUT`.

Un objet actif peut aussi avoir un raccourci clavier. Si la touche Ctrl/Commande est sujette à un conflit, l'objet actif sera prioritaire.

Ligne active

Dans l'éditeur de menus, vous pouvez spécifier si une ligne est activée ou désactivée. Une commande de menu activée peut être choisie par l'utilisateur ; une commande de menu désactivée est grise et ne peut pas être choisie. Pour désactiver une ligne de menu, désélectionnez l'option Ligne active. Dans ce cas, la ligne apparaît grise dans le menu et ne peut pas être sélectionnée.

Par défaut, 4D active automatiquement toute commande de menu ajoutée à un menu personnalisé. Vous pouvez désactiver une commande afin, par exemple, de l'activer uniquement par programmation (commandes `ENABLE MENU`

`ITEM` et `DISABLE MENU ITEM`).

Coche

Cette option de l'éditeur de menus permet d'associer par défaut une coche système à la ligne de menu. Vous pourrez ensuite gérer (masquer ou afficher) la coche au moyen des commandes du langage (`SET MENU ITEM MARK` et `Get menu item mark`).

Les coches sont généralement utilisées pour des menus à action permanente et indiquent que l'action est en cours.

Styles des polices

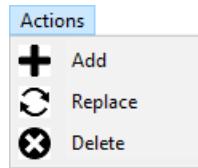
4D vous permet de personnaliser les menus en appliquant différents styles de caractères aux commandes de menus. Vous pouvez personnaliser vos menus avec les styles Gras, Italique ou Souligné, ou à l'aide de la commande `SET MENU ITEM STYLE` .

En règle générale, les styles de police doivent être appliqués à vos menus avec parcimonie, afin d'éviter de conférer une apparence confuse à votre application.

Vous pouvez également appliquer un style en saisissant des caractères spéciaux dans le titre du menu (voir ci-dessus).

Icône ligne

Vous pouvez associer une icône à une ligne de menu. Elle sera affichée directement dans le menu, à côté de la ligne :



Pour définir l'icône dans l'éditeur de menu, choisissez l'option `Ouvrir` pour ouvrir un fichier image à partir du disque. Lorsque vous sélectionnez un fichier image qui n'est pas stocké dans le dossier Resources du projet, il est automatiquement copié dans ce dossier. Une fois définie, l'icône de ligne apparaît dans la zone d'aperçu :



Pour supprimer l'icône de ligne, choisissez l'option `Pas d'icône` dans le menu de la zone "Icône ligne".

Pour définir des icônes de ligne à l'aide du langage 4D, appelez la commande `SET MENU ITEM ICON` .

Barres de menus

Les barres de menu fournissent la principale interface des applications personnalisées. Pour chaque application personnalisée, vous devez créer au moins une barre de menu avec au moins un menu. Par défaut, Menu Bar #1 est la barre de menu qui est affichée dans l'application. Vous pouvez modifier la barre de menu affichée à l'aide de la commande `SET MENU BAR`.

4D vous permet d'associer une image d'accueil personnalisée à chaque barre de menus et de prévisualiser une barre à tout moment.

Image d'accueil

Vous pouvez enrichir l'apparence de chaque barre de menus en lui associant une image d'accueil personnalisée. La fenêtre contenant l'image d'accueil est affichée en-dessous de la barre de menus lorsqu'elle apparaît. Elle peut contenir un logo ou tout type d'image. Par défaut, 4D affiche un logo comme image dans la fenêtre d'accueil :



Une image d'accueil personnalisée peut provenir de toute application graphique. 4D vous permet de coller une image du presse-papiers, d'utiliser une image de la bibliothèque ou toute image présente sur votre disque dur. Tous les formats d'image standard pris en charge par 4D sont utilisables.

L'image d'accueil peut être uniquement paramétrée dans l'éditeur de menus : sélectionnez la barre de menus à laquelle vous souhaitez associer une image d'accueil personnalisée. Notez la zone "Image de fond" à droite de la fenêtre. Pour ouvrir directement une image stockée sur votre disque, cliquez sur le bouton Ouvrir ou cliquez dans la zone "Image de fond". Un pop up menu apparaît :

- Pour coller une image se trouvant dans le Presse-papiers, choisissez la commande Coller.
- Pour ouvrir une image stockée dans un fichier disque, choisissez la commande Ouvrir. Si vous avez choisi la commande Ouvrir, une boîte de dialogue standard d'ouverture de fichiers apparaît, vous permettant de sélectionner le fichier image à utiliser. Une fois définie, l'image s'affiche en taille réduite dans la zone. Elle est alors associée à la barre de menus.



Vous pouvez visualiser le résultat final en testant la barre de menus (cf. paragraphe suivant). En mode Application, l'image est affichée dans la fenêtre d'accueil avec un format du type "tronqué centré".

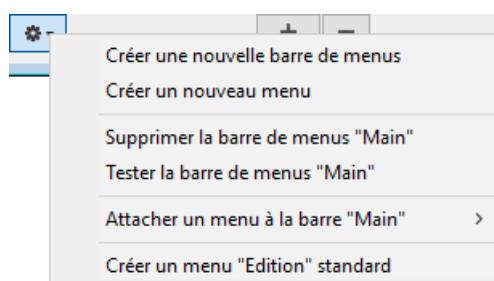
Vous pouvez choisir d'afficher ou de masquer cette fenêtre en cochant l'option **Afficher la barre d'outils dans les Propriétés**.

Pour supprimer l'image personnalisée et afficher l'image par défaut, cliquez sur le bouton **Effacer** ou cliquez dans la zone "Image de fond" et choisissez la commande **Effacer** dans le pop up menu.

Prévisualiser la barre de menus

L'éditeur de menus vous permet de visualiser à tout moment les menus personnalisés et la fenêtre d'accueil sans quitter la fenêtre de la boîte à outils.

Pour cela, il vous suffit de sélectionner la barre de menus et de choisir la commande **Tester la barre de menus "Barre n°N"** dans le menu contextuel ou le menu d'options de l'éditeur.



4D affiche un aperçu de la barre de menus ainsi que de l'écran d'accueil. Vous pouvez dérouler les menus et les sous-menus pour prévisualiser leur contenu. En revanche, les menus ne sont pas actifs. Pour pouvoir tester le fonctionnement des menus et la barre d'outils, vous devez utiliser la commande **Tester l'application** dans le menu Exécution.

Mode SDI sous Windows

Sous Windows, les Développeurs 4D peuvent configurer leurs applications fusionnées pour qu'elles fonctionnent en tant qu'applications SDI (Single-Document Interface). Dans les applications SDI, chaque fenêtre est indépendante des autres et peut avoir sa propre barre de menus. Les applications SDI sont opposées aux applications MDI (Multiple Documents Interface), où toutes les fenêtres sont contenues dans une fenêtre principale, et en dépendent.

Le concept SDI/MDI n'existe pas sur macOS. Cette fonctionnalité concerne uniquement des applications Windows, et les options s'y référant sont ignorées sous macOS.

Disponibilité du mode SDI

Le mode SDI est disponible uniquement dans l'environnement d'exécution suivant :

- Sous Windows
- Application fusionnée, monoposte ou cliente

Activation du mode SDI

L'activation et l'utilisation du mode SDI dans votre application requiert les étapes suivantes :

1. Cochez l'option Utiliser le mode SDI sous Windows dans la page "Interface" de la boîte de dialogue des Propriétés.
2. Générez une application exécutable (monoposte et/ou application cliente).

Par la suite, lorsqu'elle sera exécutée dans le contexte adéquat (voir ci-dessus), l'application fusionnée fonctionnera automatiquement en mode SDI.

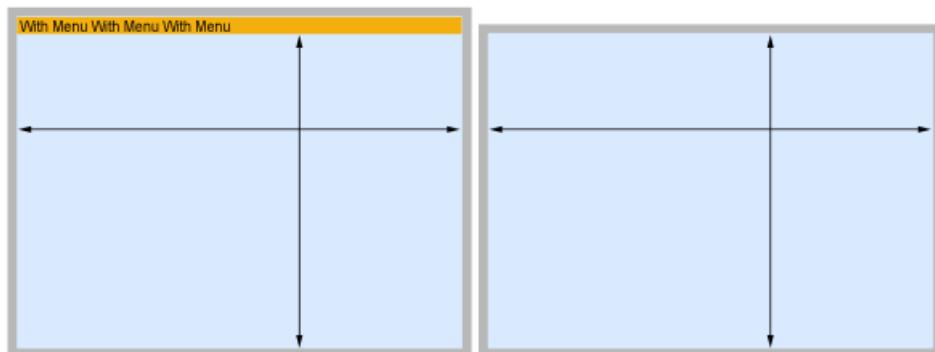
Gestion des applications en mode SDI

L'exécution d'une application 4D en mode SDI ne requiert aucune implémentation particulière : les barres de menu existantes sont automatiquement déplacées dans les fenêtres SDI elles-mêmes. Cependant, vous devez être attentif aux principes spécifiques énumérés ci-dessous :

Menus dans les fenêtres

En mode SDI, la barre de menu du process s'affiche automatiquement dans chaque fenêtre "document" ouverte durant la durée de vie du process (ceci exclut par exemple les palettes flottantes). Lorsque la barre de menu du process n'est pas visible, les raccourcis des lignes de menu restent toutefois actifs.

Les menus sont ajoutés au-dessus des fenêtres sans modifier la taille de leur contenu :



Les fenêtres peuvent donc être utilisées dans les modes MDI ou SDI sans avoir à recalculer la position des objets.

- Si l'option Afficher fenêtres : Accueil de la page "Interface" a été sélectionnée dans les Propriétés, la fenêtre d'accueil contiendra les menus qui auraient été affichés dans la fenêtre MDI. Notez également que la fermeture de la fenêtre d'accueil entraînera la sortie de l'application, tout comme dans le mode MDI.
- Si l'option Accueil n'a pas été cochée dans la base de données, les menus seront affichés uniquement dans les fenêtres ouvertes, selon les choix du Développeur.

Arrêt automatique

Lorsqu'elle est exécutée en mode MDI, une application 4D quitte simplement lorsque l'utilisateur ferme la fenêtre de l'application (fenêtre MDI). Cependant, lorsqu'elles sont exécutées en mode SDI, les applications 4D n'ont pas de fenêtre d'application et, d'autre part, la fermeture de la dernière fenêtre ouverte ne signifie pas nécessairement que l'utilisateur souhaite quitter l'application (des process sans interface peuvent être exécutés par exemple) -- mais cela peut être le cas.

Pour gérer cette situation, les applications 4D exécutées en mode SDI incluent un mécanisme pour quitter automatiquement (en appelant la commande `QUIT 4D`) lorsque les conditions suivantes sont remplies :

- l'utilisateur ne peut plus interagir avec l'application
- il n'y a pas de process utilisateur en cours
- les process 4D ou workers sont en attente d'un événement
- le serveur Web n'est pas lancé.

Lorsqu'un menu avec une action standard associée pour *quitter* est appelé, l'application quitte et toutes les fenêtres sont fermées, quel que soit l'endroit d'où le menu a été appelé.

Langage

Bien qu'il soit traité de manière transparente par 4D, le mode SDI introduit de légères variations dans la gestion de l'interface d'application. Les spécificités dans le langage 4D sont listées ci-dessous :

Commande/fonctionnalité	Spécificité en mode SDI sous Windows
<code>Open form window</code>	Options pour supporter les fenêtres flottantes en SDI (<code>Controller form window</code>) et pour supprimer la barre de menu (<code>Form has no menu bar</code>)
<code>Menu bar height</code>	Retourne la hauteur en pixels d'une ligne de barre de menu unique, même si la barre de menu a été incluse sur une ou plusieurs lignes. Retourne 0 lorsque la commande est appelée à partir d'un process sans fenêtre formulaire
<code>SHOW MENU BAR / HIDE MENU BAR</code>	Appliqué uniquement à la fenêtre formulaire courante (d'où le code est exécuté)
<code>MAXIMIZE WINDOW</code>	La fenêtre est maximisée à la taille de l'écran
<code>CONVERT COORDINATES</code>	<code>XY Screen</code> est le système de coordonnées global dans lequel l'écran principal est positionné à (0,0). Les écrans à gauche ou au-dessus de lui peuvent avoir des valeurs de coordonnées négatives et les écrans à droite ou au-dessous de lui peuvent avoir des valeurs de coordonnées supérieures à celles retournées par <code>Screen height</code> ou <code>Screen width</code> .
<code>GET MOUSE</code>	Les coordonnées globales sont relatives à l'écran
<code>GET WINDOW RECT</code>	Lorsque -1 est passé dans le paramètre fenêtre, la commande retourne 0;0;0;0
<code>On Drop database method</code>	Non supporté

Propriétés utilisateur

4D propose deux modes de fonctionnement pour les Propriétés des projets :

- Mode standard : tous les [paramètres](#) sont stockés dans le fichier [settings.4DSettings au niveau du projet](#) et sont appliqués dans tous les cas. C'est le mode par défaut, adapté à la phase de développement (toutes les applications).
- Mode propriétés utilisateur : une partie des paramètres personnalisés sont stockés dans un fichier [settings.4DSettings dans le dossier Settings](#) (pour tous les fichiers de données) ou [dans le dossier Data](#) (pour ce fichier de données) et sont utilisés à la place des paramètres de structure. Ce mode convient à la phase de déploiement pour les applications Desktop. Activez ce mode à l'aide d'une option située sur la [page Sécurité](#) des Propriétés.

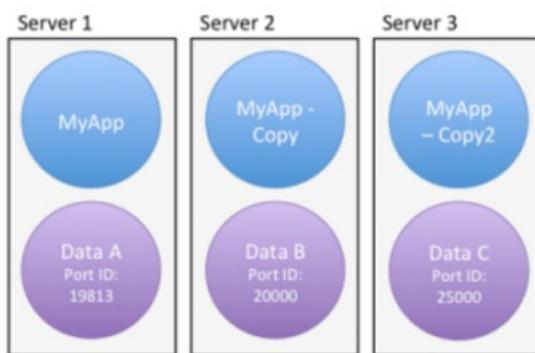
En définissant des propriétés utilisateur, vous pouvez conserver des paramètres personnalisés entre les mises à jour de vos applications 4D, ou gérer des paramètres différents pour une même application 4D déployée sur plusieurs sites différents. Cela permet également d'utiliser la programmation pour gérer les fichiers de paramètres à l'aide de XML.

4D peut générer et utiliser deux types de propriétés utilisateur :

- Les propriétés utilisateur (standard) : elles sont utilisées à la place des propriétés de structure pour tout fichier de données ouvert avec l'application.
- Propriétés utilisateur pour le fichier de données : elles peuvent être définies spécifiquement pour chaque fichier de données utilisé avec votre application, en configurant par exemple l'ID du port ou le cache du serveur.

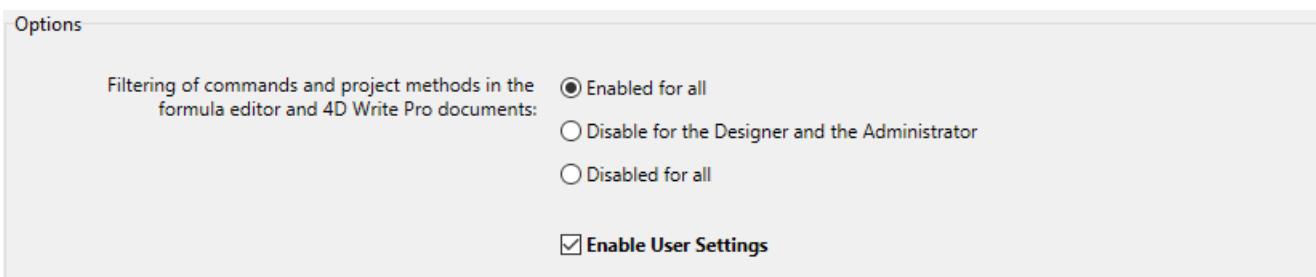
Avec cette option, vous pouvez facilement déployer et mettre à jour plusieurs copies de la même application Desktop avec plusieurs fichiers de données, chacun contenant des propriétés différentes.

Prenons par exemple la configuration suivante, où une application est dupliquée et où chaque copie utilise un paramètre d'ID de port différent. Si ce paramètre utilisateur est lié au fichier de données, vous pourrez mettre à jour l'application sans avoir à modifier manuellement l'ID du port :



Activer les propriétés utilisateur

Pour activer les paramètres utilisateur, vous devez cocher l'option Paramètres > Sécurité > Autoriser les propriétés utilisateur :

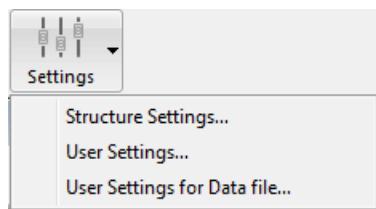


Lorsque vous cochez cette option, les paramètres sont séparés en trois boîtes de dialogue :

- Propriétés structure

- Propriétés utilisateur
- Propriétés utilisateur pour le fichier de données

Vous pouvez accéder à ces boîtes de dialogue en utilisant le menu Développement > Propriétés... ou le bouton Propriétés dans la barre d'outils :

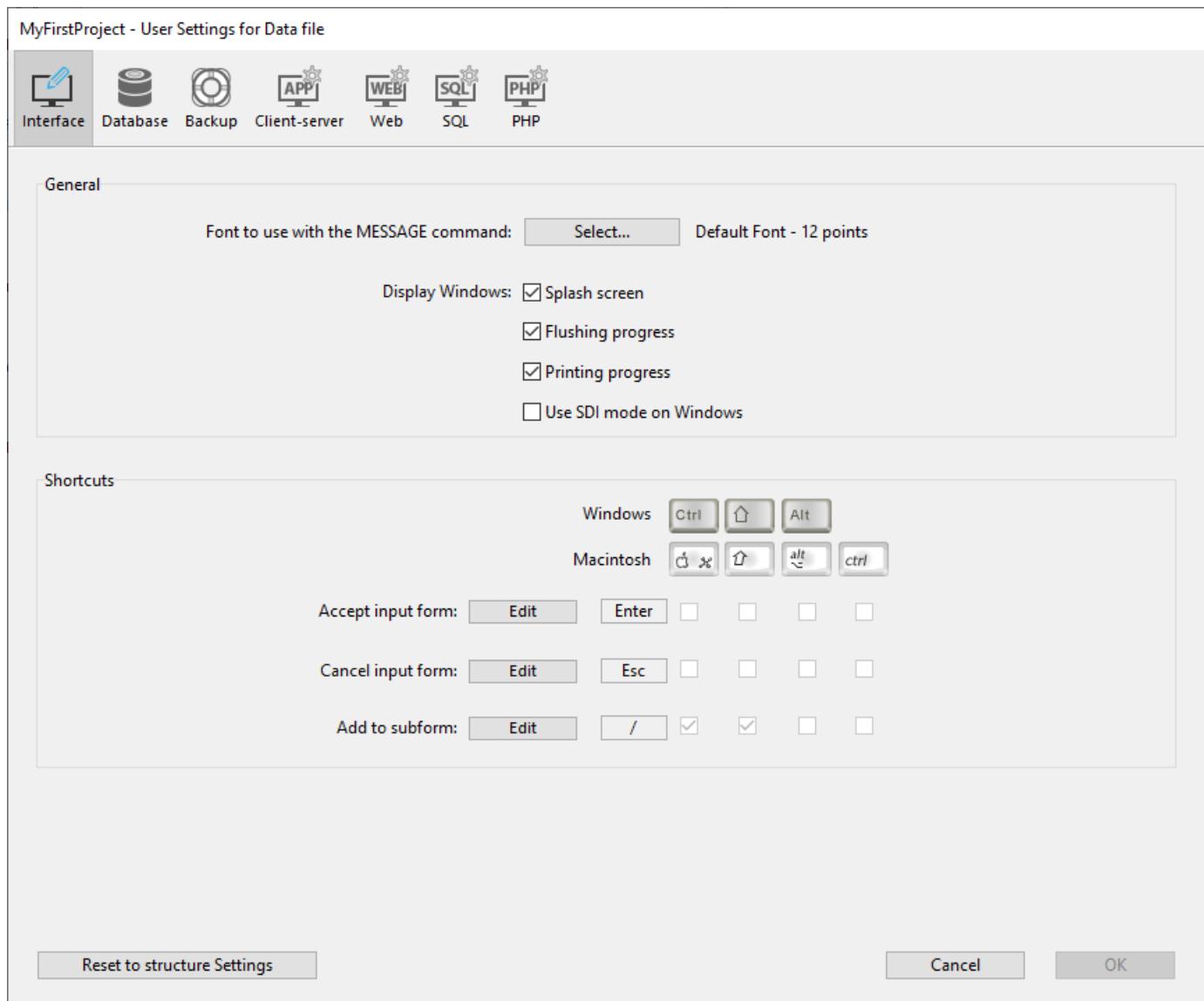


Vous pouvez également accéder à ces boîtes de dialogue à l'aide de la commande [OPEN SETTINGS WINDOW](#) avec le sélecteur *settingsType* approprié.

La boîte de dialogue Propriétés de structure est identique à la boîte de dialogue Propriétés standard et permet d'accéder à toutes ses propriétés (qui peuvent être écrasées par des propriétés utilisateur).

Propriétés utilisateur et propriétés utilisateur pour le fichier de données

Les boîtes de dialogue Propriétés utilisateur et Propriétés utilisateur pour le fichier de données contiennent une sélection de propriétés pertinentes qui peuvent être définies pour tous les fichiers de données ou pour un seul fichier de données :



Le tableau suivant répertorie les pages de paramètres que l'on trouve dans les boîtes de dialogue Paramètres utilisateur et Paramètres utilisateur pour le fichier de données et décrit leurs principales différences par rapport aux paramètres standard :

Page des Propriétés structure	Page des Propriétés utilisateur	Page des Propriétés utilisateur pour fichier de données
Page Général	N/a	N/a
Page Interface	Identique aux Propriétés structure	Identique aux Propriétés structure
Page Compilateur	N/a	N/a
Page Base de données/Stockage des données	N/a	N/a
Page Base de données/Mémoire	Identique aux Propriétés structure	Identique aux Propriétés structure
Page Sauvegarde/Périodicité	N/a	Identique aux Propriétés structure
Page Sauvegarde/Configuration	N/a	Identique aux Propriétés structure
Page Sauvegarde/Sauvegarde et restitution	N/a	Identique aux Propriétés structure
Page Client-Serveur/Options réseau	Identique aux Propriétés structure	Identique aux Propriétés structure
Page Client-Serveur/Configuration IP	Identique aux Propriétés structure	Identique aux Propriétés structure
Page Web/Configuration	Identique aux Propriétés structure	Identique aux Propriétés structure
Page Web/Options (I)	Identique aux Propriétés structure	Identique aux Propriétés structure
Page Web/Options (II)	Identique aux Propriétés structure	Identique aux Propriétés structure
Page Web/Journal (format)	Identique aux Propriétés structure	Identique aux Propriétés structure
Page Web/Journal (périodicité)	Identique aux Propriétés structure	Identique aux Propriétés structure
Page Web/Web services	Option de préfixage des méthodes non disponible	Option de préfixage des méthodes non disponible
Page SQL	Identique aux Propriétés structure	Identique aux Propriétés structure
Page PHP	Identique aux Propriétés structure	Identique aux Propriétés structure
Page sécurité	N/a	N/a
Page de compatibilité	N/a	N/a

Lorsque vous modifiez des paramètres dans cette boîte de dialogue, ils sont automatiquement enregistrés dans le fichier *settings.4DSettings* correspondant (voir ci-dessous).

SET DATABASE PARAMETER et propriétés utilisateur

Certaines propriétés utilisateur sont aussi disponibles via la commande [SET DATABASE PARAMETER](#). Pour les propriétés utilisateur, la propriété Conservé entre deux sessions est fixée à Oui.

Lorsque la fonctionnalité Propriétés utilisateur est activée, les propriétés utilisateur modifiées par la commande [SET DATABASE PARAMETER](#) sont automatiquement stockées dans les Propriétés utilisateur pour le fichier de données.

Numéro automatique table est une exception ; cette valeur de réglage est toujours stockée dans le fichier de données lui-même.

Fichiers settings.4DSettings

Lorsque vous [cochez l'option Autoriser les propriétés utilisateur](#), des fichiers de propriétés utilisateur sont automatiquement créés. Leur emplacement dépend du type de propriété utilisateur définie.

Paramètres utilisateur (standard)

Le fichier de propriété utilisateur standard est automatiquement créé et placé dans un dossier de propriétés à l'emplacement suivant :

`ProjectFolder/Settings/settings.4DSettings`

... où *ProjectFolder* est le nom du dossier contenant le fichier de structure du projet.

Dans les applications fusionnées, le fichier de propriétés utilisateur est placé à l'emplacement suivant :

- Dans les versions mono-utilisateur : `ProjectFolder/Database/Settings/settings.4DSettings`
- Dans les versions client/serveur : `ProjectFolder/Database/Settings/settings.4DSettings`

Propriétés utilisateur pour fichier de données

Le fichier de propriétés utilisateur lié au fichier de données est automatiquement créé et placé dans un dossier de propriétés à l'emplacement suivant :

`Data/Settings/settings.4DSettings`

... où *Data* est le nom du dossier contenant le fichier de données actuel de l'application.

Lorsque le fichier de données est situé au même niveau que le fichier de structure du projet, les fichiers de propriétés utilisateur basés sur la structure et sur les données partagent le même emplacement et le même fichier. La commande de menu Propriétés utilisateur pour le fichier de données... n'est pas proposée.

Les fichiers de propriétés sont des fichiers XML ; ils peuvent être lus et modifiés à l'aide des commandes XML intégrées de 4D ou d'un éditeur XML. Cela signifie que vous pouvez gérer les paramètres par programmation, notamment dans le cadre d'applications compilées et fusionnées avec 4D Volume Desktop. Lorsque vous modifiez ce fichier par programmation, les changements ne sont pris en compte que lors de la prochaine ouverture de la base de données.

Priorité des paramètres

Les propriétés peuvent être stockées à trois niveaux. Chaque paramètre défini à un niveau a priorité sur le même paramètre défini à un niveau précédent, le cas échéant :

Niveau de priorité	Nom	Emplacement	Commentaires
3 (le plus faible)	Paramètres de structure (ou Paramètres lorsque la fonction "Paramètres utilisateur" n'est pas activée)	Fichier <i>settings.4DSettings</i> dans le dossier Sources (bases projet) ou dans le dossier Settings au même niveau que le fichier de structure (bases binaires)	Emplacement unique lorsque les paramètres utilisateur ne sont pas activés. Appliqué à toutes les copies de l'application.
2	Propriétés utilisateur (tous les fichiers de données)	Fichier <i>settings.4DSettings</i> dans le dossier Settings au même niveau que le dossier Project	Remplace les propriétés de structure. Stocké dans le package de l'application.
1 (le plus élevé)	Propriétés utilisateur (fichier de données courant)	Fichier <i>settings.4DSettings</i> dans le dossier Settings au même niveau que le fichier de données	Remplace les propriétés de structure et les propriétés utilisateur. S'applique uniquement lorsque le fichier de données associé est utilisé avec l'application.

A noter que les fichiers de propriétés utilisateur ne contiennent qu'un sous-ensemble de paramètres pertinents, tandis que le fichier de structure contient tous les paramètres personnalisés, y compris les paramètres de bas niveau.

Générateur d'application

4D inclut un générateur d'application pour créer un package de projet (version finale). Ce générateur simplifie le processus de finalisation et de déploiement des applications compilées 4D. Il gère automatiquement les fonctionnalités spécifiques de différents systèmes d'exploitation et facilite le déploiement d'applications client-serveur.

Le générateur d'applications vous permet de :

- Générer une structure compilée, sans code interprété,
- Générer une application autonome exécutable, c'est-à-dire fusionnée avec 4D Volume Desktop, le moteur de base de données 4D,
- Générer différentes applications à partir de la même structure compilée via un projet XML,
- Générer des applications client-serveur homogènes,
- Générer des applications client-serveur avec mise à jour automatique des composants client et serveur.
- Enregistrer vos paramètres de génération pour une utilisation ultérieure (bouton *Enregistrer les paramètres*).

Les applications compilées sont basées sur des [fichiers .4dz files](#) qui sont en lecture seule. A noter que l'utilisation de commandes ou de fonctions qui modifient les fichiers sources (telles que `CREATE INDEX` ou `CREATE TABLE` (SQL)) n'est pas possible par défaut dans les applications compilées. Vous pouvez néanmoins créer des applications spécifiques qui prennent en charge les modifications locales en utilisant la clé XML du [PackProject](#) (voir [doc.4d.com](#)).

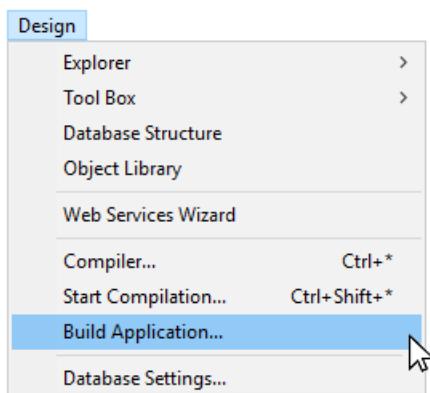
Aperçu

Générer un package de projet peut être réalisée à l'aide de :

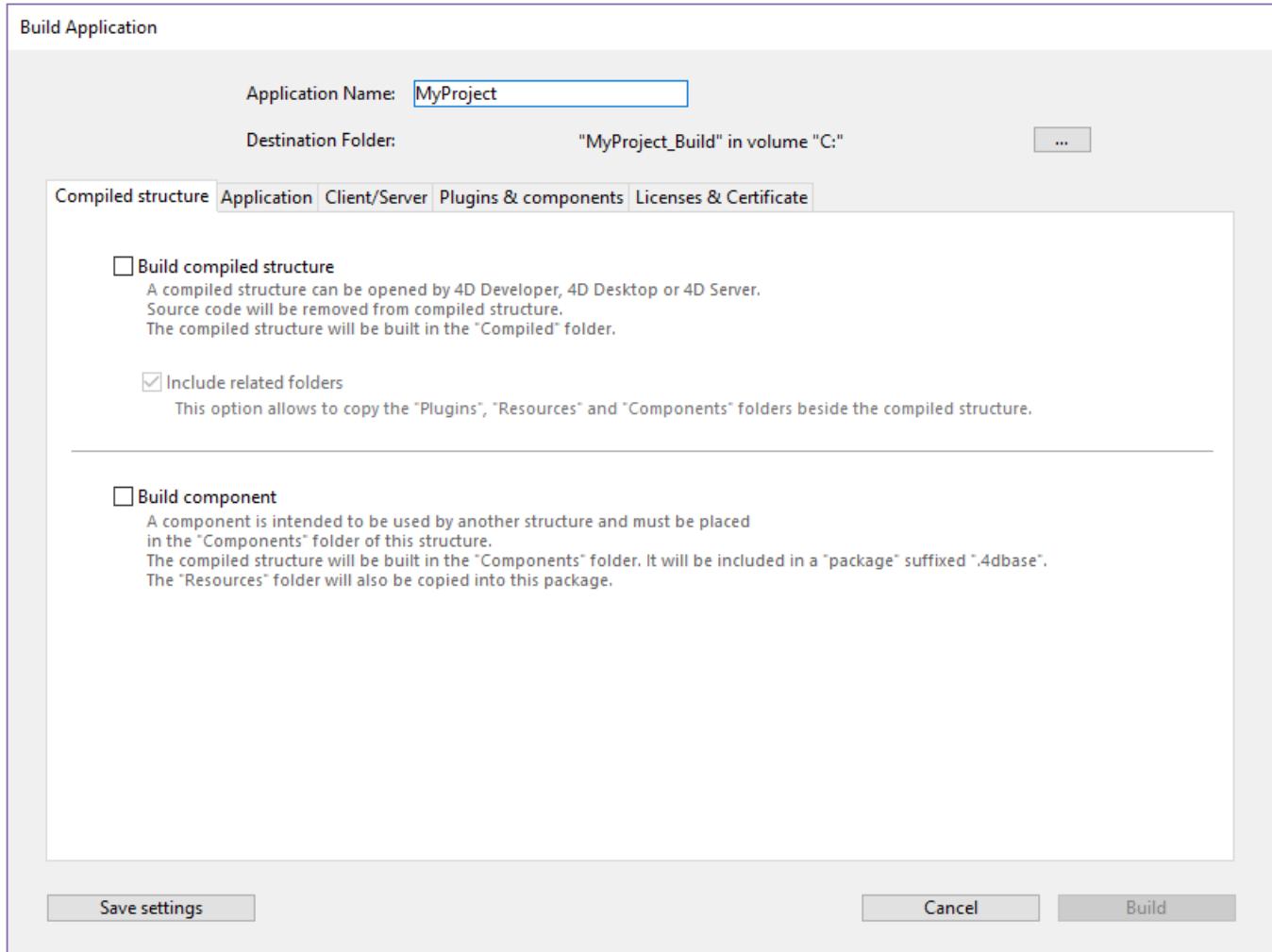
- soit la commande [BUILD APPLICATION](#),
- Utiliser la [Page Application](#) ou la [Page Client/Serveur](#) de boîte de dialogue du Générateur d'applications.

Construire application cliente

Cochez l'option Créez une application autonome et cliquez sur Générer pour créer une application autonome (double-cliquable) directement à partir de votre projet d'application.



La boîte de dialogue du générateur d'application comprend plusieurs pages accessibles via des onglets :



La génération ne peut s'effectuer qu'une fois le projet compilé. Si vous sélectionnez cette commande sans avoir préalablement compilé le projet ou si le code compilé ne correspond pas au code interprété, une boîte de dialogue d'avertissement apparaît indiquant que le projet doit être (re)compilé.

buildApp.4DSettings

Chaque paramètre de génération de l'application est sauvegardé en tant que clé XML dans le fichier XML de l'application nommé `buildApp.4DSettings`, situé dans le dossier [Settings du projet](#).

Les paramètres par défaut sont utilisés lors de la première utilisation de la boîte de dialogue du Générateur d'application. Le contenu du fichier est mis à jour, si nécessaire, lorsque vous cliquez sur Construire ou Enregistrer les paramètres. Vous pouvez définir plusieurs autres fichiers de paramètres XML pour le même projet et les utiliser à l'aide de la commande [BUILD APPLICATION](#).

Les clés XML fournissent des options supplémentaires à celles affichées dans la boîte de dialogue du Générateur d'application. La description de ces clés est détaillée dans le manuel [4D Clés XML BuildApplication](#).

Fichier d'historique

Lorsqu'une application est créée, 4D génère un fichier journal nommé `BuildApp.log.xml` dans le dossier Logs du projet. Le fichier d'historique stocke les informations suivantes pour chaque génération :

- Le début et la fin de la génération des cibles,
- Le nom et le chemin d'accès complet des fichiers générés,
- La date et l'heure de la génération,
- Toutes les erreurs qui se sont produites,
- Tout problème de signature (par exemple, un plug-in non signé).

La vérification de ce fichier peut vous aider à gagner du temps lors des prochaines étapes de déploiement, si vous avez l'intention, par exemple, de notariser votre application.

Utilisez la commande `Get 4D file (Build application log file)` pour obtenir l'emplacement du fichier journal.

Nom de l'application et dossier de destination

Application Name:

Destination Folder:

Entrez le nom de l'application dans Nom de l'application.

Spécifiez le dossier de l'application générée dans le Dossier de destination. Si le dossier spécifié n'existe pas déjà, 4D vous créera un dossier *Build*.

Page de structure compilée

Cet onglet vous permet de générer un fichier de structure compilé standard et/ou un composant compilé :

Build Application

Application Name:

Destination Folder:

Compiled structure Application Client/Server Plugins & components Licenses & Certificate

Build compiled structure
A compiled structure can be opened by 4D Developer, 4D Desktop or 4D Server.
Source code will be removed from compiled structure.
The compiled structure will be built in the "Compiled" folder.

Include related folders
This option allows to copy the "Plugins", "Resources" and "Components" folders beside the compiled structure.

Build component
A component is intended to be used by another structure and must be placed in the "Components" folder of this structure.
The compiled structure will be built in the "Components" folder. It will be included in a "package" suffixed ".4dbase".
The "Resources" folder will also be copied into this package.

Générer une structure compilée

Génère une application contenant uniquement du code compilé.

Cette fonctionnalité crée un fichier `.4dz` dans un dossier `Compiled Database/<project name>`. Par exemple, si vous avez nommé votre application «MyProject», 4D créera :

`<destination>/Compiled Database/MyProject/MyProject.4dz`

Un fichier `.4dz` est essentiellement une version compressée du dossier du projet. Les fichiers `.4dz` peuvent être utilisés par 4D Server, 4D Volume Desktop (applications fusionnées) et 4D. La taille compacte et optimisée des fichiers `.4dz` facilite le déploiement des packages de projet.

Lors de la génération de fichiers .4dz, 4D utilise par défaut un format zip standard. L'avantage de ce format est qu'il est facilement lisible par tout outil de dézippage. Si vous ne souhaitez pas utiliser ce format standard, ajoutez la clé XML `UseStandardZipFormat` avec la valeur `False` dans votre fichier `buildApp.4DSettings` (pour plus d'informations, voir le manuel [4D XML Keys Backup](#)).

Inclure les dossiers associés

Lorsque vous cochez cette option, tous les dossiers liés au projet sont recopier dans le dossier Build en tant que dossiers *Components* et *Resources*. Pour plus d'informations sur ces dossiers, veuillez vous reporter à la [description de l'architecture du projet](#).

Générer un composant

Génère un composant compilé à partir de la structure.

Un composant est un fichier de structure 4D standard dans lequel des fonctionnalités spécifiques ont été développées. Une fois le composant configuré et installé dans un autre projet 4D (le projet d'application hôte), ses fonctionnalités sont accessibles depuis le projet hôte.

Si vous avez nommé votre application *Moncomposant*, 4D créera un dossier *Component* contenant le dossier *MyComponent.4dbase* :

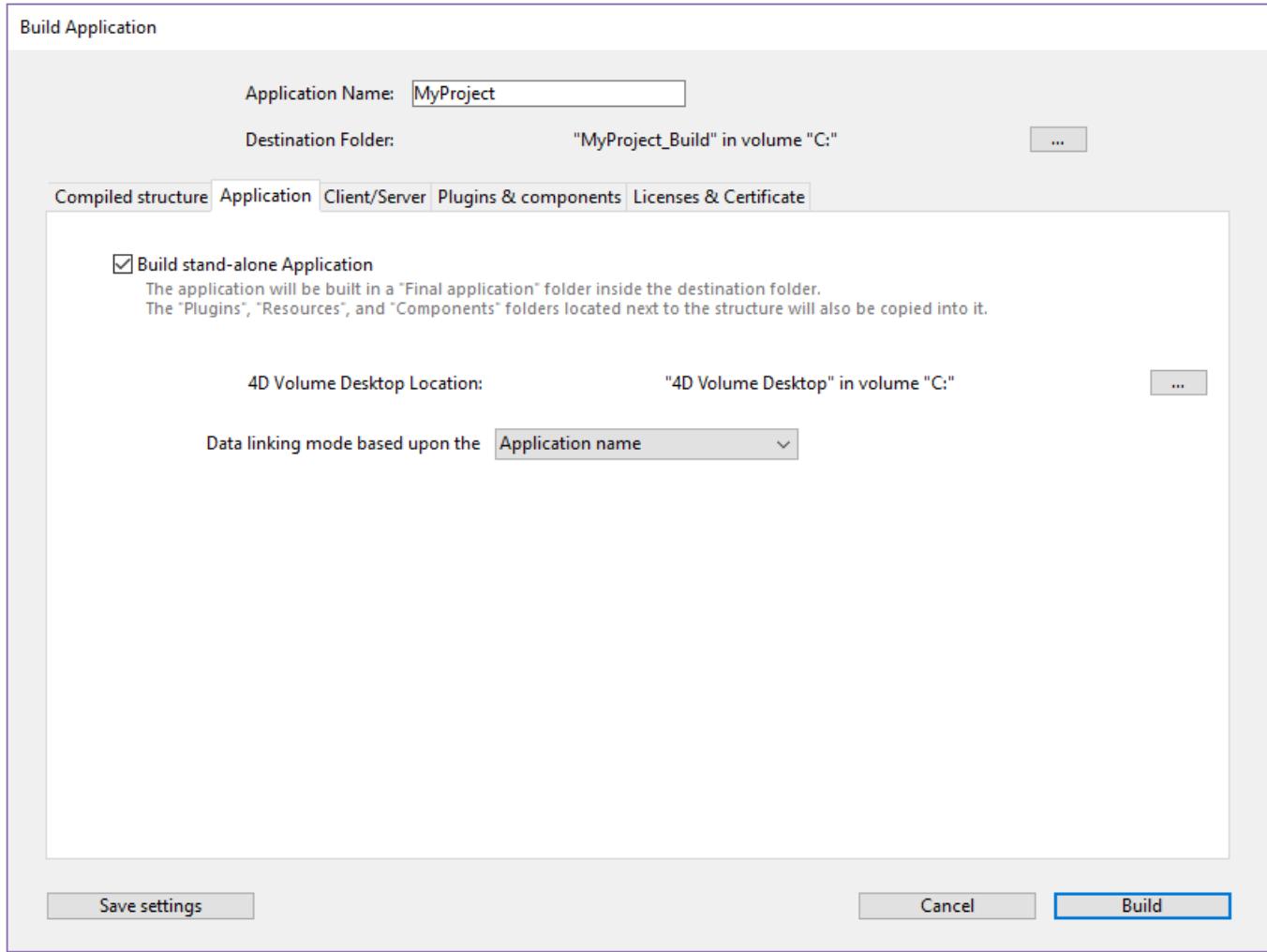
`<destination>/Components/MyComponent.4dbase/MyComponent.4DZ.`

Le dossier *MyComponent.4dbase* contient :

- fichier *MyComponent.4DZ*
- Un dossier *Resources* - toutes les ressources associées sont automatiquement copiées dans ce dossier. Les autres composants et/ou dossiers de plugins ne sont pas copiés (un composant ne peut pas utiliser de plug-ins ou d'autres composants).

Page Application

Cet onglet vous permet de créer une version autonome et monoposte de votre application :



Créer une application autonome

Cochez l'option Crée une application autonome et cliquez sur Générer pour créer une application autonome (double-cliqueable) directement à partir de votre projet d'application.

Les éléments suivants sont requis pour la création :

- 4D Volume Desktop (le moteur de base de données 4D),
- une licence appropriée

Sous Windows, cette fonctionnalité crée un fichier exécutable (.exe). Sous macOS, il gère la création de progiciels.

Le principe consiste à fusionner le fichier 4D Volume Desktop avec votre fichier de structure compilé. Les fonctionnalités offertes par le fichier 4D Volume Desktop sont liées à l'offre commerciale à laquelle vous avez souscrit. Pour plus d'informations sur ce point, reportez-vous à la documentation commerciale et au site Internet de [4D Sas](http://www.4d.com/) (<http://www.4d.com/>).

Vous pouvez définir un fichier de données par défaut ou permettre à l'utilisateur de créer et d'utiliser son propre fichier de données (cf. section [Gestion du fichier de données dans les applications finales](#)).

Il est possible d'automatiser la mise à jour des applications monopostes fusionnées moyennant l'utilisation d'une séquence de commandes du langage (cf. section [Mise à jour auto des applications serveur ou monopostes](#)).

Emplacement du 4D Volume Desktop

Afin de créer une application autonome, il convient d'abord de désigner le dossier contenant le fichier 4D Volume Desktop :

- *sous Windows*, le dossier contient notamment les fichiers 4D Volume Desktop.4DE, 4D Volume Desktop.RSR ainsi que différents fichiers et dossiers nécessaires à son fonctionnement. Ces éléments doivent être placés au premier niveau du dossier sélectionné.
- *sous macOS*, 4D Volume Desktop est fourni sous la forme d'un progiciel structuré contenant divers fichiers et

dossiers génériques.

Pour sélectionner le dossier de 4D Volume Desktop, cliquez sur le bouton [...]. Une boîte de dialogue vous permettant de désigner le dossier (Windows) ou le progiciel (macOS) de 4D Volume Desktop apparaît.

Une fois le dossier sélectionné, son chemin d'accès complet est affiché et, s'il contient effectivement 4D Volume Desktop, l'option de génération d'application exécutable est activée.

Le numéro de version de 4D Volume Desktop doit correspondre à celui du 4D Developer Edition. Par exemple, si vous utilisez 4D Developer v18, vous devez sélectionner un 4D Volume Desktop v18.

Mode de liaison des données

Cette option vous permet de sélectionner le mode de liaison entre l'application fusionnée et le fichier de données local. Deux modes de liaison sont disponibles :

- Nom de l'application (défaut) - Dans ce mode, l'application 4D ouvre automatiquement le dernier fichier de données ouvert correspondant à la structure. Cela vous permet de déplacer librement le dossier de l'application sur le disque. Il est conseillé en général pour les applications fusionnées, à moins que vous n'ayez spécifiquement besoin de dupliquer l'application.
- Chemin de l'application - Dans ce mode, l'application 4D fusionnée va lire le contenu du fichier *lastDataPath.xml* et tenter d'ouvrir le fichier de données dont l'attribut "executablePath" correspond au chemin d'accès de l'application. Si cette clé est trouvée, son fichier de données correspondant (défini via son attribut "dataFilePath") est ouvert. Sinon, le dernier fichier de données utilisé est ouvert (mode par défaut).

Pour plus d'informations sur le mode de liaison des données, reportez-vous au paragraphe [Dernier fichier de données ouvert](#).

Fichiers générés

Lorsque vous cliquez sur le bouton Générer, 4D crée automatiquement un dossier Final Application dans le Dossier de destination défini. Dans le dossier Final Application, se trouve un sous-dossier contenant le nom de l'application spécifiée.

Si vous avez nommé votre application "MyProject", vous trouverez les fichiers suivants dans ce sous-dossier (MyProject):

- *Sous Windows*
 - MonAppli.exe qui est votre exécutable et MonAppli.Rsr qui contient les ressources de l'application
 - Les dossiers 4D Extensions et Resources ainsi que les diverses librairies (DLL), le dossier Native Components et SAS Plugins -fichiers nécessaires au fonctionnement de l'application
 - Un dossier Database contenant notamment un dossier Resources et un fichier MyProject.4DZ. Ils constituent la structure compilée du projet et son dossier Resources. Note : Ce dossier contient également le dossier *Default Data*, s'il a été défini (cf. [Gestion du fichier de données dans les applications finales](#)).
 - (Facultatif) Un dossier Components et/ou un dossier Plugins contenant les fichiers des composants et/ou des plug-ins éventuellement inclus dans le projet. Pour plus d'informations sur ce point, reportez-vous à la section [Plugins et composants](#).
 - Un dossier Licences contenant, sous forme de fichier XML, la liste des numéros de licence ayant été intégrés dans l'application. Pour plus d'informations sur ce point, reportez-vous à la section [Licences & Certificat](#).
 - Les éléments supplémentaires éventuellement ajoutés dans le dossier 4D Volume Desktop (cf. paragraphe [Personnaliser le dossier 4D Volume Desktop](#)).

Tous ces éléments doivent être conservés dans le même dossier afin que l'exécutable fonctionne.

- *macOS*
 - Un progiciel (package) nommé MyProject.app contenant votre application et tous les éléments nécessaires à son fonctionnement, y compris les plug-ins, composants et licences. Pour plus d'informations sur l'intégration des composants et des plug-ins, reportez-vous à la section [Page Plugins et composants](#). Pour plus d'informations sur l'intégration des licences, reportez-vous à la [Page Licences & Certificat](#). **Note **: Sous Mac Os, la commande [Fichier application](#) du langage 4D retourne le chemin d'accès du fichier NomApplication (situé dans le dossier Contents:macOS du progiciel) et non celui du fichier .comp (dossier Contents:Resources du progiciel).

Personnaliser le dossier 4D Volume Desktop

Lors de la construction de l'application exécutable, 4D duplique le contenu du dossier 4D Volume Desktop dans le dossier *Final Application*. Vous pouvez donc parfaitement personnaliser le contenu du dossier 4D Volume Desktop d'origine en fonction de vos besoins. Vous pouvez, par exemple :

- Installer une version de 4D Volume Desktop correspondant à une langue spécifique ;
- Ajouter un dossier *PlugIns* personnalisé ;
- Personnaliser le contenu du dossier *Resources*.

Sous Mac Os, 4D Volume Desktop est fourni sous forme de progiciel. Vous devrez tout d'abord afficher son contenu (effectuez Control+clic sur son icône) afin de pouvoir le modifier.

Emplacements des fichiers Web

Si votre application exécutable est utilisée en tant que serveur Web, les fichiers et dossiers requis par le serveur doivent être installés à des emplacements spécifiques. Ces éléments sont les suivants :

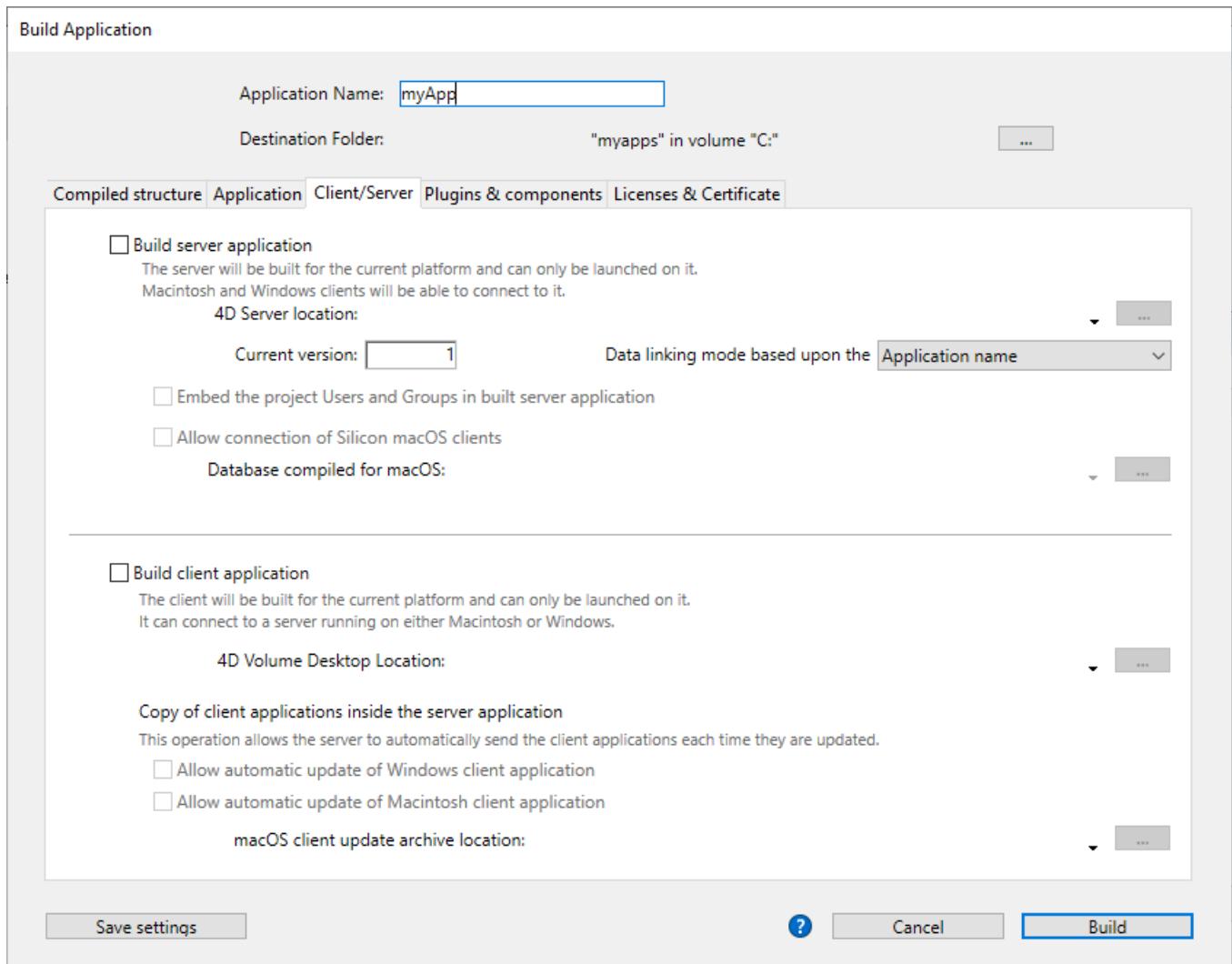
- fichiers *cert.pem* et *key.pem* (facultatifs) : ces fichiers sont utilisés pour les connexions TLS ainsi que par les commandes de cryptage des données,
- dossier racine Web par défaut.

Des éléments doivent être installée :

- Sous Windows : dans le sous-dossier *Final Application\MonAppli\Database*.
- Sous macOS : à côté du progiciel *MonProjet.app*.

Page Client/Serveur

4D vous permet de générer des applications client/serveur personnalisées, homogènes, multi-plateformes et avec option de mise à jour automatique.



Qu'est-ce qu'une application Client/Serveur ?

Une application client/serveur est issue de la combinaison de trois éléments :

- Un projet 4D compilé,
- L'application 4D Server,
- L'application 4D Volume Desktop (macOS et/ou Windows).

Une fois générée, une application client/serveur se compose de deux parties homogènes : la partie Serveur (unique), et la partie Cliente (à installer sur chaque poste client).

Si vous voulez déployer une application client/server dans un environnement hétérogène (applications clientes exécutées sur des machines Intel/AMD et Apple Silicon), il est recommandé de [compiler le projet pour tous les processeurs](#) sur une machine macOS, afin que toutes les applications clientes s'exécutent en code natif.

En outre, l'application client/serveur est personnalisée et son maniement est simplifié :

- Pour lancer la partie serveur, l'utilisateur double-clique simplement sur l'application serveur : il n'est pas nécessaire de sélectionner le fichier projet.
- Pour lancer la partie cliente, l'utilisateur double-clique simplement sur l'application cliente, qui se connecte directement à l'application serveur : il n'est pas nécessaire de choisir un serveur dans une boîte de dialogue de connexion. Le client cible le serveur soit via son nom, lorsque client et serveur sont sur le même sous-réseau, soit via son adresse IP, à définir via la clé XML `IPAddress` dans le fichier buildapp.4DSettings. Si la connexion échoue, [des mécanismes alternatifs spécifiques peuvent être mis en place](#). Il est également possible de "forcer" l'affichage de la boîte de dialogue de connexion standard en maintenant la touche Option (macOS) ou Alt (Windows) enfoncee lors du lancement de l'application cliente. Seule la partie cliente peut se connecter à la partie serveur correspondante. Si un utilisateur tente de se connecter à la partie serveur à l'aide d'une application 4D standard, un message d'erreur est retourné et la connexion est impossible.
- Une application client/serveur peut être paramétrée de telle sorte que la partie cliente [puisse être mise à jour](#)

[automatiquement via le réseau](#). Il vous suffit de créer et de distribuer une version initiale de l'application cliente, les mises à jour ultérieures sont gérées à l'aide du mécanisme de mise à jour automatique.

- Il est également possible d'automatiser la mise à jour de la partie serveur moyennant une séquence de commandes du langage ([SET UPDATE FOLDER](#) et [RESTART 4D](#)).

Construire application serveur

Cochez cette option pour générer la partie serveur de votre application pendant la phase de construction. Vous devez désigner sur votre disque l'emplacement de l'application 4D Server à utiliser. Ce 4D Server doit correspondre à la plate-forme courante (qui sera également la plate-forme de l'application du serveur).

Emplacement de 4D Server

Cliquez sur le bouton [...] et utilisez la boîte de dialogue *Rechercher un dossier* pour localiser l'application 4D Server. Sous macOS, vous devez sélectionner directement le package 4D Server.

Version courante

Utilisée pour indiquer le numéro de version courante de l'application générée. Vous pourrez par la suite accepter ou refuser les connexions des applications clientes en fonction de leur numéro de version. L'intervalle de compatibilité pour les applications client/serveur est défini à l'aide de [clés XML](#) spécifiques).

Intégrer le projet Groupe et Utilisateurs dans une application serveur

Note préliminaire : Les termes suivants sont utilisés dans cette section :

Nom	Définition
Fichier de répertoire du projet	fichier <code>directory.json</code> situé dans le dossier Settings du projet
Fichier de répertoire de l'application	fichier <code>directory.json</code> situé dans le dossier Settings du serveur 4D
Fichier de répertoire des données	fichier <code>directory.json</code> situé dans le dossier Data > Settings

Lorsque vous cochez cette option, le fichier du répertoire du projet est copié dans le fichier du répertoire de l'application au moment de la génération.

Lorsque vous exécutez une application 4D Server générée :

- Si le serveur possède un fichier de répertoire de données, il est chargé.
- Si le serveur ne possède pas de fichier de répertoire de données, le fichier de répertoire de l'application est chargé.

Le fichier du répertoire d'application est en lecture seule. Les modifications apportées aux utilisateurs, aux groupes et aux autorisations pendant l'exécution du serveur sont sauvegardées dans le fichier du répertoire des données. Si aucun fichier de répertoire de données n'existe déjà, il est automatiquement créé. Si le fichier de répertoire de l'application a été incorporé, il est dupliqué en tant que fichier de répertoire des données.

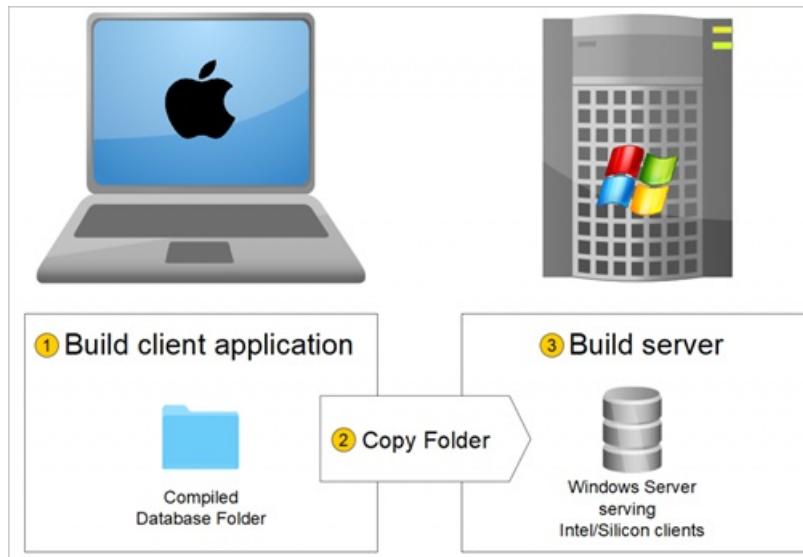
L'incorporation du fichier de répertoire du projet vous permet de déployer une application client/serveur avec une configuration de base de la sécurité des utilisateurs et des groupes. Les modifications ultérieures sont ajoutées au fichier de répertoire des données.

Chemin d'accès à la structure compilée de l'application cliente Apple Silicon/Intel utilisée pour créer un serveur Windows (voir [Autoriser la connexion des clients Silicon Mac](#)).

Lorsque vous créez un serveur sous Windows, cochez cette option pour permettre aux clients Apple Silicon de se connecter à votre application serveur. Vous pouvez alors spécifier un chemin d'accès à la structure compilée pour Apple Silicon/Intel.

Pour permettre aux clients Apple Silicon de se connecter à une application serveur créée sous Windows, vous devez d'abord créer une application cliente sous macOS, avec un projet compilé pour Apple Silicon et Intel. Cela crée automatiquement une structure compilée, identique à celle créée avec l'option [Build compiled structure](#) (sans les dossiers associés).

Vous pouvez ensuite copier cette structure sur votre machine Windows, et l'utiliser pour construire l'application serveur :



Autoriser la connexion des clients Silicon Mac

Si vous voulez déployer une application client/server dans un environnement hétérogène (applications clientes exécutées sur des machines Intel/AMD et Apple Silicon), il est recommandé de [compiler le projet pour tous les processeurs](#) sur une machine macOS, afin que toutes les applications clientes s'exécutent en code natif.

Mode de liaison des données

Cette option vous permet de sélectionner le mode de liaison entre l'application fusionnée et le fichier de données local. Deux modes de liaison sont disponibles :

- Nom de l'application (défaut) - Dans ce mode, l'application 4D ouvre automatiquement le dernier fichier de données ouvert correspondant à la structure. Cela vous permet de déplacer librement le dossier de l'application sur le disque. Il est conseillé en général pour les applications fusionnées, à moins que vous n'ayez spécifiquement besoin de dupliquer l'application.
- Chemin de l'application - Dans ce mode, l'application 4D fusionnée va lire le contenu du fichier *lastDataPath.xml* et tenter d'ouvrir le fichier de données dont l'attribut "executablePath" correspond au chemin d'accès de l'application. Si cette clé est trouvée, son fichier de données correspondant (défini via son attribut "dataFilePath") est ouvert. Sinon, le dernier fichier de données utilisé est ouvert (mode par défaut).

Pour plus d'informations sur le mode de liaison des données, reportez-vous au paragraphe [Dernier fichier de données ouvert](#).

Construire application cliente

Cochez cette option pour générer la partie cliente de votre application lors de la phase de construction.

Vous pouvez cocher cette option :

- avec l'option [Build server application](#) pour créer des parties serveur et client correspondantes pour la plate-forme courante et (éventuellement) inclure les fichiers d'archive de mise à jour automatique,
- sans sélectionner l'option [Build server application](#), généralement pour créer le fichier d'archive de mise à jour à sélectionner à partir de la plate-forme "concurrente" lors de la génération de la partie serveur.

Emplacement du 4D Volume Desktop

Désigne l'emplacement sur votre disque de l'application 4D Volume Desktop à utiliser pour construire la partie cliente de votre application.

Le numéro de version de 4D Volume Desktop doit correspondre à celui du 4D Developer Edition. Le numéro de version de 4D Volume Desktop doit correspondre à celui du 4D Developer Edition.

Ce 4D Volume Desktop doit correspondre à la plate-forme courante (qui sera également la plate-forme de l'application

cliente). Si vous souhaitez générer une version de l'application cliente pour la plate-forme "concurrente", vous devez répéter l'opération en utilisant une application 4D tournant sur cette plate-forme.

Si vous souhaitez que l'application cliente se connecte au serveur via une adresse spécifique (autre que le nom du serveur publié sur le sous-réseau), vous devez utiliser la clé XML `IPAddress` dans le fichier `buildapp.4DSettings`. Si vous souhaitez que l'application cliente se connecte au serveur via une adresse spécifique (autre que le nom du serveur publié sur le sous-réseau), vous devez utiliser la clé XML `IPAddress` dans le fichier `buildapp.4DSettings`. Vous pouvez également mettre en place des mécanismes spécifiques en cas d'échec de la connexion. Les différents scénarios proposés sont décrits dans la section [Gestion de la connexion des applications clientes](#).

Copie des applications clientes dans l'application serveur

Les options de cette zone permettent de mettre en place le mécanisme de mise à jour des parties clientes de vos applications client/serveur via le réseau à chaque nouvelle version de l'application générée. Ces options sont actives uniquement lorsque l'option Construire application cliente est cochée.

- Permettre la mise à jour automatique de l'application cliente Windows - Cochez cette option pour construire un fichier `.4darchive` qui sera envoyé à vos applications clientes sur la plate-forme Windows en cas de mise à jour.
- Permettre la mise à jour automatique de l'application cliente Macintosh - Cochez cette option pour construire un fichier `.4darchive` qui sera envoyé à vos applications clientes sur la plate-forme Macintosh en cas de mise à jour.

Le fichier `.4darchive` est copié à l'emplacement suivant :

```
<ApplicationName>_Build/Client Server executable/Upgrade4DClient/
```

Sélectionner l'archive cliente pour la plate-forme concurrente

Vous pouvez cocher l'option Permettre la mise à jour automatique... pour les applications clientes exécutées sur la plate-forme concurrente. Cette option est disponible seulement si :

- l'option Construire application serveur est cochée,
- l'option Permettre la mise à jour automatique... pour les applications clientes exécutées sur la plate-forme actuelle est cochée.

Pour cela, vous devez cliquer sur le bouton [...] et désigner l'emplacement sur votre disque du fichier à utiliser pour la mise à jour. Le fichier à sélectionner dépend de la plate-forme courante du serveur :

Plateforme du serveur courant	Fichier requis	Détails
macOS	Windows 4D Volume Desktop ou Windows client update archive	Par défaut, sélectionnez l'application <code>4D Volume Desktop</code> pour Windows. Pour sélectionner un fichier <code>.4darchive</code> précédemment créé sur Windows, appuyez sur la touche Shift tout en cliquant sur [...]
Sous Windows	macOS client update archive	Sélectionnez un fichier <code>.4darchive</code> signé, précédemment créé sur macOS

Vous pouvez construire un fichier `.4darchive` spécifique pour la plate-forme concurrente en sélectionnant uniquement [Construire application cliente](#) et l'option [Permettre la mise à jour automatique...](#) adéquate.

Comment proposer une mise à jour ?

Dans la pratique, la proposition de mise à jour des applications clientes découle automatiquement de la mise à jour de l'application serveur.

Le principe est le suivant : lors de la génération d'une nouvelle version de l'application client-serveur depuis le générateur d'applications, la nouvelle partie cliente est copiée sous forme compressée dans le sous-dossier `Upgrade4DClient` du dossier `NomApplication Server` (sous macOS, ces dossiers sont inclus dans le progiciel serveur). Si vous avez suivi le processus de génération d'une application cliente multi-plate-forme, un fichier `.4darchive` de mise à jour est disponible pour chaque plate-forme :

Pour provoquer la mise à jour des applications clientes, il suffit de remplacer l'ancienne version de l'application serveur

par la nouvelle puis de l'exécuter. Le reste du processus est automatique.

Côté client, au moment où l'“ancienne” application cliente tente de se connecter à l’application serveur mise à jour, une boîte de dialogue s'affiche sur le poste client, lui indiquant qu'une nouvelle version est disponible. L'utilisateur peut mettre sa version à jour ou annuler la boîte de dialogue.

- Si l'utilisateur clique sur OK, la nouvelle version est téléchargée sur le poste client via le réseau. A l'issue du téléchargement, l'ancienne application client quitte, la nouvelle est lancée et se connecte au serveur. A l'issue du téléchargement, l'ancienne application client quitte, la nouvelle est lancée et se connecte au serveur.
- Si l'utilisateur clique sur Annuler, la mise à jour est annulée ; si l'ancienne version de l'application cliente n'appartient pas à l'intervalle des numéros de version acceptés par le serveur (cf. paragraphe suivant), l'application quitte et la connexion est impossible. Sinon (par défaut), la connexion est établie.

Comment forcer la mise à jour ?

Dans certains cas, vous pourrez souhaiter que les applications clientes ne puissent pas annuler le téléchargement des mises à jour. Par exemple, si vous avez utilisé une nouvelle version de l'application source 4D Server, il est impératif que la nouvelle version de l'application cliente soit installée sur chaque poste client.

Pour forcer la mise à jour, il vous suffit d'exclure les versions courantes des applications clientes (N-1 et précédentes) de l'intervalle des numéros de version compatibles avec l'application serveur. Dans ce cas, le mécanisme de mise à jour n'autorisera pas la connexion des applications clientes non mises à jour. Par exemple, si la nouvelle version de l'application client-serveur est 6, vous pouvez stipuler que toute application cliente ayant un numéro de version strictement inférieur à 6 ne sera pas autorisé à se connecter.

Le [numéro de version courante](#) est défini dans la page Client/Serveur du générateur d'application. Les intervalles de numéros autorisés sont définis dans le projet d'application via des [clés XML](#) spécifiques.

En cas d'erreur

Si 4D ne peut pas effectuer la mise à jour de l'application cliente, le poste client affiche le message d'erreur suivant : "La mise à jour de l'application cliente a échoué. L'application va maintenant quitter."

Les causes possibles de cette erreur sont multiples. Lorsque vous rencontrez ce message, il est conseillé de contrôler en premier lieu les paramètres suivants :

- Chemins d'accès : vérifiez la validité des chemins d'accès définis dans le projet d'application via la boîte de dialogue du Générateur d'applications ou via des clés XML (par exemple *ClientMacFolderToWin*). Vérifiez en particulier les chemins d'accès aux versions de 4D Volume Desktop.
- Privilèges lecture/écriture : sur la machine cliente, vérifiez que l'utilisateur courant dispose de droits d'accès en écriture pour l'application cliente mise à jour.

Fichiers générés

A l'issue du processus de génération d'une application client-serveur, vous devez trouver dans le dossier de destination un nouveau dossier nommé Client Server executable. Ce dossier contient deux sous-dossiers, <ApplicationName>Client et <ApplicationName>Server.

Ces dossiers ne sont pas générés si une erreur est survenue. Dans ce cas, ouvrez le [fichier d'historique](#) pour connaître la cause de l'erreur.

Le dossier <ApplicationName>Client contient la partie cliente de l'application correspondant à la plate-forme d'exécution du générateur d'application. Ce dossier doit être installé sur chaque poste client. Le dossier <ApplicationName>Server contient la partie serveur de l'application.

Le contenu de ces dossiers diffère en fonction de la plate-forme courante :

- *Sous Windows*, chaque dossier contient le fichier exécutable de l'application, nommé <ApplicationName>Client.exe pour la partie cliente et <ApplicationName>Server.exe pour la partie serveur, ainsi que les fichiers .rsr correspondants. Les dossiers contiennent également divers fichiers et dossiers nécessaires au fonctionnement des applications et les éléments personnalisés éventuellement placés dans les dossiers 4D Volume Desktop et 4D Server d'origine.

- Sous macOS, chaque dossier contient uniquement le progiciel de l'application, nommé <ApplicationName> Client pour la partie cliente et <ApplicationName> Server pour la partie serveur. Chaque progiciel contient tous les éléments nécessaires à son fonctionnement. Sous macOS, un progiciel est lancé via un double-clic.

Les progiciels macOS générés contiennent les mêmes éléments que les sous-dossiers Windows. You can display their contents (Control+click on the icon) in order to be able to modify them.

If you checked the "Allow automatic update of client application" option, an additional subfolder called *Upgrade4DClient* is added in the <ApplicationName>Server folder/package.

Si vous avez coché l'option "Permettre la mise à jour automatique de l'application cliente", un sous-dossier supplémentaire nommé *Upgrade4DClient* est ajouté dans le dossier/progiciel <ApplicationName>Server. Ce sous-dossier contient l'application cliente au format macOS et/ou Windows sous forme de fichier compressé. Ce fichier est utilisé lors de la mise à jour automatique des applications clientes.

Emplacements des fichiers Web

Si la partie serveur et/ou la partie cliente de votre application exécutable est utilisée en tant que serveur Web, les fichiers et dossiers requis par le serveur doivent être installés à des emplacements spécifiques. Ces éléments sont les suivants :

- fichiers *cert.pem* et *key.pem* (facultatifs) : ces fichiers sont utilisés pour les connexions TLS ainsi que par les commandes de cryptage des données,
- Dossier racine Web (DossierWeb) par défaut.

Des éléments doivent être installée :

- Sous Windows
 - Application serveur : dans le sous-dossier *Client Server executable/<ApplicationName>Server\Server Database*.
 - Application cliente : dans le sous-dossier *Client Server executable/<ApplicationName>Client*.
- sous macOS
 - Application serveur : à côté du progiciel <ApplicationName>Server.
 - Application cliente : à côté du progiciel <ApplicationName>Client.

Intégrer une structure compilée dans la partie cliente

4D permet d'intégrer une structure compilée dans une application cliente. Cette fonctionnalité peut être utilisée, par exemple, pour fournir aux utilisateurs une application "portail" donnant accès aux différentes applications serveur, via la commande `OPEN DATABASE` exécutant un fichier `.4dlink`.

Pour activer cette fonctionnalité, ajoutez les clés `DatabaseToEmbedInClientWinFolder` et/ou `DatabaseToEmbedInClientMacFolder` dans le fichier de configuration *buildApp*. Lorsque l'une de ces clés est présente, le processus de génération de l'application cliente génère une application monoposte : la structure compilée, au lieu du fichier *EnginedServer.4Dlink*, est placée dans le dossier "Database".

- Si un dossier "Data" par défaut existe dans l'application monoposte, une licence est intégrée.
- Si un dossier "Data" par défaut n'existe pas dans l'application monoposte, elle sera exécutée sans le fichier de données et sans licence.

Le scénario standard est le suivant :

1. Dans la boîte de dialogue du Générateur d'application, sélectionnez l'option "Générer une structure compilée" pour produire un .4DC ou un .4DZ pour utiliser l'application en monoposte.
2. Dans le fichier *buildApp.4DSettings* de l'application client-serveur, utilisez la ou les clés xml suivantes pour indiquer le chemin du dossier contenant l'application compilée monoposte :
 - `DatabaseToEmbedInClientWinFolder`
 - `DatabaseToEmbedInClientMacFolder`
3. Générez l'application client-serveur. Cela produira les effets suivants :
 - Le dossier de l'application monoposte est copié intégralement dans le dossier "Database" du client fusionné

- o le fichier `EnginedServer.4Dlink` du dossier "Database" n'est pas généré
- o les fichiers .4DC, .4DZ, .4DIndy de la copie de l'application monoposte sont renommés à l'aide du client fusionné
- o la clé `PublishName` n'est pas copiée dans le `info.plist` du client fusionné
- o si l'application monoposte ne possède pas de dossier "Data" par défaut, le client fusionné sera exécuté sans données.

Les fonctionnalités de mise à jour automatique de 4D Server (clé `CurrentVers`, commande `SET UPDATE FOLDER`, etc.) fonctionnent avec une application monoposte, comme avec une application distante standard. Lors de la connexion, l'application monoposte compare sa clé `CurrentVers` à la plage de version 4D Server. Si elle se trouve en dehors de plage, l'application cliente monoposte mise à jour est téléchargée depuis le serveur et l'Updater lance le processus de mise à jour locale.

Personnalisation des noms de dossier de cache client et/ou serveur

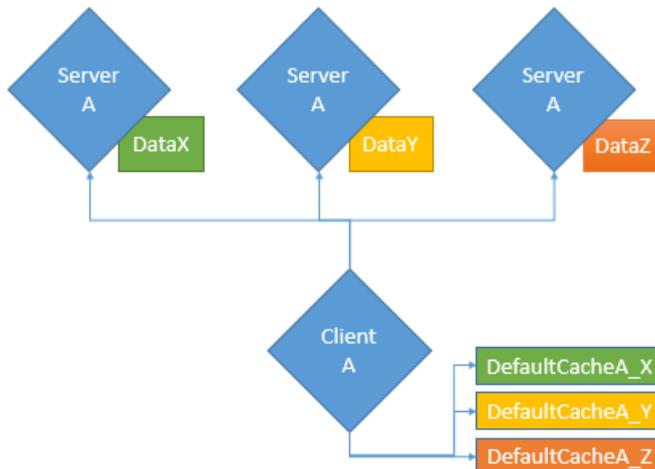
Les dossiers de cache client et serveur sont utilisés pour stocker des éléments partagés tels que des ressources ou des composants. Ils sont nécessaires pour gérer les échanges entre le serveur et les clients distants. Les applications client/serveur utilisent les chemins d'accès par défaut pour les dossiers de cache système client et serveur.

Dans certains cas spécifiques, vous devrez personnaliser les noms de ces dossiers pour implémenter des architectures spécifiques (voir ci-dessous). 4D vous fournit les clés `ClientServerSystemFolderName` et `ServerStructureFolderName` à définir dans le fichier de paramètres `buildApp`.

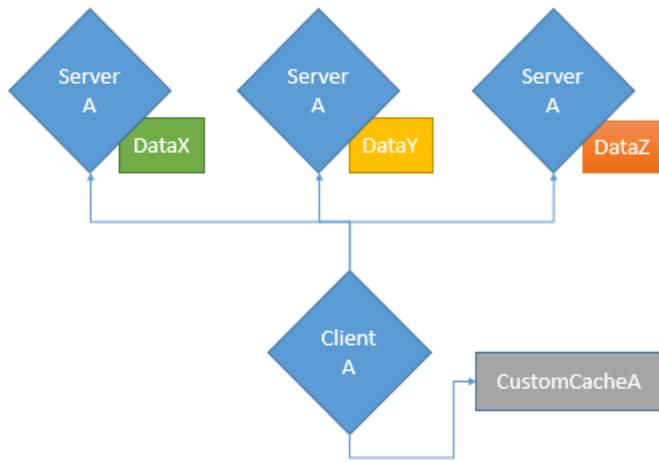
Dossier de cache client

La personnalisation du nom du dossier de cache côté client peut être utile lorsque votre application cliente est utilisée pour se connecter à plusieurs serveurs fusionnés qui sont similaires mais qui utilisent des ensembles de données différents. Dans ce cas, pour enregistrer plusieurs téléchargements inutiles de ressources locales identiques, vous pouvez utiliser le même dossier de cache local personnalisé.

- Configuration par défaut (*pour chaque connexion à un serveur, un dossier cache spécifique est téléchargé/mis à jour*) :



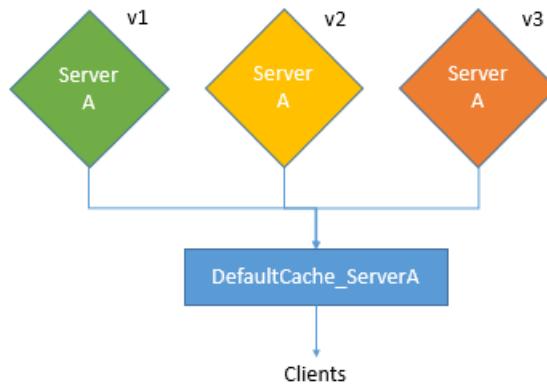
- À l'aide de la clé `ClientServerSystemFolderName` (*un seul dossier de cache est utilisé pour tous les serveurs*) :



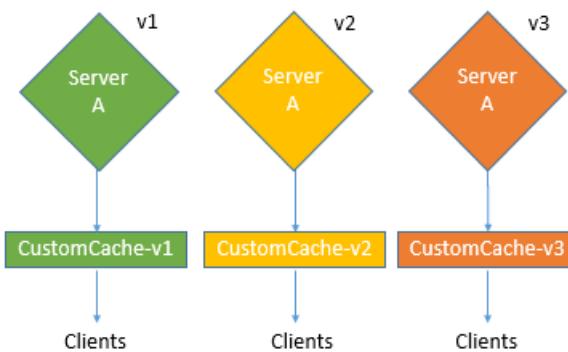
Dossier de cache du serveur

La personnalisation du nom du dossier de cache côté serveur est utile lorsque vous exécutez plusieurs applications serveur identiques créées avec différentes versions de 4D sur le même ordinateur. Si vous souhaitez que chaque serveur utilise son propre ensemble de ressources, vous devez personnaliser le dossier de cache du serveur.

- Configuration par défaut (*les mêmes applications serveur partagent le même dossier de cache*) :



- À l'aide de la clé `ServerStructureFolderName` (*un dossier de cache dédié est utilisé pour chaque application serveur*) :



Page Plugins et composants

On this tab, you set each [plug-in](#), [component](#), and [module](#) that you will use in your stand-alone or client/server application.

La page liste les éléments chargés par l'application 4D courante :

- La colonne Actif indique les éléments qui seront intégrés dans l'application générée. Par défaut, tous les éléments sont inclus. To exclude a plug-in, a component, or a module, deselect the check box next to it.
 - Plugins and components column - Displays the name of the plug-in/component/module.
 - ID column - Displays the element's identification number (if any).
 - Type column - Indicates the type of item: Plug-in, Component, or Module.

Adding plug-ins or components

Si vous souhaitez intégrer d'autres plug-ins ou composants dans l'application exécutable, il vous suffit de les placer dans un dossier PlugIns ou Components à côté de l'application 4D Volume Desktop ou de l'application 4D Server. Le mécanisme de copie du contenu du dossier de l'application source (cf. paragraphe [Personnaliser le dossier 4D Volume Desktop](#)) permet d'intégrer tout type de fichier à l'application exécutable.

En cas de conflit entre deux versions différentes d'un même plug-in (l'une chargée par 4D et l'autre placée dans le dossier de l'application source), la priorité revient au plug-in installé dans le dossier de 4D Volume Desktop/4D Server. En revanche, la présence de deux instances d'un même composant empêchera l'ouverture de l'application.

The use of plug-ins and/or components in a deployment version may require license numbers.

Deselecting modules

A module is a built-in code library used by 4D to control specific features. If you know that your built application does not use any of the features covered by a module, you can deselect it in the list to reduce the size of your application files.

Warning: Deselecting a module could prevent your built application from working as expected. If you are not sure about what each module does, leave them selected.

100% certain that a module is never called by your application, it is recommended to keep it selected.

The following optional modules can be deselected:

- CEF: Chromium embedded library. It is necessary to run [Web areas](#) that use the embedded rendering engine and [4D View Pro areas](#). Calling such areas when CEF is deselected will display blank areas and/or generate errors.
- MeCab: Library used for text indexing in Japanese language (see this [settings paragraph](#)). Deselecting this module will force text indexes to be rebuilt in Japanese language.

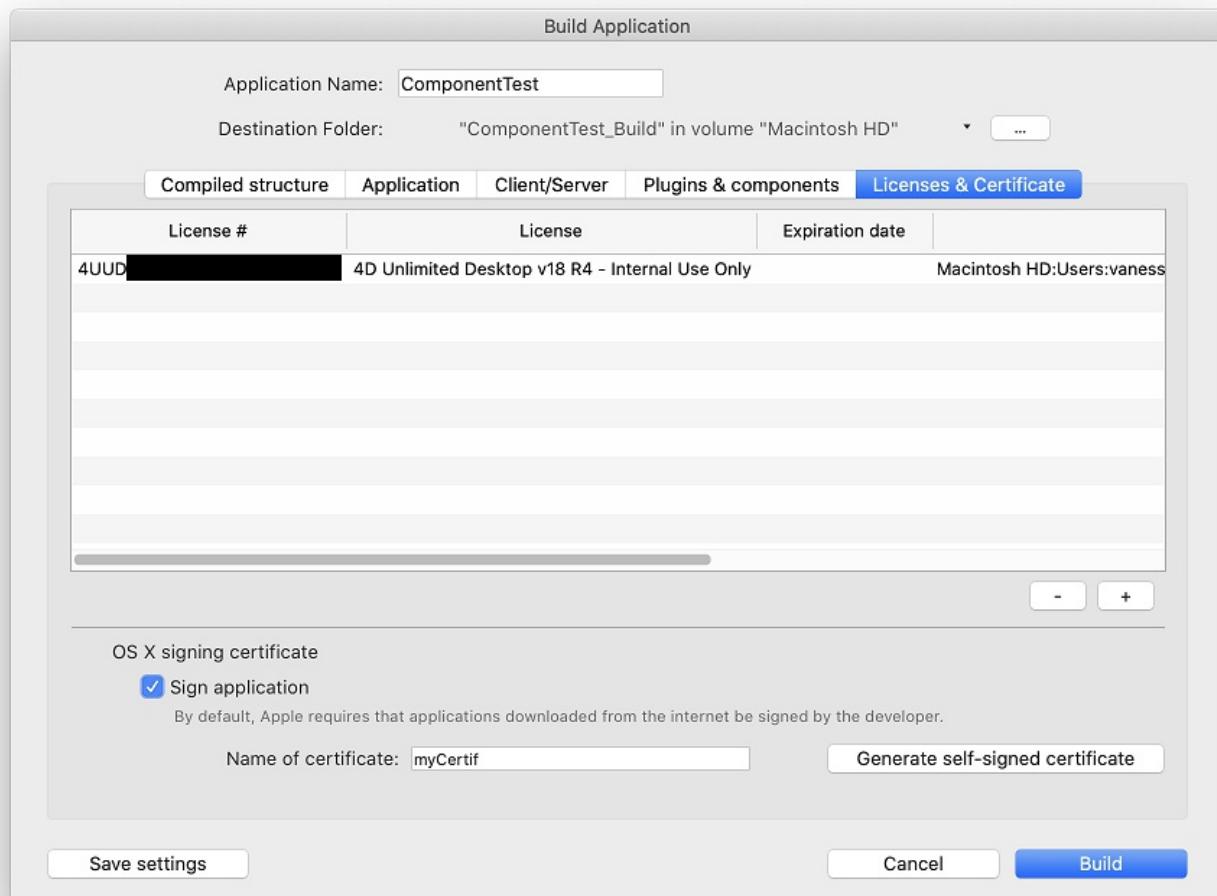
If you deselect MeCab for an application in Japanese language used on heterogeneous platforms, make sure to deselect it on both client/server build and [client application build](#) (for the concurrent platform), otherwise major malfunctions will occur in the application.

- PHP: Necessary to use PHP features and commands in 4D (see this [settings paragraph](#)).
- SpellChecker: Used for built-in [spellchecking features](#) and commands available for input areas and 4D Write Pro areas.
- 4D Updater: Controls the [automatic update](#) of client parts and is used by the `SET UPDATE FOLDER` command for [automated server updates](#).

Page Licences & Certificat

La page Licences & Certificat vous permet de :

- spécifier le ou les numéro(s) de licence que vous souhaitez intégrer dans votre application exécutable monoposte
- signer l'application à l'aide d'un certificat sous macOS.



Cet onglet affiche la liste des licences de déploiement disponibles que vous pouvez intégrer dans votre application. Par défaut, la liste est vide. Vous devez explicitement ajouter votre licence *4D Developer Professional* ainsi que chaque licence* *4D Desktop Volume** liée à utiliser dans l'application générée. Vous pouvez ajouter un numéro *4D Developer Professional* et ses licences associées autres que ceux en cours d'utilisation.

Pour ajouter ou supprimer des licences, utilisez les boutons [+] et [-] situés en bas de la fenêtre.

Lorsque vous cliquez sur le bouton [+], une boîte de dialogue d'ouverture de document apparaît, affichant par défaut le contenu du dossier *[Licenses]* de votre poste. Pour plus d'informations sur l'emplacement de ce dossier, reportez-vous à la commande [Get 4D folder](#).

Vous devez désigner les fichiers contenant votre licence *Developer* et ainsi que vos licences de déploiement. Ces fichiers ont été générés ou mis à jour au moment de l'acquisition de la licence *4D Developer Professional* et des licences *4D Desktop Volume*.

Une fois que vous avez sélectionné un fichier, la liste indique les caractéristiques de la licence qu'il contient.

- N° Licence : numéro de licence du produit
- Licence : nom du produit
- Date d'expiration : date d'expiration de la licence (le cas échéant)
- Chemin d'accès : emplacement sur le disque

Si la licence est invalide, un message vous le signale.

Vous pouvez désigner autant de fichiers valides que vous voulez. Lors de la génération de l'application exécutable, 4D utilisera les licences les plus appropriées.

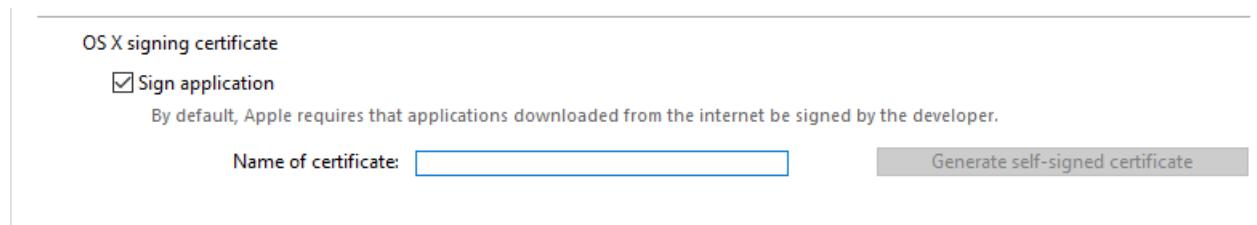
Des licences "R" dédiées sont requises pour générer des applications basées sur des versions "R-release" (les numéros de licence des produits "R" débutent par "R-4DDP").

A l'issue de la génération, un nouveau fichier de licence de déploiement est automatiquement inclus dans un dossier Licences placé à côté de l'application exécutable (Windows) ou dans le progiciel (macOS).

Certification des applications sous OS X

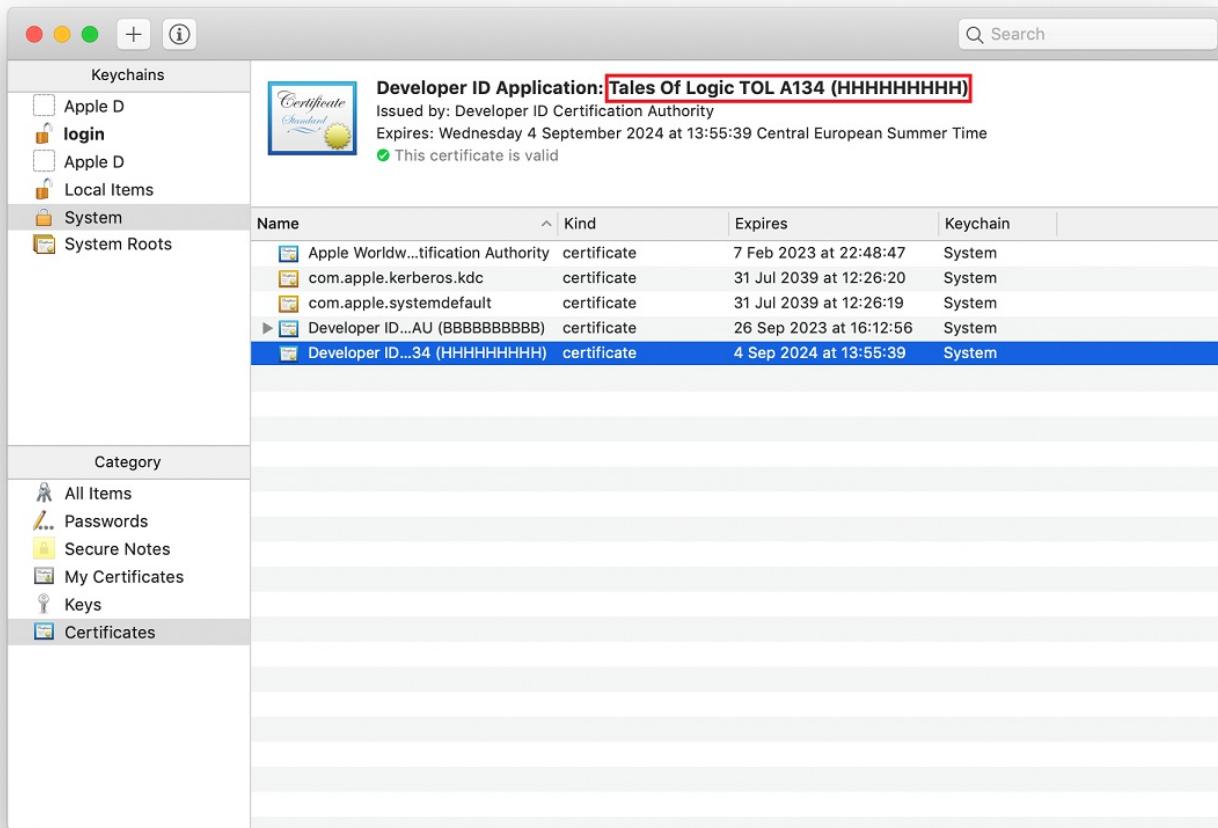
Le Générateur d'application permet de signer les applications 4D fusionnées sous macOS (applications monoposte, composants, 4D Server et parties clientes sous macOS). Signer une application permet d'autoriser son exécution par la fonctionnalité Gatekeeper de macOS lorsque l'option "Mac App Store et Développeurs identifiés" est sélectionnée (cf. "A propos de Gatekeeper" ci-dessous).

- Cochez l'option Signer l'application pour inclure la certification dans le processus de génération de l'application pour macOS :



L'option est affichée sous Windows et macOS mais n'est prise en compte que pour les versions macOS.

- Nom du certificat : saisissez dans cette zone le nom de votre certificat développeur validé par Apple. Le nom d'un certificat est généralement le nom du certificat dans l'utilitaire Trousseau d'accès (la partie en rouge dans l'exemple suivant) :



Pour obtenir un certificat de développeur auprès d'Apple, Inc., vous pouvez utiliser les commandes du menu Trousseaux d'accès ou vous connecter à la page suivante :

<http://developer.apple.com/library/mac/#documentation/Security/Conceptual/CodeSigningGuide/Procedures/Procedures.html>.

Le certificat requiert la présence de l'utilitaire codesign d'Apple. Cet utilitaire est fourni par défaut et se trouve généralement dans le dossier "/usr/bin/". En cas d'erreur, vérifiez que cet utilitaire est présent sur votre disque.

- Générer un certificat auto-signé - exécute le «Certificate Assistant» qui vous permet de générer un certificat auto-signé. Si vous ne disposez pas d'un certificat de développeur Apple, vous devez fournir un certificat auto-signé. Avec ce certificat, aucun message d'alerte ne s'affiche si l'application est déployée en interne. Si l'application est déployée en externe (c'est-à-dire via http ou e-mail), au lancement, macOS affiche un message d'alerte indiquant que le développeur de l'application n'est pas identifié. L'utilisateur peut "forcer" l'ouverture de l'application.

Dans le «Certificate Assistant», veillez à sélectionner les options appropriées :



4D recommande de souscrire au programme Apple Developer Program pour accéder aux "Developer Certificates" nécessaires à la notarisation des applications (voir ci-dessous).

A propos de Gatekeeper

Gatekeeper est une fonction de sécurité d'OS X permettant de contrôler l'exécution des applications téléchargées depuis Internet. Si une application téléchargée ne provient pas de l'Apple Store ou n'est pas signée, elle est rejetée et ne peut être lancée.

Sur les machines Apple Silicon, les **composants** 4D doivent être signés. Un composant non signé générera une erreur au démarrage de l'application ("lib4d-arm64.dylib can't be opened...").

L'option Signer l'application du Générateur d'application de 4D permet de générer des applications et des composants compatibles avec cette option par défaut.

À propos de la notarisation

La notarisation des applications est fortement recommandée par Apple à partir de macOS 10.14.5 (Mojave) et 10.15 (Catalina), car les applications non notariées déployées via Internet sont bloquées par défaut.

Les [fonctionnalités de signature intégrées](#) ont été mises à jour pour répondre à toutes les exigences d'Apple et permettre l'utilisation du service de notarisation d'Apple. La notarisation elle-même doit être réalisée par le développeur et est indépendante de 4D (à noter également qu'elle nécessite l'installation de Xcode). Veuillez vous référer à [ce post du blog 4D](#) qui fournit une description, par étapes, du processus de notarisation.

Pour plus d'informations sur le concept de notarisation, veuillez consulter [cette page sur le site Apple developer](#).

Personnaliser les icônes d'une application

4D associe une icône par défaut aux applications exécutables (monopostes et client-serveur). Vous pouvez cependant la personnaliser pour chaque application.

- Sous macOS - La personnalisation de l'icône de votre application est prise en charge par 4D lors de la construction de l'application exécutable. Pour cela, vous devez, avant la construction du fichier de l'application, créer un fichier d'icône (type *icns*) et le placer à côté du dossier de structure.

Apple, Inc. fournit un outil spécifique pour générer les fichiers d'icône *icns* (pour plus d'informations, veuillez consulter la [documentation d'Apple](#)).

Votre fichier d'icône doit avoir le même nom que le fichier du projet et comporter l'extension **.icns**.

- Sous Windows - La personnalisation de l'icône de votre application est prise en charge par 4D lors de la construction de l'application exécutable. Pour cela, vous devez, avant la construction du fichier de l'application, créer un fichier d'icône (extension *.ico*) et le placer à côté du fichier de structure interprété (fichier du projet).

Votre fichier d'icône doit avoir le même nom que le fichier de structure interprété et comporter l'extension *.ico*. 4D prend automatiquement ce fichier en compte lors de la génération de l'application exécutable.

Vous pouvez également définir des [clés XML](#) spécifiques dans le fichier `buildApp.4DSettings` pour désigner chaque icône devant être utilisée. Les clés suivantes sont disponibles :

- `RuntimeVLIconWinPath`
- `RuntimeVLIconMacPath`
- `ServerIconWinPath`
- `ServerIconMacPath`
- `ClientMacIconForMacPath`
- `ClientWinIconForMacPath`
- `ClientMacIconForWinPath`
- `ClientWinIconForWinPath`

Gestion des fichiers de données

Ouverture du fichier de données

Lorsqu'un utilisateur lance une application fusionnée ou une mise à jour (applications monopostes ou applications client-serveur), 4D va tenter d'ouvrir un fichier de données valide. Plusieurs emplacements sont successivement examinés par l'application.

La séquence de lancement d'une application fusionnée est la suivante :

1. 4D tente d'ouvrir le Dernier fichier de données ouvert, [comme décrit ci-dessous](#) (non applicable lors du lancement initial).
2. S'il n'est pas trouvé, 4D tente d'ouvrir en mode lecture seule le fichier de données situé dans le dossier de données par défaut au même niveau que le fichier *.4DC*.

3. S'il n'est pas trouvé, 4D tente d'ouvrir le fichier de données par défaut standard (même nom et même emplacement que le fichier .4DZ).
4. S'il n'est pas trouvé, 4D affiche une boîte de dialogue standard de sélection/création de fichier de données.

Dernier fichier de données ouvert

Chemin d'accès du dernier fichier de données

Toute application autonome ou serveur générée avec 4D stocke le chemin d'accès du dernier fichier de données ouvert dans le dossier de préférences de l'utilisateur de l'application.

L'emplacement du dossier de préférences de l'utilisateur de l'application correspond au chemin retourné par l'instruction suivante :

```
prefsUtilisateur:=Get 4D folder(Dossier 4D actif)
```

Le chemin d'accès du fichier de données est stocké dans un fichier dédié, nommé *lastDataPath.xml*.

Grâce à cette architecture, lorsque vous fournissez une mise à jour de votre application, le fichier de données de l'utilisateur local (le dernier fichier de données utilisé) est automatiquement ouvert dès le premier lancement.

Ce mécanisme est généralement adapté aux déploiements standard. Cependant, dans des cas spécifiques, par exemple si vous dupliquez vos applications fusionnées, vous pouvez avoir besoin de modifier la manière dont le fichier de données est lié à l'application.

Configurer le mode de liaison des données

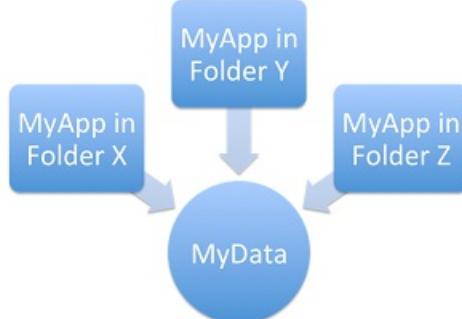
4D utilise automatiquement, avec vos applications compilées, le dernier fichier de données ouvert. Par défaut, le chemin d'accès du fichier de données est stocké dans le dossier de préférences de l'utilisateur de l'application et est lié au nom de l'application.

Ce fonctionnement peut s'avérer inadapté si vous souhaitez dupliquer une application fusionnée destinée à utiliser différents fichiers de données. En effet, les applications dupliquées vont en fait partager le même dossier de préférences de l'utilisateur et donc, toujours utiliser le même fichier de données -- même si le fichier de données est renommé, car l'application utilisera toujours le dernier fichier de données ouvert par l'application.

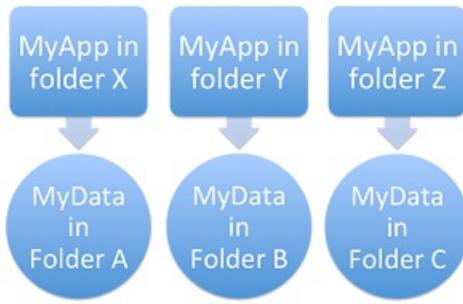
4D vous permet donc de lier votre chemin de fichier de données au chemin de l'application. Dans ce cas, le fichier de données sera relié via un chemin spécifique et ne sera plus simplement le dernier fichier utilisé. 4D vous permet donc de lier votre chemin de fichier de données au chemin de l'application.

Ce mode vous permet de dupliquer vos applications fusionnées sans rompre le lien vers le fichier de données. Cependant, avec cette option, si le package d'application est déplacé sur le disque, l'utilisateur sera invité à entrer un fichier de données, car le chemin de l'application ne correspondra plus à l'attribut "executablePath" (après qu'un utilisateur ait sélectionné un fichier de données, le fichier *lastDataPath.xml* est mis à jour en conséquence).

Duplication lorsque les données sont liées par le nom de l'application :



Duplication lorsque les données sont liées par le chemin de l'application :



Vous sélectionnez le mode de liaison des données lors de la phase de génération de l'application. Vous pouvez :

- Utiliser la [Page Application](#) ou la [Page Client/Serveur](#) de boîte de dialogue du Générateur d'applications.
- Utiliser la clé XML LastDataPathLookup (application monoposte ou application serveur).

Définir un dossier de données par défaut

4D vous permet de définir un fichier de données par défaut lors de la phase de construction de l'application. Au premier lancement de l'application, en l'absence de fichier local (cf. [séquence de lancement décrite ci-dessus](#)), le fichier de données par défaut est automatiquement ouvert silencieusement en mode lecture seule par 4D. Cela vous donne un meilleur contrôle sur la création et/ou l'ouverture des fichiers de données lors du premier lancement d'une application fusionnée.

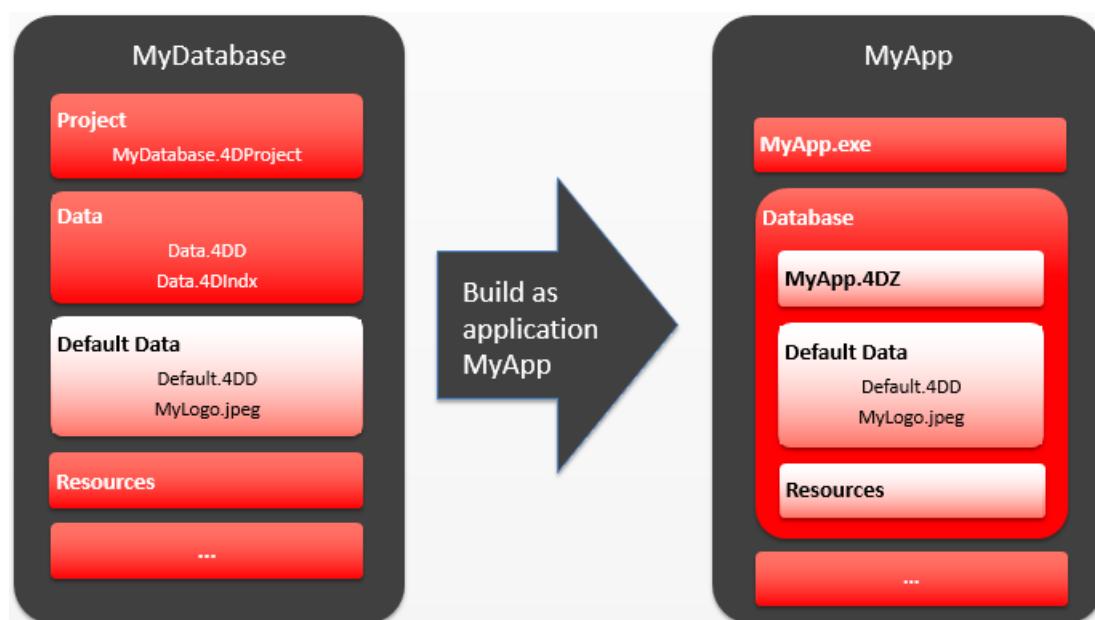
Plus particulièrement, il permet de répondre aux besoins suivants :

- Eviter l'affichage de la boîte de dialogue d'ouverture de fichier de données de 4D au lancement d'une nouvelle application fusionnée ou d'une mise à jour. Vous pouvez détecter, par exemple dans la , que le fichier de données par défaut a été ouvert et donc, exécuter votre propre code et/ou boîtes de dialogue permettant de créer ou de sélectionner un fichier de données local.
- Permettre la distribution d'applications fusionnées comportant des données en lecture seulement (par exemple des applications de démonstration).

Pour définir et utiliser un fichier de données par défaut :

- Vous devez fournir un fichier de données par défaut (nommé "Default.4DD") et le stocker dans un dossier spécifique (nommé "Default Data") à l'intérieur du dossier du projet d'application. Ce fichier doit être accompagné de tous les fichiers nécessaires, en fonction de la configuration du projet : index (.4DIndx), blobs externes, journal, etc. Il est de votre responsabilité de livrer un fichier de données par défaut valide. A noter que, comme le fichier de données par défaut est ouvert en mode lecture seule, il est recommandé de désélectionner l'option "Utiliser le fichier d'historique" dans le fichier de structure original avant de créer le fichier de données.
- Au moment de la génération de l'application, le dossier de données par défaut est intégré dans l'application fusionnée. Tous les fichiers présents dans ce dossier par défaut sont également embarqués.

Le schéma suivant illustre cette fonctionnalité :



Lorsque le fichier de données par défaut est détecté au premier lancement, il est silencieusement ouvert en mode lecture seulement, vous permettant ainsi d'exécuter toute opération personnalisée (à condition qu'elle ne modifie pas le fichier de données lui-même).

Gestion de la connexion des applications clientes

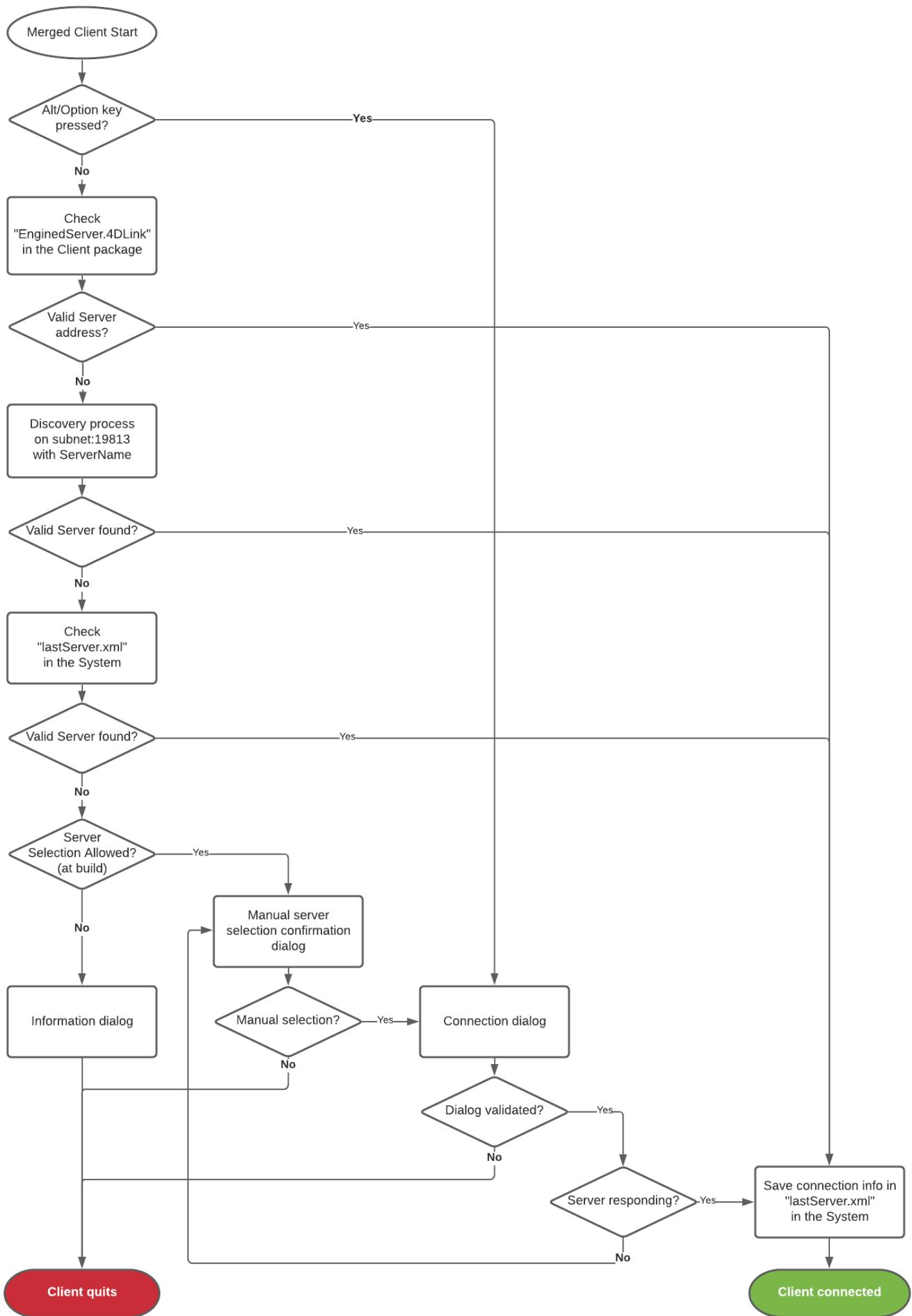
La gestion des connexions des applications clientes recouvre les mécanismes par lesquels une application cliente fusionnée se connectera au serveur cible, une fois en environnement de production.

Scénario de connexion

Le processus de connexion des applications clientes fusionnées prend en charge les cas où le serveur dédié n'est pas disponible. Le scénario du démarrage d'une application cliente 4D est le suivant :

1. L'application cliente tente de se connecter au serveur via le service de découverte (basé sur le nom du serveur, publié sur le même sous-réseau que l'application cliente).
OU
Si des informations de connexion valides sont présentes dans le fichier "EnginedServer.4DLink" à l'intérieur de son dossier, l'application cliente tente de se connecter à l'adresse du serveur spécifiée dans ce fichier.
2. En cas d'échec, l'application cliente tente de se connecter au serveur à l'aide des informations présentes dans le dossier de préférences utilisateur de l'application (fichier "lastServer.xml", cf. dernière étape).
3. En cas d'échec, l'application cliente affiche une boîte de dialogue d'erreur de connexion.
 - Si l'utilisateur clique sur le bouton Sélectionner... (lorsqu'il été autorisé par le développeur 4D au moment de la génération de l'application, voir ci-dessous), la boîte de dialogue standard "Connexion au serveur" est affichée.
 - Si l'utilisateur clique sur le bouton Quitter, l'application client quitte.
4. Si la connexion est établie avec succès, les paramètres de cette connexion sont sauvegardés dans le dossier de préférences utilisateur de l'application cliente, ce qui permettra de les réutiliser ultérieurement en cas de besoin.

La procédure complète est décrite dans le diagramme suivant :



Sauvegarde du chemin du dernier serveur

Le chemin du dernier serveur utilisé est automatiquement sauvegardé dans un fichier nommé "lastServer.xml" placé

dans le dossier de préférences utilisateur de l'application cliente. Ce dossier est situé à l'emplacement suivant :

```
prefsUtilisateur:=Get 4D folder(Dossier 4D actif)
```

Ce mécanisme permet de prendre en charge le cas où le serveur cible primaire est temporairement indisponible pour une raison quelconque (par exemple pour une opération de maintenance). Lorsque ce cas se produit pour la première fois, la boîte de dialogue de sélection de serveur est affichée (si elle est autorisée, cf. ci-dessous) et l'utilisateur peut manuellement sélectionner un serveur alternatif, dont le chemin est alors sauvegardé si la connexion est établie et validée. Toute indisponibilité ultérieure sera alors automatiquement prise en charge à l'aide des paramètres de connexion présents dans le fichier "lastServer.xml".

- Lorsque les applications clientes ne peuvent pas bénéficier du service de découverte, par exemple à cause de la configuration réseau, il reste recommandé que le développeur indique un nom d'hôte au cours de la génération de l'application à l'aide de la clé **IPAddress** dans le fichier "BuildApp.xml". Le mécanisme de sauvegarde du chemin du dernier serveur est conçu pour les cas d'indisponibilité temporaire uniquement.
- Dans tous les cas, il est possible de maintenir la touche Alt/Option au démarrage de l'application cliente afin d'afficher la boîte de dialogue de sélection du serveur.

Accès à la boîte de dialogue de sélection de serveur en cas d'erreur

Vous pouvez choisir d'afficher ou non la boîte de dialogue standard de sélection de serveur sur les applications clientes fusionnées lorsque le serveur ne répond pas. La configuration dans ce cas dépend de la valeur de la clé XML **ServerSelectionAllowed** sur le poste qui génère l'application client/serveur. Vous disposez de trois possibilités :

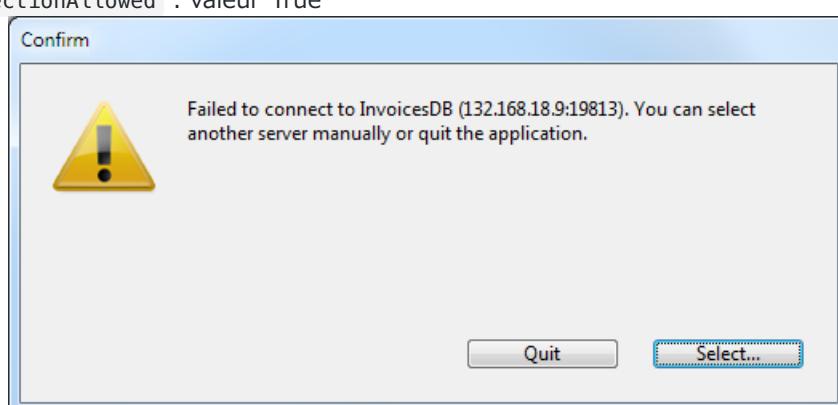
- Affichage d'un message d'erreur sans accès possible à la boîte de dialogue de sélection de serveur. Fonctionnement par défaut. L'application peut uniquement quitter.

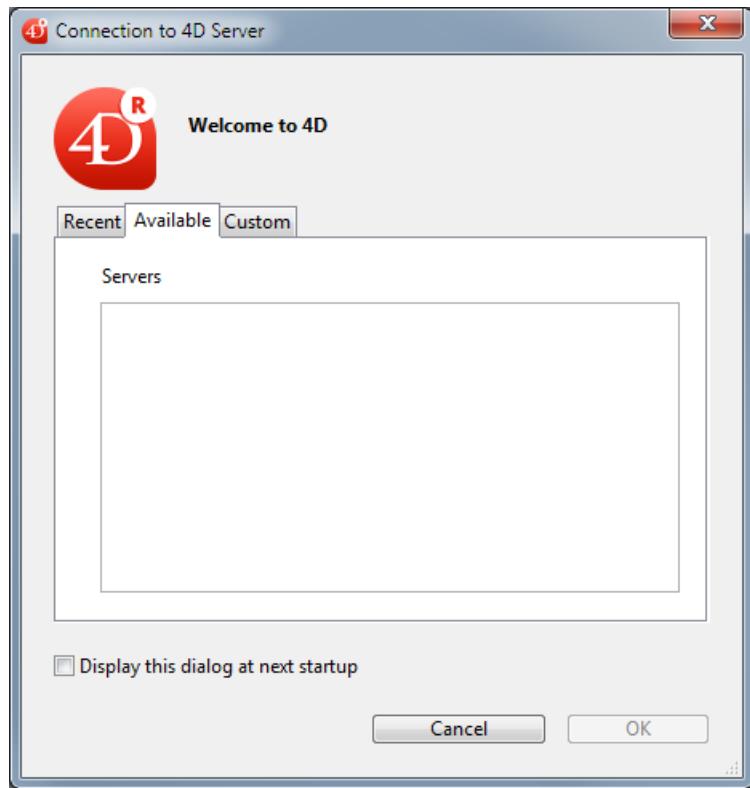
Clé Xml **ServerSelectionAllowed** : valeur False ou clé omise



- Affichage d'un message d'erreur avec accès possible à la boîte de dialogue de sélection de serveur. L'utilisateur peut accéder à la fenêtre de sélection de serveur en cliquant sur le bouton Sélectionner...

Clé XML **ServerSelectionAllowed** : valeur True





Automatic updating of server or single-user applications

In principle, updating server applications or merged single-user applications require user intervention (or programming custom system routines): whenever a new version of the merged application is available, you have to exit the application in production and manually replace the old files with the new ones; then restart the application and select the current data file.

You can automate this procedure to a large extent using the following language commands: `SET UPDATE FOLDER`, `RESTART 4D`, and also `Get last update log path` for monitoring operations. The idea is to implement a function in your 4D application triggering the automatic update sequence described below. It can be a menu command or a process running in the background and checking at regular intervals for the presence of an archive on a server.

You also have XML keys to elevate installation privileges so that you can use protected files under Windows (see the [4D XML Keys BuildApplication manual](#)).

Here is the scenario for updating a server or merged single-user application:

1. You transfer, for example using an HTTP server, the new version of the server application or the merged single-user application onto the machine in production.
2. In the application in production, you call the `SET UPDATE FOLDER` command: this command designates the location of the folder where the "pending" update of the current application is found. Optionally, you can copy in this folder the custom elements of the version in production (user files).
3. In the application in production, call the `RESTART 4D` command: this command automatically triggers execution of a utility program named "updater" that exits the current application, replaces it using the "pending" update if one is specified, and restarts the application with the current data file. The former version is renamed.

This sequence is compatible with Windows server applications run as a Service.

Update log

The installation procedure produces a log file detailing the update operations of merged applications (client, server or single-user) on the target machines. This file is useful for analyzing any errors that occur during the installation process.

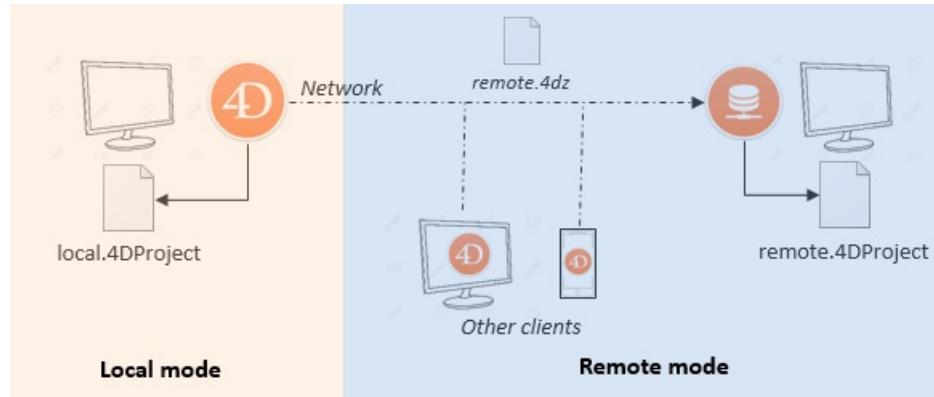
The update log is named `YYYY-MM-DD_HH-MM-SS_log_X.txt`, for example, `2021-08-25_14-23-00_log_1.txt` for a file created on August 25, 2021 at 14:23.

This file is created in the "Updater" application folder, within the system user folder. You can find out the location of this file at any time using the [Get last update log path](#) command.

Client/Server Management

4D Desktop applications can be used in a Client/Server configuration, either as merged client/server applications or as remote projects.

- merged client/server applications are generated by the [Build Application manager](#). They are used for application deployments.
- remote projects are [.4DProject](#) files opened by 4D Server and accessed with 4D in remote mode. The server sends a [.4dz](#) version of the project ([compressed format](#)) to the remote 4D, thus structure files are read-only. This configuration is usually used for application testing.



Connecting to a remote project from the same machine as 4D Server allows modifying the project files. This [specific feature](#) allows to develop a client/server application in the same context as the deployment context.

Opening a merged client/server application

A merged client/server application is customized and its starting is simplified:

- Pour lancer la partie serveur, l'utilisateur double-clique simplement sur l'application serveur : il n'est pas nécessaire de sélectionner le fichier projet.
- Pour lancer la partie cliente, l'utilisateur double-clique simplement sur l'application cliente, qui se connecte directement à l'application serveur :

These principles are detailed in the [Build Application](#) page.

Ouvrir un projet distant

La première fois que vous vous connectez à un projet 4D Server via un 4D distant, vous utiliserez généralement la boîte de dialogue de connexion standard. Thereafter, you will be able to connect directly using the Open Recent Projects menu or a 4DLink shortcut file.

Pour vous connecter à distance à un projet 4D Server :

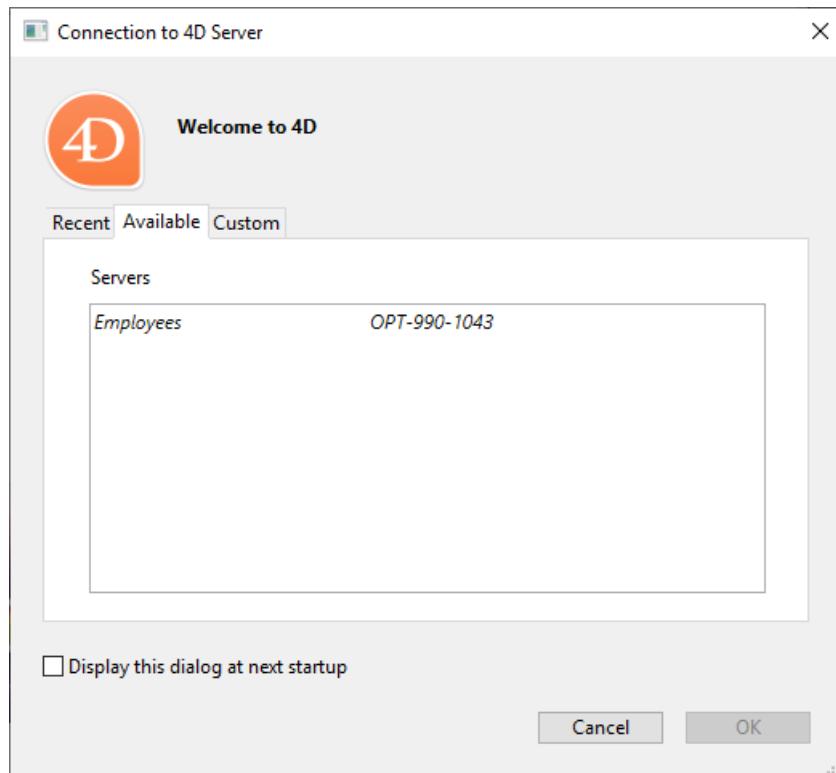
1. Sélectionnez Se connecter à 4D Server dans la boîte de dialogue de l'Assistant de bienvenue,

OR

Sélectionnez Ouvrir > Projet distant... à partir du menu Fichier ou du bouton Ouvrir de la barre d'outils.

La boîte de dialogue de connexion à 4D Server apparaît. Cette boîte de dialogue comporte trois onglets : Récent, Disponible et Personnalisé.

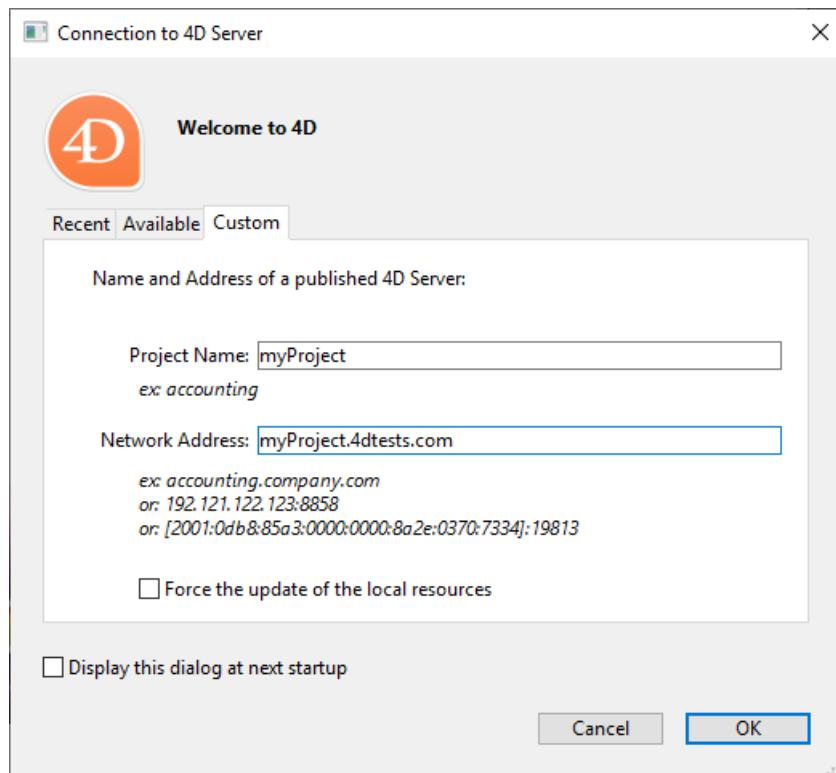
Si 4D Server est connecté au même réseau que le 4D distant, sélectionnez Disponible. 4D Server inclut un système de diffusion TCP/IP intégré qui, par défaut, publie le nom des projets 4D Server disponibles sur le réseau. La liste est triée par ordre d'apparition et est mise à jour dynamiquement.



Pour vous connecter à un serveur de la liste, double-cliquez sur son nom ou sélectionnez-le et cliquez sur le bouton OK.

Un accent circonflexe (^) est placé avant le nom des projets publiés avec l'option de chiffrement activée.

Si le projet publié n'est pas affiché dans la liste Disponible, sélectionnez Personnalisé. La page Personnalisé vous permet de vous connecter à un serveur publié sur le réseau en utilisant son adresse réseau et en lui attribuant un nom personnalisé.



- Nom du projet : définit le nom local du projet 4D Server. Ce nom sera utilisé dans la page Récent pour faire référence au projet.
- Adresse réseau : L'adresse IP de la machine sur laquelle le 4D Server a été lancé.

Si deux serveurs sont exécutés simultanément sur la même machine, l'adresse IP doit être suivie de deux points et d'un numéro de port, par exemple : 192.168.92.104:19814 .

Par défaut, le port de publication d'un 4D Server est 19813. Ce numéro peut être modifié dans les paramètres du projet.

Une fois que cette page attribue un serveur, cliquez sur le bouton **OK** pour vous connecter au serveur.

Si le projet est publié avec l'option de chiffrement activée, vous devez ajouter un accent circonflexe (^) avant le nom, sinon la connexion sera refusée. Pour plus d'informations, reportez-vous à la section Chiffrement des connexions client/serveur.

Une fois la connexion au serveur établie, le projet distant sera répertorié dans l'onglet Récent.

Mettre à jour des fichiers de projet sur le serveur

4D Server automatically creates and sends the remote machines a [.4dz version](#) of the *.4DProject* project file (not compressed) in interpreted mode.

- Une version .4dz mise à jour du projet est automatiquement produite lorsque cela est nécessaire, c'est-à-dire lorsque le projet a été modifié et rechargé par 4D Server. Le projet est rechargé :
 - automatiquement, lorsque la fenêtre de l'application 4D Server arrive à l'avant de l'OS ou lorsque l'application 4D sur la même machine enregistre une modification (voir ci-dessous).
 - lorsque la commande `RELOAD PROJECT` est exécutée. L'appel de cette commande est nécessaire lorsque, par exemple, vous avez extrait une nouvelle version du projet depuis la plateforme de contrôle de version.

Mettre à jour des fichiers de projet sur les machines distantes

Lorsqu'une version .4dz mise à jour du projet a été produite sur 4D Server, les machines 4D distantes connectées doivent se déconnecter et se reconnecter à 4D Server afin de bénéficier de la version mise à jour.

Using 4D and 4D Server on the same machine

Lorsque 4D se connecte à un 4D Server sur la même machine, l'application se comporte comme 4D en mode monoposte et l'environnement de développement permet d'éditer les fichiers du projet. This feature allows you to develop a client/server application in the same context as the deployment context.

A chaque fois que 4D effectue une action Enregistrer tout depuis l'environnement de développement (explicitement depuis le menu Fichier ou implicitement en passant en mode application par exemple), 4D Server recharge de manière synchrone les fichiers du projet. 4D attend que 4D Server termine le rechargement des fichiers du projet avant de continuer.

However, you need to pay attention to the following behavior differences compared to [standard project architecture](#):

- le dossier `userPreferences.{username}` utilisé par 4D ne correspond pas au même dossier utilisé par 4D Server dans le dossier projet. Au lieu de cela, il s'agit d'un dossier dédié, nommé "userPreferences", stocké dans le dossier système du projet (c'est-à-dire au même emplacement que lors de l'ouverture d'un projet .4dz).
- le dossier utilisé par 4D pour les données dérivées n'est pas le dossier "DerivedData" du dossier projet. Il s'agit plutôt d'un dossier dédié nommé "DerivedDataRemote" situé dans le dossier système du projet.
- le fichier `catalog.4DCatalog` n'est pas édité par 4D mais par 4D Server. Les informations du catalogue sont synchronisées à l'aide des requêtes client/serveur
- le fichier `directory.json` n'est pas édité par 4D mais par 4D Server. Les informations du répertoire sont synchronisées à l'aide des requêtes client/serveur
- 4D utilise ses propres composants internes et plug-ins au lieu de ceux de 4D Server.

Il n'est pas recommandé d'installer des plug-ins ou des composants au niveau de l'application 4D ou 4D Server.

Composants de développement

Un composant 4D est un ensemble de fonctions, méthodes et de formulaires 4D représentant une ou plusieurs fonctionnalité(s) qu'il est possible [d'installer et d'utiliser dans des applications 4D](#). Par exemple, vous pouvez développer un composant 4D de courrier électronique gérant tous les aspects de l'envoi, la réception et le stockage d'emails au sein des applications 4D.

Vous pouvez développer des composants 4D pour vos propres besoins et les garder privés. Vous pouvez également [partager vos composants avec la communauté 4D](#).

Définitions

- Projet utilisé comme matrice : Projet 4D utilisé pour le développement du composant. C'est un projet standard, sans attribut spécifique. Il constitue un seul composant.
- Projet hôte : projet dans lequel un composant est installé et utilisé.
- Composant : Projet utilisé comme matrice, pouvant être compilé ou [généré](#), copié dans le dossier [Components](#) de l'application hôte et dont le contenu est utilisé dans l'application hôte.

Principes de base

La création et l'installation des composants 4D s'effectuent directement depuis 4D :

- Pour installer un composant, il suffit de copier les fichiers du composant dans le dossier [Components du projet](#). Vous pouvez utiliser des alias ou des raccourcis.
- Un projet peut être à la fois "matrice" et "hôte", c'est-à-dire qu'un projet utilisé comme matrice peut lui-même utiliser un ou plusieurs composants. En revanche, un composant ne peut pas lui-même utiliser de "sous-composants".
- Un composant peut appeler la plupart des éléments 4D : des classes, des fonctions, des méthodes projet, des formulaires projet, des barres de menus, des listes à choix multiples, etc. Il ne peut pas appeler des méthodes base et des triggers.
- Il n'est pas possible d'exploiter le datastore, des tables standard ou des fichiers de données dans les composants 4D. En revanche, un composant peut créer et/ou utiliser des tables, des champs et des fichiers de données via les mécanismes des bases externes. Les bases externes sont des bases 4D indépendantes manipulées via les commandes SQL.
- Un projet hôte fonctionnant en mode interprété peut utiliser des composants interprétés ou compilés. Un projet hôte fonctionnant en mode compilé ne peut pas utiliser de composants interprétés. Dans ce cas, seuls les composants compilés peuvent être utilisés.

Portée des commandes du langage

Hormis les [Commandes non utilisables](#), un composant peut utiliser toute commande du langage 4D.

Lorsqu'elles sont appelées depuis un composant, les commandes s'exécutent dans le contexte du composant, à l'exception de la commande `EXECUTE METHOD` ou `EXECUTE FORMULA` qui utilise le contexte de la méthode désignée par la commande. A noter également que les commandes de lecture du thème "Utilisateurs et groupes" sont utilisables depuis un composant mais lisent les utilisateurs et les groupes du projet hôte (un composant n'a pas d'utilisateurs et groupes propres).

Les commandes `EXECUTE METHOD` et `Get database parameter` constituent aussi une exception à ce principe : leur portée est globale à l'application. Lorsque ces commandes sont appelées depuis un composant, elles s'appliquent au projet d'application hôte.

Par ailleurs, des dispositions spécifiques sont définies pour les commandes `Structure file` et `Get 4D folder` lorsqu'elles sont utilisées dans le cadre des composants.

La commande `COMPONENT LIST` permet de connaître la liste des composants chargés par le projet hôte.

Commandes non utilisables

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

- ON EVENT CALL
- Method called on event
- SET PICTURE TO LIBRARY
- REMOVE PICTURE FROM LIBRARY
- SAVE LIST
- ARRAY TO LIST
- EDIT FORM
- CREATE USER FORM
- DELETE USER FORM
- CHANGE PASSWORD
- EDIT ACCESS
- Set group properties
- Set user properties
- DELETE USER
- CHANGE LICENSES
- BLOB TO USERS
- SET PLUGIN ACCESS

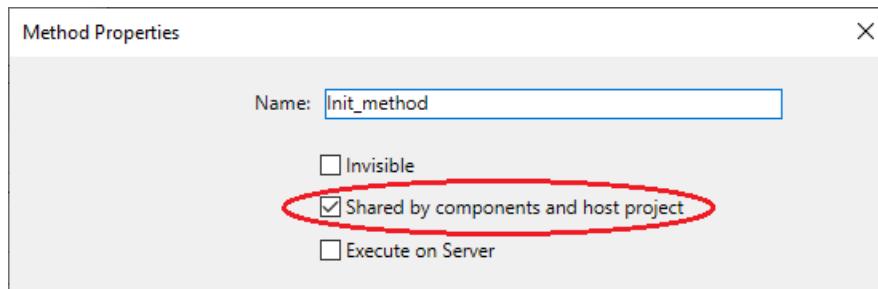
Notes :

- La commande `Table` du formulaire courant retourne `Nil` lorsqu'elle est appelée dans le contexte d'un formulaire projet. Par conséquent, elle ne peut pas être utilisée dans un composant.
- Les commandes SQL de définition de données (`CREATE TABLE`, `DROP TABLE`, etc.) ne peuvent pas être utilisées dans les composants. Elles sont néanmoins prises en charge avec des bases de données externes (voir la commande SQL `CREATE DATABASE`).

Partage des méthodes projet

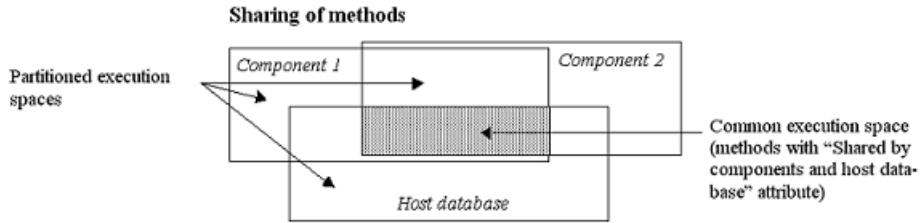
Toutes les méthodes projet d'un projet utilisé comme matrice sont par définition incluses dans le composant (le projet est le composant), ce qui signifie qu'elles peuvent être appelées et exécutées dans le composant.

En revanche, par défaut ces méthodes projet ne seront ni visibles ni appelables par le projet hôte. Dans le projet utilisé comme matrice, vous devez désigner explicitement les méthodes que vous souhaitez partager avec le projet hôte en cochant la case Partagée par les composants et le projet hôte dans la boîte de dialogue des propriétés de la méthode :



Les méthodes projet partagées peuvent être appelées dans le code du projet hôte (mais elles ne pourront pas être modifiées dans l'éditeur de méthodes du projet hôte). Ces méthodes constituent les points d'entrée du composant.

A l'inverse, pour des raisons de sécurité, par défaut un composant ne peut pas exécuter de méthode projet appartenant au projet hôte. Dans certains cas, vous pourrez avoir besoin d'autoriser un composant à accéder à des méthodes projet de votre projet hôte. In certain cases, you may need to allow a component to access the project methods of your host project.



Une fois que les méthodes projets hôtes sont disponibles pour les composants, vous pouvez exécuter une méthode hôte depuis l'intérieur d'un composant à l'aide des commandes `EXECUTE FORMULA` ou `EXECUTE METHOD`. Par exemple :

```
// Méthode hôte
component_method("host_method_name")
```

```
// component_method
C_TEXT($1)
EXECUTE METHOD($1)
```

Une base hôte interprétée qui contient des composants interprétés peut être compilée ou soumise à un contrôle syntaxique si elle n'appelle pas de méthodes du composant interprété. Dans le cas contraire, une boîte de dialogue d'avertissement apparaît lorsque vous tentez de lancer la compilation ou un contrôle syntaxique et il ne sera pas possible d'effectuer l'opération.

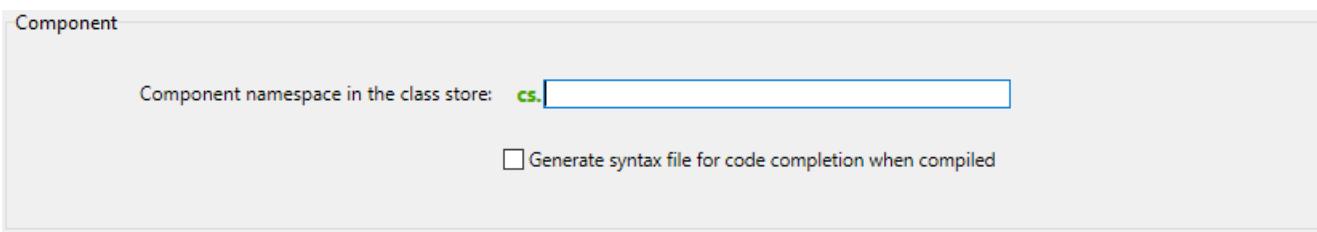
A noter qu'une méthode interprétée peut appeler une méthode compilée, mais pas l'inverse, sauf si vous utilisez les commandes `EXECUTE METHOD` et `EXECUTE FORMULA`.

Partage des classes et des fonctions

Par défaut, les classes et fonctions des composants ne peuvent pas être appelées à partir de l'éditeur de méthodes 4D du projet hôte. By default, component classes and functions cannot be called from the 4D Code Editor of the host project. En outre, vous pouvez contrôler la manière dont les classes et les fonctions des composants sont suggérées dans l'éditeur de méthodes de l'hôte.

Déclaration du namespace

Pour permettre aux classes et aux fonctions de votre composant d'être exposées dans les projets hôtes, saisissez une valeur dans [Component namespace in the class store](#) dans la page [Général](#) des Propriétés du projet utilisé comme matrice. Par défaut, l'espace est vide : les classes du composant ne sont pas disponibles en dehors du contexte du composant.



Un *namespace* garantit qu'aucun conflit n'émerge lorsqu'un projet hôte utilise différents composants dont les classes ou les fonctions ont des noms identiques. Un namespace doit être conforme aux [règles de dénomination des propriétés](#).

Lorsque vous saisissez une valeur, vous déclarez que les classes et les fonctions du composant seront disponibles dans [user class store \(cs\)](#) du code du projet hôte, par le biais du namespace `cs.<value>`. Par exemple, si vous entrez "eGeometry" comme namespace, en supposant que vous avez créé une classe `Rectangle` contenant une fonction `getArea()`, une fois votre projet installé comme composant, le développeur du projet hôte peut écrire :

```
//dans le projet hôte
var $rect: cs.eGeometry.Rectangle
$rect:=cs.eGeometry.Rectangle.new(10;20)
$area:=$rect.getArea()
```

Bien entendu, il est recommandé d'utiliser un nom distinctif pour éviter tout conflit. Si une classe utilisateur portant le même nom qu'un composant existe déjà dans le projet, la classe utilisateur est prise en compte et les classes du composant sont ignorées.

Les classes ORDA d'un composant ne sont pas disponibles dans le projet hôte. Par exemple, s'il existe une dataclass nommée Employees dans votre composant, vous ne pourrez pas utiliser une classe "cs.Mycomponent.Employee" dans le projet hôte.

Classes cachées

Comme dans tout projet, vous pouvez créer des classes et des fonctions cachées dans le composant en préfixant les noms par un caractère de soulignement ou ("_"). Lorsqu'un [namespace est défini](#), les classes et fonctions cachées du composant n'apparaîtront pas comme des suggestions lors de l'utilisation de la complétion de code.

Notez cependant qu'elles peuvent toujours être utilisées si vous connaissez leurs noms. Par exemple, la syntaxe suivante est valable même si la classe `_Rectangle` est cachée :

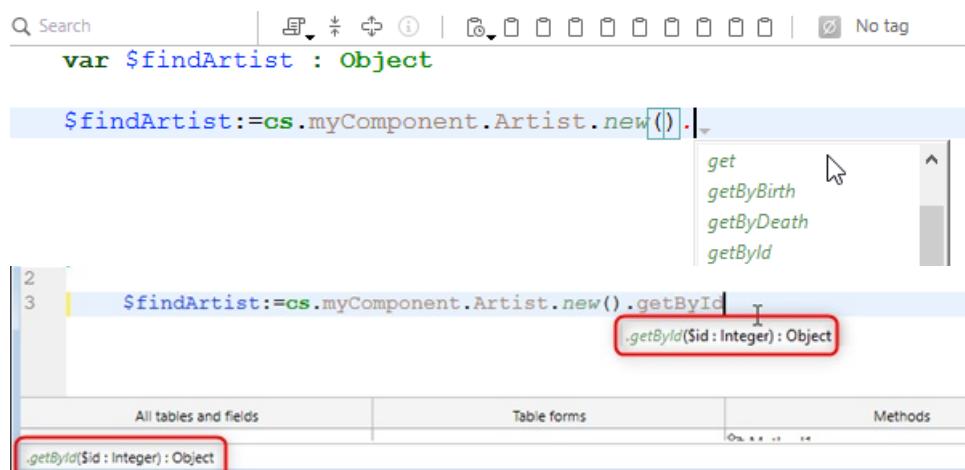
```
$rect:=cs.eGeometry._Rectangle.new(10;20)
```

Les fonctions non cachées à l'intérieur d'une classe cachée apparaissent comme des suggestions lorsque vous utilisez la complétion de code avec une classe qui en [hérite](#). Par exemple, si un composant possède une classe `Teacher` qui hérite d'une classe `_Person`, la complétion de code pour `Teacher` suggère des fonctions non cachées de `_Person`.

Complétion de code pour les composants compilés

Pour faciliter l'utilisation de votre composant par les développeurs, vous pouvez cocher l'option [Generate syntax file for code completion when compiled](#) dans la page [Général](#) des Propriétés du projet utilisé comme matrice.

Un fichier de syntaxe (format JSON) est alors automatiquement créé lors de la phase de compilation, complété par la syntaxe des classes, fonctions et [méthodes 'exposed'](#) de votre composant, et placé dans le dossier `¥Resources¥en.lproj` du projet du composant. 4D utilise le contenu de ce fichier syntaxique pour générer une aide contextuelle dans l'éditeur de code, telle que la complétion de code et la syntaxe des fonctions :



Si vous ne saisissez pas de [namespace](#), les ressources des classes et des méthodes 'exposed' ne sont pas générées, même si l'option de fichier de syntaxe est cochée.

Passage de variables

Les composants et les projets hôtes ne partagent pas de variables locales, process ou interprocess. The local, process and interprocess variables are not shared between components and host projects.

Exemple utilisant un tableau :

```
//Dans le projet hôte :  
ARRAY INTEGER(MyArray;10)  
AMethod(->MyArray)  
  
//Dans le composant, la méthode projet UneMéthode contient :  
APPEND TO ARRAY($1->;2)
```

Exemples utilisant des variables :

```
C_TEXT(myvariable)  
component_method1(->myvariable)
```

```
C_POINTER($p)  
$p:=component_method2(...)
```

Sans pointeur, un composant peut toujours accéder à la valeur d'une variable de base hôte (mais pas à la variable elle-même) et vice versa :

```
//Dans la base hôte  
C_TEXT($input_t)  
$input_t:="DoSomething"  
component_method($input_t)  
// component_method obtient "DoSomething" in $1 (mais pas la variable $input_t)
```

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

- La commande `Get pointer` ne retournera pas un pointeur vers une variable du projet hôte si elle est appelée depuis un composant et inversement.
- L'usage de pointeurs dans ce cas doit respecter le principe suivant : l'interpréteur peut dépointer un pointeur construit en mode compilé mais à l'inverse, en mode compilé, il n'est pas possible de dépointer un pointeur construit en mode interprété. L'usage de pointeurs dans ce cas doit respecter le principe suivant : l'interpréteur peut dépointer un pointeur construit en mode compilé mais à l'inverse, en mode compilé, il n'est pas possible de dépointer un pointeur construit en mode interprété. Illustrons ce principe par l'exemple suivant : soient deux composants, C (compilé) et I (interprété) installés dans le même projet hôte.
- Si le composant C définit la variable `mavarC`, le composant I peut accéder à la valeur de cette variable en utilisant le pointeur `->mavarC`.
- Si le composant I définit la variable `mavarI`, le composant C ne peut pas accéder à cette variable en utilisant le pointeur `->mavarI`. Cette syntaxe provoque une erreur d'exécution.
- La comparaison de pointeurs via la commande `RESOLVE POINTER` est déconseillée avec les composants car le principe de cloisonnement des variables autorise la coexistence de variables de même nom mais au contenu radicalement différent dans un composant et le projet hôte (ou un autre composant). Le type de la variable peut même être différent dans les deux contextes. Si les pointeurs `monptr1` et `monptr2` pointent chacun sur une variable, la comparaison suivante produira un résultat erroné :

```

RESOLVE_POINTER(monptr1;vNomVar1;vnumtable1;vnumchamp1)
RESOLVE_POINTER(monptr2;vNomVar2;vnumtable2;vnumchamp2)
If(vNomVar1=vNomVar2)
//Ce test retourne Vrai alors que les variables sont différentes

```

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

```
If(monptr1=monptr2) //Ce test retourne Faux
```

Gestion des erreurs

Une [méthode de gestion d'erreurs](#) installée par la commande `ON ERR CALL` s'applique à l'application en cours d'exécution uniquement. En cas d'erreur générée par un composant, la méthode d'appel sur erreur `ON ERR CALL` du projet hôte n'est pas appelée, et inversement.

Accès aux tables du projet hôte

Bien que les composants ne puissent pas utiliser de tables, les pointeurs permettent au projet hôte et au composant de communiquer. Par exemple, voici une méthode pouvant être appelée depuis un composant :

```
// appeler une méthode composant
methCreateRec(>[PERSONNES];->[PERSONNES]Nom;"Julie Andrews")
```

Dans le composant, le code de la méthode `methCreateRec` :

```

C_POINTER($1) //Pointeur vers une table du projet hôte
C_POINTER($2) //Pointeur vers un champ du projet hôte
C_TEXT($3) // Valeur à insérer

$tablepointer:=$1
$fieldpointer:=$2
CREATE RECORD($tablepointer->

$fieldpointer->:=$3
SAVE RECORD($tablepointer->

```

Dans le contexte d'un composant, 4D suppose qu'une référence à un formulaire table est une référence au formulaire table hôte (car les composants ne peuvent pas avoir de tables)

Utilisation de tables et de champs

Un composant ne peut pas utiliser les tables et les champs définis dans la structure 4D du projet utilisé comme matrice. En revanche, il peut créer et utiliser des bases externes, et donc utiliser des tables et des champs en fonction de ses besoins. Les bases externes sont créées et gérées via le langage SQL. En revanche, il peut créer et utiliser des bases externes, et donc utiliser des tables et des champs en fonction de ses besoins. Utiliser une base externe signifie désigner temporairement cette base comme base courante, c'est-à-dire comme base cible des requêtes SQL exécutées par 4D. Les bases externes sont créées à l'aide de la commande SQL `CREATE DATABASE`.

Exemple

Le code suivant est inclus dans un composant et effectue trois actions élémentaires avec une base de données externe :

- création de la base de données externe si elle n'existe pas déjà,
- ajout de données dans la base de données externe,
- lecture de données depuis la base de données externe.

Création de la base de données externe :

```
<>MyDatabase:=Get 4D folder+"\MyDB" // (Windows) stocke les données dans un répertoire autorisé
Begin SQL
    CREATE DATABASE IF NOT EXISTS DATAFILE :[<>MyDatabase];
    USE DATABASE DATAFILE :[<>MyDatabase];
    CREATE TABLE IF NOT EXISTS KEEPIT
    (
        ID INT32 PRIMARY KEY,
        kind VARCHAR,
        name VARCHAR,
        code TEXT,
        sort_order INT32
    );

    CREATE UNIQUE INDEX id_index ON KEEPIT (ID);

    USE DATABASE SQL_INTERNAL;

End SQL
```

Ecriture dans la base de données externe :

```
$Ptr_1:=$2 // récupère les données à partir du projet hôte via des pointeurs
$Ptr_2:=$3
$Ptr_3:=$4
$Ptr_4:=$5
$Ptr_5:=$6
Begin SQL

    USE DATABASE DATAFILE :[<>MyDatabase];

    INSERT INTO KEEPIT
    (ID, kind, name, code, sort_order)
    VALUES
    (:[$Ptr_1], :[$Ptr_2], :[$Ptr_3], :[$Ptr_4], :[$Ptr_5]);

    USE DATABASE SQL_INTERNAL;

End SQL
```

Lecture dans une base de données externe :

```
$Ptr_1:=$2 // accède aux données du projet hôte via des pointeurs  
$Ptr_2:=$3  
$Ptr_3:=$4  
$Ptr_4:=$5  
$Ptr_5:=$6
```

Begin SQL

```
USE DATABASE DATAFILE :[<>MyDatabase];  
  
SELECT ALL ID, kind, name, code, sort_order  
FROM KEEPIT  
INTO :$Ptr_1, :$Ptr_2, :$Ptr_3, :$Ptr_4, :$Ptr_5;  
  
USE DATABASE SQL_INTERNAL;
```

End SQL

Utilisation de formulaires

- Seuls les "formulaires projet" (formulaires non associés à une table en particulier) peuvent être exploités directement dans un composant. Seuls les "formulaires projet" (formulaires non associés à une table en particulier) peuvent être exploités directement dans un composant.
- Un composant peut faire appel à des formulaires table du projet hôte. A noter qu'il est nécessaire dans ce cas d'utiliser des pointeurs plutôt que des noms de table entre [] pour désigner les formulaires dans le code du composant.

Si un composant utilise la commande `ADD RECORD`, le formulaire Entrée courant du projet hôte sera affiché, dans le contexte du projet hôte. Par conséquent, si le formulaire comporte des variables, le composant n'y aura pas accès.

- Vous pouvez publier des formulaires de composants comme sous-formulaires dans les projets hôtes. Avec ce principe, vous pouvez notamment développer des composants proposant des objets graphiques. Par exemple, les Widgets proposés par 4D sont basés sur l'emploi de sous-formulaires en composants.

Dans le contexte d'un composant, tout formulaire projet référencé doit appartenir au composant. Par exemple, à l'intérieur d'un composant, le fait de référencer un formulaire projet hôte à l'aide de `DIALOG` ou de `Open form window` déclenchera une erreur.

Utilisation de ressources

Les composants peuvent utiliser des ressources situées dans le dossier Ressources du composant.

Les mécanismes automatiques sont opérationnels : les fichiers XLIFF présents dans le dossier Resources d'un composant seront chargés par ce composant.

Dans un projet hôte contenant un ou plusieurs composants, chaque composant ainsi que les projets hôtes ont leur propre «chaîne de ressources». Dans un projet hôte contenant un ou plusieurs composants, chaque composant ainsi que les projets hôtes ont leur propre «chaîne de ressources». Les ressources sont divisées entre les différents projets : il n'est pas possible d'accéder aux ressources du composant A depuis le composant B ou depuis le projet hôte.

Exécution du code d'initialisation

Un composant peut exécuter automatiquement du code 4D lors de l'ouverture ou de la fermeture de la base hôte, par exemple pour charger et/ou sauvegarder les préférences ou les états utilisateur liés au fonctionnement de la base hôte.

L'exécution du code d'initialisation ou de fermeture se fait au moyen de la méthode base `On Host Database Event`.

Pour des raisons de sécurité, vous devez autoriser explicitement l'exécution de la méthode base `On Host Database Event` dans la base hôte afin de pouvoir l'appeler. Pour des raisons de sécurité, vous devez autoriser explicitement l'exécution de la méthode base `On Host Database Event` dans la base hôte afin de pouvoir l'appeler.

Protection des composants : la compilation

Par défaut, tout le code d'un projet utilisé comme matrice installé comme composant est virtuellement visible depuis le projet hôte. En particulier :

- Les méthodes projet partagées sont accessibles dans la Page Méthodes de l'Explorateur et peuvent être appelées dans les méthodes du projet hôte. Leur contenu peut être sélectionné et copié dans la zone de prévisualisation de l'Explorateur. Elles peuvent également être visualisées dans le débogueur. Il n'est toutefois pas possible de les ouvrir dans l'éditeur de méthodes ni de les modifier.
- Les autres méthodes projet du projet utilisé comme matrice n'apparaissent pas dans l'Explorateur mais peuvent également être visualisées dans le débogueur du projet hôte.
- Les classes et fonctions non cachées peuvent être visualisées dans le débogueur [si un namespace est déclaré](#).

Pour assurer la protection du code d'un composant, [compilez et générerez](#) simplement le projet utilisé comme matrice et fournissez-le sous forme de fichier .4dz. Lorsqu'un projet compilé utilisé comme matrice est installé comme composant :

- Les méthodes, classes et fonctions du projet partagé peuvent être appelées dans les méthodes du projet hôte. Les méthodes du projet partagé sont également visibles sur la page Méthodes de l'Explorateur. En revanche, leur contenu n'apparaît pas dans la zone de prévisualisation ni dans le débogueur.
- Les autres méthodes projet du projet utilisé comme matrice n'apparaissent jamais.

Partage des composants

We encourage you to support the 4D developer community by sharing your components, preferably on the [GitHub platform](#). We recommend that you use the `4d-component` topic to be correctly referenced.

Plug-ins de développement

Pourquoi un plugin ?

Bien que 4D propose des centaines de méthodes intégrées permettant de manipuler des objets et des enregistrements, et d'implémenter une interface utilisateur, une utilisation ou des fonctionnalités spéciales (parfois dépendantes de la plate-forme) peuvent être nécessaires : l'une peut nécessiter ODBC sous Windows, une autre peut nécessiter des services Apple sous macOS, et un autre peut souhaiter implémenter des outils statistiques spécifiques, une connexion à un réseau social, une plateforme de paiement, un accès à un fichier sur le réseau, une interface utilisateur spéciale ou une structure d'image privée.

Il est évident que couvrir tous les domaines des systèmes d'exploitation macOS et Windows au moyen de commandes 4D mènerait certainement à un produit contenant des milliers de commandes et, dans le même temps, la plupart des utilisateurs n'auraient pas besoin d'un si grand ensemble de fonctionnalités. De plus, la création d'un outil aussi complet rendrait l'environnement 4D incroyablement complexe et demanderait des mois d'étude à la plupart des utilisateurs avant de pouvoir obtenir des résultats utiles.

La nature modulaire de l'environnement 4D permet la création d'applications de base, mais n'exclut pas le développement de systèmes extrêmement complexes. L'architecture du plug-in 4D ouvre l'environnement 4D à tout type d'application ou d'utilisateur. Les plug-ins 4D multiplient la puissance et la productivité de cette application ou de l'utilisateur.

Qu'est-ce qu'un plug-in et à quoi sert-il ?

Un plug-in est un morceau de code que 4D lance au démarrage. Il ajoute des fonctionnalités à 4D et augmente ainsi sa capacité.

Habituellement, un plug-in fait des choses :

- Que 4D ne peut pas effectuer (c'est-à-dire une technologie de plate-forme spécifique),
- Qui sera très difficile à écrire en utilisant uniquement 4D,
- Qui sont uniquement disponibles en tant que point d'entrée de plug-in

Un plug-in contient généralement un ensemble de routines données au développeur 4D. Il peut gérer une zone externe et exécuter un processus externe.

- Une routine de plug-in est une routine écrite en langage natif (généralement C ou C++) qui déclenche une action.
- Une zone externe est une partie d'un formulaire pouvant presque tout afficher et interagir avec l'utilisateur si nécessaire.
- Un processus externe est un processus qui s'exécute seul, généralement en boucle, et qui fait quasiment tout ce qu'il souhaite. Tout le code de process appartient au plug-in, 4D est simplement présent pour recevoir/envoyer des événements au process.

Note importante

Un plug-in peut être très simple, avec une seule routine effectuant une très petite tâche, ou très complexe, impliquant une centaine de routines et de domaines. Cependant, chaque développeur de plug-in doit se rappeler qu'un plug-in est un "échantillon" de code. C'est le plug-in qui s'exécute dans 4D, et non l'inverse. En tant que morceau de code, c'est l'hôte de 4D; ce n'est pas une application autonome. Il partage le temps CPU et la mémoire avec 4D et d'autres plug-ins. Il doit donc s'agir d'un code poli, qui utilise exactement ce qui est nécessaire à son fonctionnement. Par exemple, dans les longues boucles, un plug-in doit appeler `PA_Yield()` pour donner du temps au planificateur 4D, à moins que sa tâche ne soit critique aussi bien pour lui que pour l'application.

Comment créer un plug-in ?

Sur GitHub, 4D fournit un [plug-in SDK](#) open source, contenant le plug-in API 4D et l'assistant de plug-in 4D :

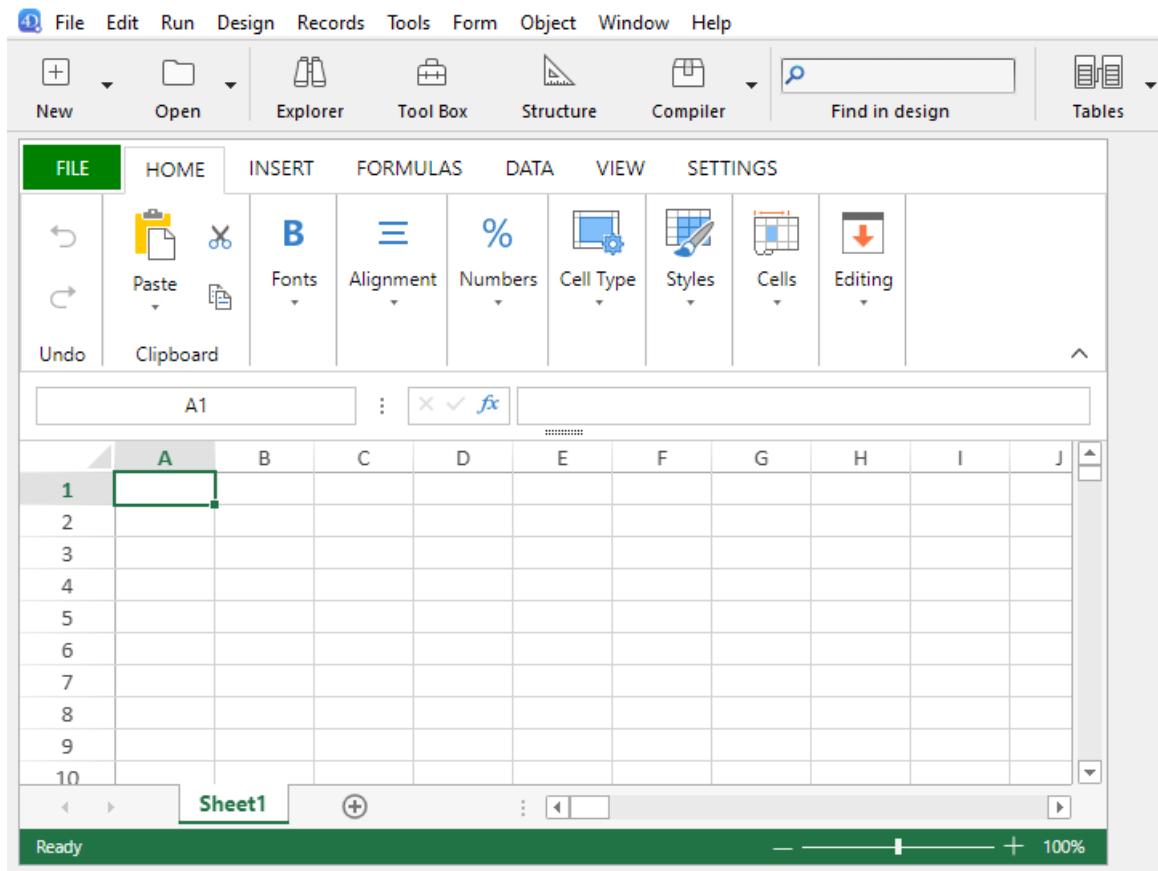
- le [Plugin API de 4D](#), écrit en C, ajoute plus de 400 fonctions qui vous aident à créer facilement vos propres plug-ins pour ajouter de nouvelles fonctionnalités à votre application 4D. Les fonctions du plug-in API de 4D gèrent toutes les interactions entre l'application 4D et votre plug-in.
- [L'assistant de plug-in 4D](#) est un outil essentiel qui simplifie la tâche de développement des plug-ins 4D. Il écrit le code dont 4D a besoin pour interagir correctement avec un plug-in et le charger, afin de vous concentrer sur votre propre code.

Partager des plug-ins

Nous vous encourageons à soutenir la communauté des développeurs 4D en partageant vos plug-ins, notamment sur la [plateforme GitHub](#). Afin d'être correctement référencé, nous vous recommandons d'utiliser la rubrique [4d-plugin](#).

Prise en main

4D View Pro is a [4D component](#) that includes a [4D form area](#) and specific [methods](#). Il vous permet d'intégrer des fonctionnalités avancées de tableau dans vos projets.



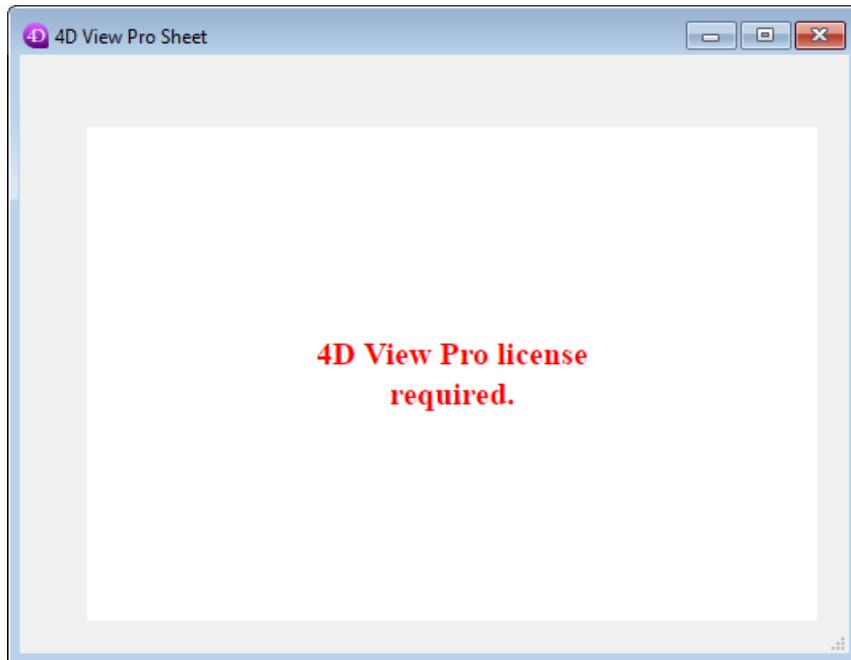
Une tableur est une application contenant une grille de cellules dans lesquelles vous pouvez saisir des informations, effectuer des calculs ou afficher des images. 4D View Pro is powered by the [SpreadJS spreadsheet solution](#) integrated in 4D.

L'intégration de zones 4D View Pro dans vos formulaires vous permet d'importer et d'exporter des documents de type tableur à l'aide des commandes 4D View Pro.

Installation et activation

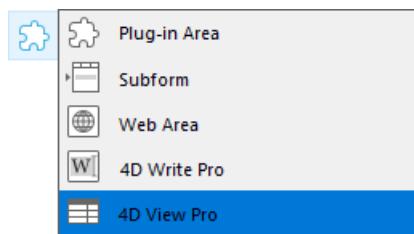
Les fonctionnalités de 4D View Pro sont directement incluses dans 4D, ce qui en facilite le déploiement et la gestion. Aucune installation supplémentaire n'est nécessaire.

Cependant, 4D View Pro nécessite une licence. Vous devez activer cette licence dans votre application afin d'utiliser ses fonctionnalités. Lorsque vous utilisez ce composant sans licence, le contenu d'un objet nécessitant une fonctionnalité 4D View Pro ne s'affiche pas au moment de l'exécution; au lieu de cela, un message d'erreur :



Insertion d'une zone 4D View Pro

Les documents 4D View Pro sont affichés et modifiés manuellement dans un [objet du formulaire 4D](#) nommé 4D View Pro. Pour sélectionner cet objet, cliquez sur le dernier outil de la barre d'objets :



Vous pouvez également sélectionner une zone 4D View Pro préconfigurée dans la [bibliothèque d'objets](#).

Les zones 4D View Pro peuvent également être [crées et utilisées hors écran](#).

Vous pouvez [configurer la zone](#) à l'aide de la liste de propriétés et des méthodes 4D View Pro.

Fondamentaux de la sélection, de la saisie et de la navigation

Les feuilles de calcul sont composées de lignes et de colonnes. Un numéro est associé à chaque ligne. Une lettre (ou un groupe de lettres lorsque le nombre de colonnes est supérieur au nombre de lettres de l'alphabet) est associée à chaque colonne. L'intersection d'une ligne et d'une colonne constitue une cellule. Les cellules peuvent être sélectionnées et leur contenu modifié.

Sélection des cellules, des colonnes et des lignes

- Pour sélectionner une cellule, il suffit de cliquer dessus ou d'utiliser les flèches de direction du clavier. Son contenu (ou sa formule) s'affiche dans la cellule.
- Pour sélectionner plusieurs cellules continues, faites glisser la souris d'une extrémité à l'autre de la sélection. Vous pouvez également cliquer sur les deux extrémités de la sélection tout en maintenant la touche Maj enfoncée.
- Pour sélectionner toutes les cellules de la feuille de calcul, cliquez sur la cellule en haut à gauche de la zone :



- Pour sélectionner une colonne, cliquez sur la lettre (ou le groupe de lettres) correspondant(e).
- Pour sélectionner une ligne, cliquez sur le chiffre correspondant.

- Pour sélectionner un groupe de cellules non continues, maintenez la touche Ctrl (Windows) ou Command (macOS) enfoncée et cliquez sur chaque cellule à sélectionner.
- Pour désélectionner des cellules, il suffit de cliquer n'importe où dans la feuille de calcul.

Saisie de données

Un double-clic sur une cellule permet de passer en mode saisie dans la cellule concernée. Si la cellule n'est pas vide, le curseur d'insertion se place après le contenu de la cellule.



Les données peuvent être saisies directement lorsqu'une cellule est déjà sélectionnée, même si le curseur d'insertion n'est pas visible. La saisie remplace alors le contenu de la cellule.

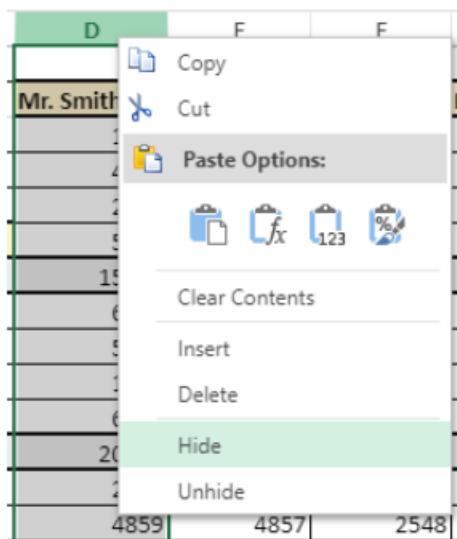
La touche Tab valide la saisie de la cellule et sélectionne la cellule à sa droite. La combinaison des touches Shift + Tab valide la saisie de la cellule et sélectionne la cellule située à sa gauche.

La touche Retour chariot valide la saisie de la cellule et sélectionne la cellule située en dessous. La combinaison des touches Shift + Retour chariot valide la saisie de la cellule et sélectionne la cellule située au-dessus.

Les touches de direction (flèches) permettent de déplacer une cellule dans la direction indiquée par la flèche.

Utilisation du menu contextuel

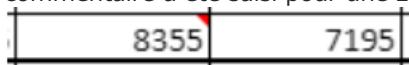
Les zones 4D View Pro bénéficient d'un menu contextuel automatique qui offre des fonctions d'édition standard telles que le copier-coller, mais aussi des fonctions de base de tableau :



Les fonctions Copier/Couper et Coller du menu contextuel fonctionnent uniquement dans la zone de la feuille de calcul, elles n'ont pas accès au presse-papiers du système. Les raccourcis système tels que Ctrl+c/Ctrl+v fonctionnent néanmoins et peuvent être utilisés pour échanger des données entre la zone et d'autres applications.

En fonction de la zone cliquée, les options suivantes sont également disponibles :

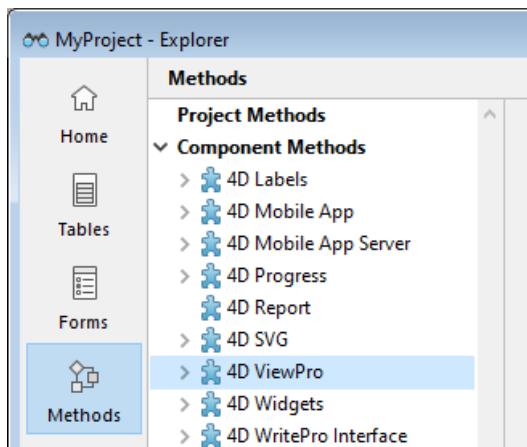
- cliquez sur un en-tête de colonne ou de ligne : Insérer, Supprimer, Masquer ou Décompresser le contenu
- cliquez sur une cellule ou une plage de cellules :
 - Filter: allows hiding row through filters (see [Filtering rows](#) in the SpreadJS documentation).
 - Trier : permet de trier le contenu des colonnes.
 - Insérer un commentaire : permet à l'utilisateur de saisir un commentaire pour une zone. Lorsqu'un commentaire a été saisi pour une zone, la cellule supérieure gauche de la zone affiche un petit triangle rouge :



Utilisation des méthodes 4D View Pro

Les méthodes 4D View Pro peuvent être utilisées dans l'éditeur de méthodes 4D, tout comme les commandes du langage 4D.

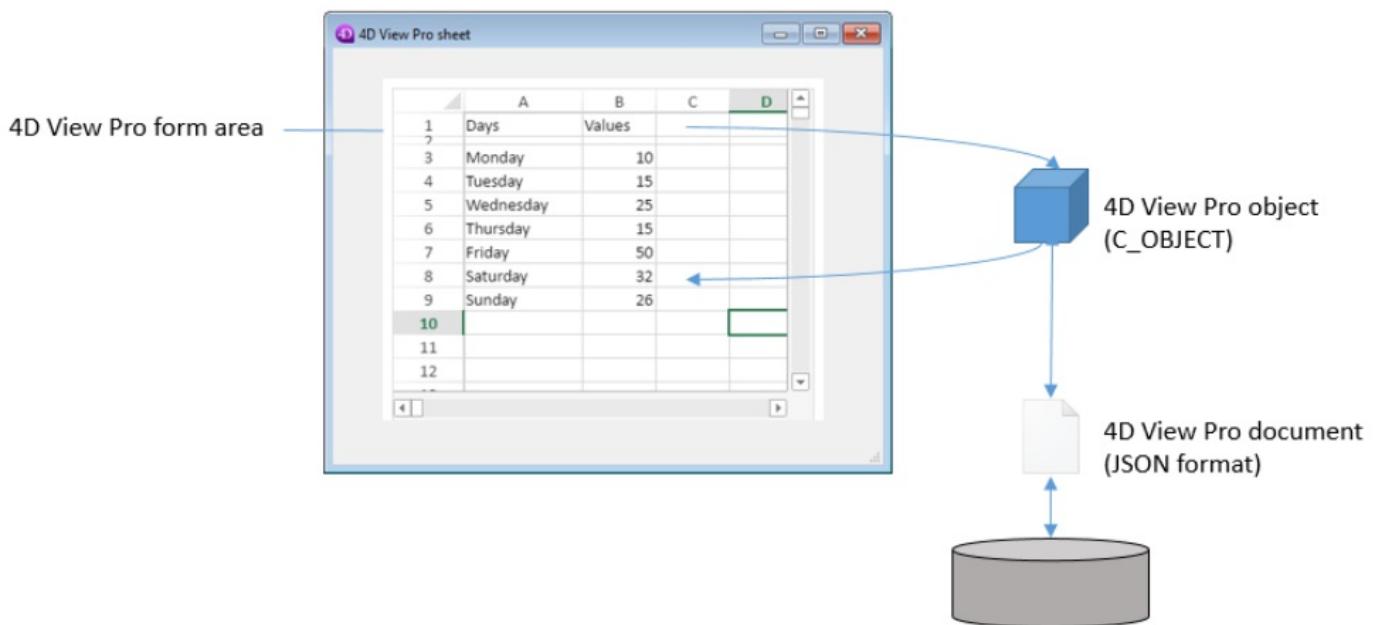
4D View Pro étant un composant 4D intégré, vous pouvez accéder à sa liste de méthodes à partir de l'Explorateur, dans la section Méthodes composants :



Pour une liste détaillée des méthodes composants, voir [Liste des méthodes](#).

Traiter une zone 4D View Pro

Une zone 4D View Pro gère plusieurs objets et éléments.



La plupart des méthodes 4D View Pro nécessitent un paramètre `vpAreaName`, qui est le [nom de la zone de formulaire 4D View Pro](#) (objet de formulaire 4D). Ce nom est la propriété du [nom de l'objet](#).

Par exemple, si vous souhaitez définir le nombre total de colonnes d'une zone nommée "myVpArea", saisissez le code suivant :

```
VP SET COLUMN COUNT("myVpArea";5)
```

Lorsque vous chargez un objet 4D View Pro dans une zone de formulaire, 4D génère l'événement formulaire [On VP Ready](#) une fois que toute la zone est chargée. Vous devez exécuter tout code 4D View Pro manipulant la zone dans cet événement, sinon une erreur est retournée.

Utilisation d'objets de plage

Certaines méthodes 4D View Pro nécessitent un paramètre *rangeObj*. Dans 4D View Pro, une plage est un objet qui fait référence à une zone dans une feuille de calcul. Cette zone peut être composée d'une ou plusieurs cellules. À l'aide des méthodes 4D View Pro, vous pouvez créer des plages et les passer à d'autres méthodes pour lire ou écrire dans des emplacements spécifiques de votre document.

Par exemple, pour créer un objet plage pour les cellules suivantes :

	A	B	C	D	E	F
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						

Vous pouvez utiliser la méthode [VP Cells](#) :

```
var $myRange : Object  
$myRange:=VP Cells("ViewProArea";2;4;2;3) // C5 to D7
```

Vous pouvez ensuite passer `$myRange` à une autre méthode 4D View Pro pour modifier ces cellules (par exemple ajouter une bordure à l'ensemble des cellules avec [VP SET BORDER](#)).

Les objets plage 4D View Pro sont composés de plusieurs propriétés :

- area - Le nom de la zone 4D View Pro
- ranges - Une collection d'objet(s) plage. Les propriétés disponibles dans chaque objet de gamme dépendent du type d'objet plage. Par exemple, un objet plage de type colonne comprendra uniquement les propriétés `.column` et `.sheet`.

Propriété		Type	Description	Disponible pour
area		Texte	Nom d'objet formulaire zone 4D View Pro	toujours disponible
ranges		collection	Collection de plage(s)	toujours disponible
	[].name	Texte	Nom de plage	name
	[].sheet	number	Indice de la feuille (par défaut, indice de la feuille courante) (le comptage démarre à 0)	cellule, cellules, ligne, lignes, colonne, colonnes, tout, nom
	[].row	number	Indice de la ligne (le comptage démarre à 0)	cellule, cellules, ligne, lignes
	[].rowCount	number	Row count	cellule, lignes
	[].column	number	Indice de la colonne (le comptage démarre à 0)	cellule, cellules, colonne, colonnes
	[].columnCount	number	Column count	cellules, colonnes

Import et export de documents

4D View Pro prend en charge l'import et l'export de plusieurs formats de documents :

- `.4vp`
- `.xlsx`
- `.txt` et `.csv`

- .pdf (à des fins d'export uniquement)

Pour plus de détails, consultez la description de [VP IMPORT DOCUMENT](#) et [VP EXPORT DOCUMENT](#).

Configuration des zones 4D View Pro

Les propriétés de la zone 4D View Pro peuvent être configurées à l'aide de la liste des propriétés. Les propriétés des feuilles de calcul sont disponibles à l'aide du langage.

Propriétés de la zone de formulaire

À l'aide de la liste de propriétés de la zone, vous pouvez configurer les [propriétés objet 4D View Pro](#) telles que Nom de l'objet, [Variable ou Expression](#), Apparence, Action, et Evénements.

Sélection d'une interface utilisateur

Vous pouvez sélectionner l'interface à utiliser avec vos zones de formulaire 4D View Pro dans la [Liste de propriétés](#), dans [Apparence](#) :



Vous pouvez également utiliser les propriétés JSON `userInterface` et `withFormulaBar` (uniquement avec l'interface "toolbar").

Les interfaces permettent d'effectuer des modifications de base et de manipuler des données. Les modifications définies par l'utilisateur sont enregistrées dans l'objet 4D View Pro lorsque l'utilisateur enregistre le document.

Ruban

Barre d'outils

Lorsque l'interface Barre d'outils est activée, l'option [Afficher barre de formule](#) s'affiche. Si la case de l'option est cochée, la barre de formule est visible en-dessous de l'interface Barre d'outils.

Lorsque la barre de formule est visible :

Features

Les interfaces de Ruban et de Barre d'outils regroupent les fonctionnalités qui s'y rattachent dans des onglets :

Onglet	Actions	Interface Ruban	Interface Barre d'outils
File	Gestion de fichiers	X	
Accueil	Apparence du texte	X	X
Insérer	Ajouter des éléments	X	X
Formules	Calculs de formules et bibliothèque	X	X
Données	Gestion des données	X	X
Affichage	Présentation visuelle	X	X
Settings	Présentation de la feuille	X	

Événements formulaire

Les événements formulaire suivants sont disponibles dans la Liste de propriétés des zones 4D View Pro.

Certains de ces événements sont des événements formulaire standard (disponibles pour tous les objets actifs) et d'autres sont des événements formulaire spécifiques à 4D View Pro. Certains événements formulaire standard fournissent des informations étendues dans l'objet retourné par la commande [FORM Event](#) lorsqu'ils sont générés pour les zones 4D View Pro. Le tableau suivant indique les événements standard et les événements spécifiques ou ceux qui fournissent des informations supplémentaires aux zones 4D View Pro :

Événements 4D standard	Événements 4D View Pro spécifiques et étendus
On Load	On VP Ready
On Getting Focus	Sur clic
On Losing Focus	Sur double clic
On Unload	Sur clic entête
	Sur après modification
	Sur nouvelle sélection
	Sur redimensionnement colonne
	On Row Resize
	On VP Range Changed

Options feuille

L'objet options feuille 4D View Pro vous permet de contrôler plusieurs options pour vos zones 4D View Pro. Cet objet est géré par les commandes suivantes :

- [VP SET SHEET OPTIONS](#)
- [VP Get sheet options](#)

Apparence de la feuille

Propriété		Type	Description
allowCellOverflow		boolean	Indique si les données peuvent déborder sur des cellules vides adjacentes.
sheetTabColor		string	Une chaîne couleur utilisée pour représenter la couleur de l'onglet de la feuille, telle que "red", "#FFFF00", "rgb(255,0,0)", "Accent 5", etc.
frozenlineColor		string	Une chaîne couleur utilisée pour représenter la couleur de ligne figée, telle que "red", "#FFFF00", "rgb(255,0,0)", "Accent 5", etc.
clipBoardOptions		entier long	L'option presse-papiers. Valeurs disponibles : <code>vk clipboard paste options all</code> , <code>vk clipboard paste options formatting</code> , <code>vk clipboard paste options formulas</code> , <code>vk clipboard paste options formulas and formatting</code> , <code>vk clipboard paste options values</code> , <code>vk clipboard paste options values and formatting</code>
gridline		object	Les options du quadrillage.
	color	string	Une chaîne couleur utilisée pour représenter la couleur du quadrillage, telle que "red", "#FFFF00", "rgb(255,0,0)", "Accent 5", etc.
	showVerticalGridline	boolean	Indique s'il affiche le quadrillage vertical.
	showHorizontalGridline	boolean	Indique s'il affiche le quadrillage horizontal.
rowHeaderVisible		boolean	Indique si l'en-tête de la ligne est visible.
colHeaderVisible		boolean	Indique si l'en-tête de la colonne est visible.
rowHeaderAutoText		entier long	Indique si l'en-tête de la ligne affiche des lettres ou des chiffres ou s'il n'affiche rien. Valeurs disponibles : <code>vk header auto text blank</code> , <code>vk header auto text letters</code> , <code>vk header auto text numbers</code>
colHeaderAutoText		entier long	Indique si l'en-tête de la colonne affiche des lettres ou des chiffres ou s'il n'affiche rien. Valeurs disponibles : <code>vk header auto text blank</code> , <code>vk header auto text letters</code> , <code>vk header auto text numbers</code>
selectionBackColor		string	La couleur d'arrière-plan de la sélection dans la feuille (de préférence au format Rgba).
selectionBorderColor		string	La couleur de bordure de la sélection dans la feuille.
sheetAreaOffset		object	Les options de sheetAreaOffset.
	left	entier long	Le décalage gauche de la feuille à partir de l'hôte.
	top	entier long	Le décalage supérieur de la feuille à partir de l'hôte.

Toutes les propriétés sont optionnelles.

Protection de la feuille

Pour verrouiller l'intégralité de la feuille, il suffit de mettre la propriété `isProtected` à vrai. Vous pouvez alors déverrouiller les cellules individuellement en définissant la propriété de style de la cellule [verrouillée](#).

Propriété		Type	Description
isProtected		boolean	Indique si les cellules protégées de cette feuille peuvent être modifiées ou non.
protectionOptions		object	Une valeur qui indique les éléments que les utilisateurs peuvent modifier. Si null : le paramètre protectionOptions est réinitialisé.
	allowSelectLockedCells	boolean	Indique si l'utilisateur peut sélectionner des cellules verrouillées, optionnel. Vrai par défaut.
	allowSelectUnlockedCells	boolean	Indique si l'utilisateur peut sélectionner des cellules non verrouillées, optionnel. Vrai par défaut.
	allowSort	boolean	Indique si l'utilisateur peut trier des plages, optionnel. Faux par défaut.
	allowFilter	boolean	Indique si l'utilisateur peut filtrer des plages, optionnel. Faux par défaut.
	allowEditObjects	boolean	Indique si l'utilisateur peut modifier des objets flottants, optionnel. Faux par défaut.
	allowResizeRows	boolean	Indique si l'utilisateur peut redimensionner des lignes, optionnel. Faux par défaut.
	allowResizeColumns	boolean	Indique si l'utilisateur peut redimensionner des colonnes, optionnel. Faux par défaut.
	boolean	boolean	Indique si l'utilisateur peut effectuer des opérations de glissement pour insérer des lignes, optionnel. Faux par défaut.
	allowDragInsertColumns	boolean	Indique si l'utilisateur peut effectuer des opérations de glissement pour insérer des colonnes, optionnel. Faux par défaut.
	allowInsertRows	boolean	Indique si l'utilisateur peut insérer des lignes, optionnel. Faux par défaut.
	allowInsertColumns	boolean	Indique si l'utilisateur peut insérer des colonnes, optionnel. Faux par défaut.
	allowDeleteRows	boolean	Indique si l'utilisateur peut supprimer des lignes, optionnel. Faux par défaut.
	allowDeleteColumns	boolean	Indique si l'utilisateur peut supprimer des colonnes, optionnel. Faux par défaut.

Toutes les propriétés sont optionnelles.

Format des cellules

La définition d'un modèle de format garantit que le contenu de vos documents 4D View Pro s'affiche comme souhaité. Les formats peuvent être définis à l'aide de l'[interface](#) 4D View Pro sélectionnée, ou à l'aide des méthodes [VP SET VALUE](#) ou [VP SET NUM VALUE](#).

4D View Pro dispose de formats intégrés pour les chiffres, les dates, les heures et le texte, mais il vous est possible de créer vos propres modèles pour formater le contenu des cellules à l'aide de caractères et de codes spéciaux.

Par exemple, lorsque vous utilisez les méthodes [VP SET VALUE](#) ou [VP SET NUM VALUE](#) pour saisir des montants dans une facture, vous souhaiteriez que les symboles monétaires (\$, €, ¥, etc.) soient alignés, quel que soit l'espace requis par le nombre (c'est-à-dire que le montant soit de 5,00 \$ ou de 5 000,00 \$). Vous pourriez utiliser des caractères de formatage et indiquer le motif `(#$#,##0.00)` qui afficherait les montants comme indiqué :

\$	4,180.00
\$	15.00
\$	200.00
\$	15,600.00
\$	1,672.00

A noter que lorsque vous créez vos propres modèles, seul l'affichage des données est modifié. La valeur des données reste inchangée.

Format texte et format numérique

Les formats numériques s'appliquent à tous les types de chiffres (ex : nombres positifs, négatifs, les zéros).

Caractère	Description	Exemple
0	Placeholder that displays zeros.	#.00 affichera 1.10 au lieu de 1.1
.	Displays a decimal point	0.00 affichera 1999.00 au lieu de 1999
,	Displays the thousands separator in a number. Thousands are separated by commas if the format contains a comma enclosed by number signs "#" or by zeros. A comma following a digit placeholder scales the number by 1,000.	#,0 affichera 12200000 au lieu de 12,200,000
_	Skips the width of the next character.	Usually used in combination with parentheses to add left and right indents, _(and _) respectively.
@	Formatter for text. Applies the format to all text in the cell	"[Red]@" applies the red font color for text values.
*	Repeats the next character to fill the column width.	0*- will include enough dashes after a number to fill the cell, whereas *0 before any format will include leading zeros.
" "	Displays the text within the quotes without interpreting it.	"8%" sera affiché comme suit : 8%
%	Displays numbers as a percentage of 100.	8% sera affiché comme suit : .08
#	Digit placeholder that does not display extra zeros. If a number has more digits to the right of the decimal than there are placeholders, the number is rounded up.	#.# affichera 1.5 au lieu de 1.54
?	Digit placeholder that leaves space for extra zeros, but does not display them. Typically used to align numbers by decimal point.	\$?? displays a maximum of 2 decimals and causes dollar signs to line up for varying amounts.
¥	Displays the character following it.	#.00? affichera 123.00? au lieu de 123
/	When used with numbers, displays them as fractions. When used with text, date or time codes, displayed "as-is".	#/# affichera 3/4 au lieu de .75
[]	Creates conditional formats.	[>100][GREEN]#,##0;[<=-100][YELLOW]#,##0;[BLUE]#,##0
E	Scientific notation format.	#E+# - affichera 2E+6 au lieu de 1,500,500
[color]	Formats the text or number in the color specified	[Green]####.[Red]-####.###

Exemple

```
//Définir la valeur de la cellule sur $125,571.35
VP SET VALUE(VP Cell("ViewProArea";3;2);New object("value";125571.35;"format";"$_(* #,##0.00_)")
```

Formats date et heure

4D View Pro fournit les constantes suivantes pour les modèles de date et heure au format ISO 8601 :

Constante	Valeur	Commentaire
vk pattern full date time	"fullDateTimePattern"	ISO 8601 format for the full date and time in current localization. USA default pattern: "ddd, dd MMMM yyyy HH:mm:ss"
vk pattern long date	"longDatePattern"	ISO 8601 format for the full date in current localization. USA default pattern: "ddd, dd MMMM yyyy"
vk pattern long time	"longTimePattern"	ISO 8601 format for the time in current localization. USA default pattern: "HH:mm:ss"
vk pattern month day	"monthDayPattern"	ISO 8601 format for the month and day in current localization. USA default pattern: "MMM dd"
vk pattern short date	"shortDatePattern"	Abbreviated ISO 8601 format for the date in current localization. USA default pattern: "MM/dd/yyyy"
vk pattern short time	"shortTimePattern"	Abbreviated ISO 8601 format for the time in current localization. USA default pattern: "HH:mm"
vk pattern sortable date time	"sortableDateTimePattern"	ISO 8601 format for the date and time in current localization which can be sorted. USA default pattern: "yyyy-MM-dd'T'HH:mm:ss"
vk pattern universal sortable date time	"universalSortableDateTimePattern"	ISO 8601 format for the date and time in current localization using UTC which can be sorted. USA default pattern: "yyyy-MM-dd HH:mm:ss'Z'"
vk pattern year month	"yearMonthPattern"	ISO 8601 format for the month and year in current localization. USA default pattern: "yyyy MMMM"

Exemple

```
//Définir la valeur de la cellule sur une date et une heure spécifique
VP SET VALUE(VP Cell("ViewProArea";3;9);New object("value";!2024-12-18!);"time";?14:30:10?;"format";vk p
```

Formats date et heure personnalisés

To create your own date and time patterns, in your current localization, you can use combinations of the following codes:

	Code (non sensible à la casse)	Description	Exemple
Date			(January 1, 2019)
	m	Numéro du mois sans le premier zéro	1
	mm	Numéro du mois avec le zéro en préfixe	01
	mmm	Nom du mois, court	Jan
	mmmm	Nom du mois, long	January
	d	Numéro du jour sans le zéro en préfixe	1
	dd	Numéro du jour avec le zéro en préfixe	01
	ddd	Jour de la semaine, court	Tue
	ddd	Jour de la semaine, long	Tuesday
	yy	Année, format court	19
	yyyy	Année, long format	2019
Heure			(2:03:05 PM)
	h	Heure sans le zéro en préfixe. 0-23	2
	hh	Heure avec le zéro en préfixe. 00-23	02
	m	Minutes sans le zéro en préfixe. 0-59	3
	mm	Minutes avec le zéro en préfixe. 00-59	03
	s	Secondes sans le zéro en préfixe. 0-59	5
	ss	Secondes avec le zéro en préfixe. 00-59	05
	[h]	Temps écoulé en heures	14 (peut aller au delà de 24)
	[mm]	Temps écoulé en minutes	843
	[ss]	Temps écoulé en secondes	50585
	AM/PM	Périodes de la journée. S'il est omis, c'est le format de 24 heures qui est utilisé.	PM

The code 'm' is interpreted depending on its position in the pattern. If it's immediately after 'h' or 'hh' or immediately before 's' or 'ss', it will be interpreted as minutes, otherwise it will be interpreted as months.

Symboles supplémentaires

In addition to the special characters and codes described in the previous sections, there are additional characters and symbols that can be used in your format patterns. These additional characters and symbols do not require a \$ or "" and do not impact the interpretation of the format pattern. They appear "as-is" within the pattern.

Caractère	Description	Exemple
+ et -	Plus and minus signs	### + ### = ###,###
()	Left and right parenthesis	(-###.##)
:	Colon	hh:mm:ss
^	Caret	#^#
'	Apostrophe	'#####
{ }	Curly brackets	{###,###,###}
< >	Less-than and greater than signs	## >##
=	Equal sign	#+#=##
/	Forward slash. When used with numbers, displays them as fractions.	mm/dd/yyyy
!	Exclamation point	\$###.00!
&	Ampersand	"Hello" & "Welcome"
~	Tilde	~##
	Space character	
€	Euro	€###.00
£	British Pound	£###.00
¥	Japanese Yen	¥###.00
\$	Dollar sign	\$###.00
¢	Cent sign	.00¢

Attributs d'impression

4D View Pro print attributes allow you to control all aspects of printing 4D View Pro areas. These attributes are handled by the following commands:

- [VP SET PRINT INFO](#)
- [VP Get print info](#)

Colonnes / Lignes

Les attributs de ligne et de colonne sont utilisés pour identifier le début, la fin et la répétition des lignes et colonnes.

Propriété	Type	Description
columnEnd	entier long	The last column to print in a cell range. Default value = -1 (all columns)
columnStart	entier long	The first column to print in a cell range. Default value = -1 (all columns)
repeatColumnEnd	entier long	The last column of a range of columns to print on the left of each page. Default value = -1 (all columns)
repeatColumnStart	entier long	The first column of a range of columns to print on the left of each page. Default value = -1 (all columns)
repeatRowEnd	entier long	The last row of a range of rows to print on the top of each page. Default value = -1 (all rows)
repeatRowStart	entier long	The first row of a range of rows to print at the top of each page. Default value = -1 (all rows)
rowEnd	entier long	The last row to print in a cell range. Default value = -1 (all rows)
rowStart	entier long	The first row to print in a cell range. Default value = -1 (all rows)

Headers / Footers

Header and footer attributes are used to specify text or images in the left, right, and center header/footer sections.

Propriété	Type	Description
footerCenter	Texte	The text and format of the center footer on printed pages.
footerCenterImage	picture text*	The image for the center section of the footer.
footerLeft	Texte	The text and format of the left footer on printed pages.
footerLeftImage	picture text*	The image for the left section of the footer.
footerRight	Texte	The text and format of the right footer on printed pages.
footerRightImage	picture text*	The image for the right section of the footer.
headerCenter	Texte	The text and format of the center header on printed pages.
headerCenterImage	picture text*	The image for the center section of the header.
headerLeft	Texte	The text and format of the left header on printed pages.
headerLeftImage	picture text*	The image for the left section of the header.
headerRight	Texte	The text and format of the right header on printed pages.
headerRightImage	picture text*	The image for the right section of the header.

* If using text type, pass the filepath (absolute or relative) of the image. If you pass a relative path, the file should be located next to the database structure file. In Windows, the file extension must be indicated. No matter the type used to set an image, the image itself (not a reference) is stored in the 4D View Pro area and is returned by [VP Get print info](#).

Caractères spéciaux

The following special characters allow the automatic addition or formatting of information in the header and footer when the 4D View Pro area is printed.

Caractère	Description	Exemple	Résultat
&	Escape character	printInfo.headerLeft:="This is page &P."	
P	Current page	printInfo.headerLeft:="This is page &P."	This is page 5.
N	Page count	printInfo.headerLeft:="There are &N pages."	There are 10 pages.
D	Current date (yyyy/mm/dd format)	printInfo.headerLeft:="It is &D."	It is 2015/6/19.
T	Current time	printInfo.headerLeft:="It is &T."	It is 16:30:36.
G	Image	printInfo.headerLeftImage:=smiley printInfo.headerLeft:="&G"	
S	Strikethrough	printInfo.headerLeft:="&SThis is text."	This is text.
U	Souligné	printInfo.headerLeft:="&UTHis is text."	<u>This is text.</u>
B	Gras	printInfo.headerLeft:="&BThis is text."	This is text.
I	Italique	printInfo.headerLeft:="&IThis is text."	<i>This is text.</i>
nom : "Gestionnaire REST : < nom du process > "	Font prefix	printInfo.headerLeft:="&"Lucida Console"&14This is text."	This is text.
K	Text Color prefix	printInfo.headerLeft:="&KFF0000This is text."	This is text.
F	Workbook name	printInfo.headerLeft:="&F"	2019 Monthly Revenue Forecasts
A	Spreadsheet name	printInfo.headerLeft:="&A"	June 2019 revenue forecast

Margins

Margin attributes are used to specify the 4D View Pro area margins for printing. Expressed in hundreds of an inch.

Propriété	Type	Description	
margin	object	The print margins	
	top	entier long	Top margin, in hundredths of an inch. Default value = 75
	bottom	entier long	Bottom margin, in hundredths of an inch. Default value = 75
	left	entier long	Left margin, in hundredths of an inch. Default value = 70
	right	entier long	Right margin, in hundredths of an inch. Default value = 70
	header	entier long	Header offset, in hundredths of an inch. Default value = 30
	footer	entier long	Footer offset, in hundredths of an inch. Default value = 30

Orientation

Orientation attributes are used to specify the direction the printed page layout.

This attribute defines rendering information only.

Propriété	Type	Description
orientation	entier long	Page orientation. Available values: <code>vk print page orientation landscape</code> , <code>vk print page orientation portrait</code> (default)

Page

Page attributes are used to specify general document print settings.

Propriété	Type	Description
blackAndWhite	boolean	Printing in black and white only. Default value = false Note: PDFs are not affected by this attribute. Colors in PDFs remain.
centering	entier long	How the contents are centered on the printed page. Available values: <code>vk print centering both</code> , <code>vk print centering horizontal</code> , <code>vk print centering none</code> (default), <code>vk print centering vertical</code>
firstPageNumber	entier long	The page number to print on the first page. Default value = 1
pageOrder	entier long	The order pages are printed. Available values: <code>vk print page order auto</code> (default), <code>vk print page order down then over</code> , <code>vk print page order over then down</code> .
pageRange	Texte	The range of pages for printing
qualityFactor	entier long	The quality factor for printing (1 - 8). The higher the quality factor, the better the printing quality, however printing performance may be affected. Default value = 2
useMax	boolean	Only columns and rows with data are printed. Default value = true
zoomFactor	réel	The amount to enlarge or reduce the printed page. Default value = 1

Paper Size

Paper size attributes are used to specify the dimensions or model of paper to use for printing. There are two ways to define paper size:

- Custom size - height and width attributes
- Standard size - kind attribute

Propriété		Type	Description
paperSize		object	Paper dimensions (height, width) or specific format (kind) for printing.
	height	entier long	Height of the paper, in hundredths of an inch.
	width	entier long	Width of the paper, in hundredths of an inch.
	kind	Texte	Name of standard paper size (e.g., A2, A4, legal, etc.) returned by <code>Get Print Option</code> . Default value = "letter"

Echelle

Scale attributes are used to specify printing optimization and adjustments.

Propriété	Type	Description
bestFitColumns	boolean	Column width is adjusted to fit the largest text width for printing. Default value = "false"
bestFitRows	boolean	Row height is adjusted to fit the tallest text height for printing. Default value = "false"
fitPagesTall	entier long	The number of vertical pages (portrait orientation) to check when optimizing printing. Default value = -1
fitPagesWide	entier long	The number of horizontal pages (landscape orientation) to check when optimizing printing. Default value = -1

Show / Hide

Show / Hide attributes are used to specify the visibility (printing) of 4D View Pro area elements.

Propriété	Type	Description
showBorder	boolean	Prints the outline border. Default value = "true"
show.ColumnHeader	entier long	Column header print settings. Available values: <code>vk print visibility hide</code> , <code>vk print visibility inherit</code> (default), <code>vk print visibility show</code> , <code>vk print visibility show once</code>
show.GridLine	boolean	Prints the gridlines. Default value = "false"
show.RowHeader	entier long	Row headers print settings. Available values: <code>vk print visibility hide</code> , <code>vk print visibility inherit</code> (default), <code>vk print visibility show</code> , <code>vk print visibility show once</code>

Watermark

Watermark attributes are used to superimpose text or an image onto the 4D View Pro area.

Propriété		Type	Description
watermark		collection	Collection of watermark settings. Default value: undefined
	[].height	entier long	The height of the watermark text / image.
	[].imageSrc	picture text*	The watermark text / image.
	[].page	Texte	The page(s) where the watermark is printed. For all pages: "all". For specific pages: page numbers or page ranges separated by commas. Ex.: "1,3,5-12"
	[].width	entier long	The width of the watermark text / image.
	[].x	entier long	The horizontal coordinate of the top left point of the watermark text / image.
	[].y	entier long	The vertical coordinate of the top left point of the watermark text / image.

* If using text type, pass the filepath (absolute or relative) of the image. If you pass a relative path, the file should be located next to the database structure file. In Windows, the file extension must be indicated. No matter the type used to set an image, the image itself (not a reference) is stored in the 4D View Pro area and is returned by [VP Get print info](#).

Style Objects

4D View Pro style objects and style sheets allow you to control the graphical aspects and the look of your 4D View Pro documents.

Style objects & Style sheets

Style objects contain the style settings. They can be used either in a style sheet or on their own. Style objects can also be used in addition to a style sheet so that different settings can be specified for individual cell ranges without affecting the rest of the document. You can use style objects directly with the [VP SET CELL STYLE](#) and [VP SET DEFAULT STYLE](#) commands.

A style sheet groups together a combination of properties in a style object to specify the look of all of the cells in your 4D View Pro documents. Style sheets saved with the document can be used to set the properties for a single sheet, multiple sheets, or an entire workbook. When created, a 4D View Pro style sheet is given a name which is saved within the style sheet in the "name" property. This allows a style sheet to be easily used and, if thoughtfully selected, can facilitate its identification and purpose (e.g., Letterhead_internal, Letterhead_external).

Style sheets are created with the [VP ADD STYLESHEET](#) command and applied with the the [VP SET DEFAULT STYLE](#) or [VP SET CELL STYLE](#) commands. You can remove a style sheet with the [VP REMOVE STYLESHEET](#) command.

The [VP Get stylesheet](#) command can be used to return the style object of a single style sheet or you can use the [VP Get stylesheets](#) command to retrieve a collection of style objects for multiple style sheets.

Style object properties

Exemple :

```

$style:=New object
$style.hAlign:=vk horizontal align left
$style.font:="12pt papyrus"
$style.backColor:="#E6E6FA" //light purple color

VP SET DEFAULT STYLE("myDoc";$style)

```

Background & Foreground

Propriété	Type	Description	Possible values
backColor	Texte	Defines the color of the background.	CSS color "#rrggb" syntax (preferred syntax), CSS color "rgb(r,g,b)" syntax (alternate syntax), CSS color name (alternate syntax)
backgroundImage	picture, text	Specifies a background image.	Can be specified directly or via the image path (full path or file name only). If the file name only is used, the file must be located next to the database structure file. No matter how set (picture or text), a picture is saved with the document. This could impact the size of a document if the image is large. Note for Windows: File extension must be included.
backgroundImageLayout	entier long	Defines the layout for the background image.	vk image layout center, vk image layout none, vk image layout stretch, vk image layout zoom
foreColor	Texte	Defines the color of the foreground.	CSS color "#rrggb" syntax (preferred syntax), CSS color "rgb(r,g,b)" syntax (alternate syntax), CSS color name (alternate syntax)

Bordures

Propriété		Type	Description	Possible values
borderBottom, borderLeft, borderRight, borderTop, diagonalDown, diagonalUp		object	Defines the corresponding border line	
	color	Texte	Defines the color of the border. Default = black.	CSS color "#rrggb" syntax (preferred syntax), CSS color "rgb(r,g,b)" syntax (alternate syntax), CSS color name (alternate syntax)
	style	entier long	Defines the style of the border. Default = empty. Cannot be null or undefined.	vk line style dash dot, vk line style dash dot dot, vk line style dashed, vk line style dotted, vk line style double, vk line style empty, vk line style hair, vk line style medium, vk line style medium dash dot, vk line style medium dash dot dot, vk line style medium dashed, vk line style slanted dash dot, vk line style thick

Police et texte

Propriété	Type	Description	Possible values
-----------	------	-------------	-----------------

Propriété	Type	Description	Possible values	
font	Texte	<p>Specifies the font characteristics in CSS font shorthand ("font-style font-variant font-weight font-size/line-height font-family"). Example: "14pt Century Gothic". The font-size and font-family values are mandatory. If one of the other values is missing, their default values are used. Note: If a font name contains a space, the name must be within quotes.</p>	<p>Possible values A CSS font shorthand. 4D provides utility commands to handle font characteristics as objects: VP Font to object and VP Object to font</p>	
formatter	Texte	Pattern for value/time property.	Number/text/date/time formats, special characters. See Cell Format .	
isVerticalText	boolean	Specifies text direction.	True = vertical text, False = horizontal text.	
labelOptions	object	Defines cell label options (watermark options).		
	alignment	entier long	Specifies the position of the cell label. Optional property.	<pre>vk_label_alignment_top left , vk_label_alignment_bottom_left , vk_label_alignment_top_center , vk_label_alignment_bottom_center , vk_label_alignment_top_right , vk_label_alignment_bottom_right</pre>
	visibility	entier long	Specifies the visibility of the cell label. Optional property.	<pre>vk_label_visibility_auto , vk_label_visibility_hidden , vk_label_visibility_visible</pre>
	foreColor	Texte	Defines the color of the foreground. Optional property.	CSS color "#rrggbb" syntax (preferred syntax), CSS color "rgb(r,g,b)" syntax (alternate syntax), CSS color name (alternate syntax)
	font	Texte	Specifies the font characteristics with CSS font shorthand ("font-style font-variant font-weight font-size/line-height font-family"). The font-size and font-family values are mandatory.	
textDecoration		entier long	Specifies the decoration added to text.	<pre>vk_text_decoration_double_underline , vk_text_decoration_line_through , vk_text_decoration_none , vk_text_decoration_overline , vk_text_decoration_underline</pre>
textIndent		entier long	Defines the unit of text indentation. 1 = 8 pixels	

Propriété	Type	Description	Possible values
textOrientation	long	Defines the rotation angle of the text in a cell. Number between -90 and 90	
watermark	Texte	Defines the watermark (cell label) content	
wordWrap	boolean	Specifies if text should be wrapped.	True = wrapped text, False = unwrapped text

Affichage

Propriété	Type	Description	Possible values
cellPadding	Texte	Defines the cell padding	
hAlign	entier long	Defines the horizontal alignment of cell contents.	vk horizontal align center , vk horizontal align general , vk horizontal align left , vk horizontal align right
locked	boolean	Specifies cell protection status. Note, this is only available if sheet protection is enabled.	True = locked, False = unlocked.
shrinkToFit	boolean	Specifies if the contents of the cell should be reduced.	True = reduced content, False = no reduction.
tabStop	boolean	Specifies if the focus to the cell can be set using the Tab key.	True = Tab key sets focus, False = Tab key does not set focus.
vAlign	entier long	Specifies the vertical alignment of cell contents.	vk vertical align bottom , vk vertical align center , vk vertical align top

Style information

Propriété	Type	Description
name	Texte	Defines the name of the style
parentName	Texte	Specifies the style that the current style is based on. Values from the parent style will be applied, then any values from the current style are applied. Changes made in the current style will not be reflected in the parent style. Only available when using a style sheet.

Objet 4D View Pro

The 4D View Pro [object](#) stores the whole spreadsheet contents. It is automatically handled by 4D View Pro. You can set or get this object using the [VP IMPORT FROM OBJECT](#) or [VP Export to object](#) methods.

It contains the following properties:

Propriété	Value type	Description
version	Longint	Internal component version
dateCreation	Timestamp	Creation date
dateModified	Timestamp	Last modification date
meta	Object	Free contents, reserved for the 4D developer
spreadJS	Object	Reserved for the 4D View Pro component

4D View Pro Form Object Variable

The 4D View Pro form object variable is the [object](#) variable associated to the 4D View Pro form area. It manages information used by the 4D View Pro object.

The 4D View Pro form object variable is for information purposes only (i.e., debugging). Under no circumstances should it be modified.

It contains the following properties:

Propriété	Value type	Description
ViewPro.area	Text	Nom de la zone 4D View Pro
ViewPro.callbacks	Object	Stores temporary information necessary for commands requiring callbacks such as importing and exporting.
ViewPro.commandBuffers	Collection	Stores sequentially the commands called by the method and executes them as a batch (rather than individually) upon exiting the method, or if a command returns a value or the VP FLUSH COMMANDS is called. This mechanism increases performance by reducing the number of requests sent.
ViewPro.events	Object	Event list.
ViewPro.formulaBar	Booléen	Indicates whether or not the formula bar is displayed. Available only for the "toolbar" interface.
ViewPro.initiated	Booléen	Indicates whether or not the 4D View Pro area has been initialized (see On VP Ready event).
ViewPro.interface	Text	Specifies the type of user interface:"ribbon", "toolbar", "none".

Formules et fonctions

Utilisation des formules

Une formule de feuille de calcul est une expression qui calcule la valeur d'une cellule.

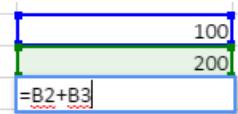
Saisie des formules

Pour saisir une formule dans une zone de 4D View Pro :

1. Sélectionnez la cellule dans laquelle vous allez saisir la formule ou la fonction.
2. Saisissez = (le signe égal).
3. Tapez la formule et appuyez sur la touche Entrée.

Lorsque vous écrivez une formule, vous pouvez utiliser différents raccourcis :

- cliquez sur une cellule pour entrer sa référence dans la formule :



- tapez la première lettre d'une fonction à saisir. Un menu contextuel listant les fonctions et références disponibles apparaît, pour vous permettre de sélectionner les éléments souhaités :

□

Vous pouvez également créer des formules nommées qui peuvent être appelées via leur nom. Pour ce faire, saisissez ces formules à l'aide de la commande [VP ADD FORMULA NAME](#).

Opérateurs et opérandes

Toutes les formules ont des opérandes et des opérateurs :

- Opérateurs : voir [Valeurs et opérateurs](#) ci-dessous.
- Les opérandes comprennent plusieurs catégories :
 - Les [valeurs](#) (5 types de données sont pris en charge)
 - Les [références](#) à d'autres cellules (relatives, absolues, mixtes ou par nom)
 - [fonctions standard de tableau](#)
 - Les [fonctions 4D](#) basées sur des formules 4D et donnant accès à des variables, champs, méthodes, commandes ou expressions 4D.

Valeurs et opérateurs

4D View Pro prend en charge cinq types de données. Pour chaque type de données, des valeurs littérales et des opérateurs spécifiques sont pris en charge.

Types de données	Valeurs	Opérateurs
Nombre	1.2 1.2 E3 1.2E-3 10.3x	+ (addition) - (soustraction) * (multiplication) / (division) ^ (exposant, le nombre de fois qu'il faut multiplier un nombre par lui-même) % (pourcentage -- diviser par cent le nombre précédent l'opérateur)
Date	10/24/2017	+ (date + nombre de jours -> date) + (date + heure -> date + heure) - (date - nombre de jours -> date) - (date - date -> nombre de jours entre les deux)
Heure	10:12:10	Opérateurs sur les durées : + (addition) - (soustraction) * (durée * nombre -> durée) / (durée/ nombre-> durée)
Chaine	'Sophie' ou "Sophie"	& (concaténation)
Booléen	TRUE ou FALSE	-

Opérateurs de comparaison

Les opérateurs suivants peuvent être utilisés avec deux opérandes de même type :

Opérateur	Comparaison
=	est égal à
<>	est différent de
>	supérieur à
<	inférieur à
>=	supérieur ou égal à
<=	inférieur ou égal à

Préséance des opérateurs

Liste des opérateurs, du plus important au moins important :

Opérateur	Description
()	Parenthèse (pour regrouper)
-	Négatif
+	Plus
%	Pourcentage
^	Exposant
* et /	Multiplier et diviser
+ et -	Ajouter et soustraire
&	Concaténer
= > < >= <= <>	Comparer

Références de cellules

Les formules font souvent référence à d'autres cellules par des adresses de cellule. Vous pouvez copier ces formules dans d'autres cellules. Par exemple, la formule suivante, saisie dans la cellule C8, additionne les valeurs des deux cellules situées au-dessus et affiche le résultat.

= C6 + C7

Cette formule fait référence aux cellules C6 et C7. En d'autres termes, le logiciel 4D View Pro reçoit l'instruction de se référer à ces autres cellules pour trouver les valeurs à utiliser dans la formule.

Lorsque vous copiez ou déplacez ces formules vers de nouveaux emplacements, chaque adresse de cellule dans cette formule change ou reste la même, selon la façon dont elle est tapée.

- Une référence qui change est appelée référence relative et fait référence à une cellule en fonction de sa distance à gauche/droite et en haut/bas par rapport à la cellule contenant la formule.
- Une référence qui pointe toujours vers une cellule particulière est appelée référence absolue.
- Vous pouvez également créer une référence mixte qui pointe toujours vers une ligne ou une colonne fixe.

Notation des références

Si vous utilisez uniquement des coordonnées de cellule, telle qu'une cellule C5 par exemple, 4D View Pro interprète la référence comme étant relative. Vous pouvez faire de la référence une référence absolue en précédant la lettre et le chiffre du signe dollar, comme dans \$C\$5 .

Vous pouvez combiner les références absolues et relatives en insérant un signe dollar devant la lettre ou le chiffre seul, par exemple, \$C5 ou C\$5 . Une référence mixte vous permet de spécifier la ligne ou la colonne comme absolue, tout en permettant à l'autre partie de l'adresse de se référer de manière relative.

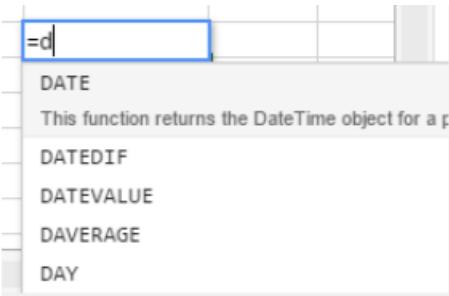
Un moyen pratique, rapide et précis de spécifier une référence absolue consiste à nommer la cellule et à utiliser ce nom à la place de l'adresse de la cellule. Une référence à une cellule nommée est toujours absolue. Vous pouvez créer ou modifier des cellules ou des plages de cellules nommées à l'aide de la méthode VP ADD RANGE NAME .

Le tableau suivant montre l'effet des différentes notations :

Exemple	Type de référence	Description
C5	Relative	La référence concerne l'emplacement relatif de la cellule C5, en fonction de l'emplacement de la cellule dans laquelle la référence est utilisée pour la première fois
\$C\$5	Absolue	La référence est absolue. Elle fait toujours référence à la cellule C5, quel que soit l'endroit où elle est utilisée.
\$C5	Mixed	La référence est toujours faite à la colonne C, mais la référence de ligne est relative à l'emplacement de la cellule dans laquelle la référence est utilisée pour la première fois.
C\$5	Mixed	La référence est toujours la ligne 5, mais la référence de la colonne est relative à l'emplacement de la cellule dans laquelle la référence est utilisée pour la première fois
Nom de cellule	Absolue	La référence est absolue. Elle fait toujours référence à la cellule ou à la plage nommée , quel que soit l'endroit où la référence est utilisée.

Fonctions intégrées

Les fonctions de tableur sont des formules prédéfinies utilisées pour calculer les valeurs des cellules. Lorsque vous tapez la première lettre de la fonction à saisir, un menu contextuel répertoriant les fonctions et références disponibles apparaît, vous permettant de sélectionner les éléments souhaités :



Consultez la [liste étendue des fonctions de SpreadJS](#) pour obtenir des détails et des exemples.

Fonctions 4D

4D View Pro vous permet de définir et d'appeler des fonctions 4D personnalisées, qui exécutent des [formules 4D](#). L'utilisation de fonctions personnalisées 4D étend les possibilités de vos documents 4D View Pro et permet des interactions avancées avec la base de données 4D.

Les fonctions personnalisées 4D permettent d'accéder, à partir de vos formules 4D View Pro :

- aux variables process 4D,
- aux champs,
- aux méthodes projet,
- aux commandes de langage 4D,
- ou à toute autre expression 4D valide.

Les fonctions personnalisées 4D peuvent recevoir des [paramètres](#) de la zone 4D View Pro, et retourner des valeurs.

Déclarez toutes vos fonctions à l'aide de la méthode [VP SET CUSTOM FUNCTIONS](#). Voici quelques exemples :

```

o:=New object

//Nom de la fonction dans 4D View Pro : "DRIVERS_LICENCE"
$o.DRIVERS_LICENCE:=New object

/variable process
$o.DRIVERS_LICENCE.formula:=Formula(DriverLicence)

//champ table
$o.DRIVERS_LICENCE.formula:=Formula([Users]DriverLicence)

//méthode projet
$o.DRIVERS_LICENCE.formula:=Formula(DriverLicenceState)

//Commande 4D
$o.DRIVERS_LICENCE:=Formula(Choose(DriverLicence; "Obtained"; "Failed"))

//expression et paramètre 4D
$o.DRIVERS_LICENCE.formula:=Formula(ds.Users.get($1).DriverLicence)
$o.DRIVERS_LICENCE.parameters:=New collection
$o.DRIVERS_LICENCE.parameters.push(New object("name"; "ID"; "type"; Is longint))

```

Voir également [4D View Pro : Utilisation des formules 4D dans votre feuille de calcul \(article de blog\)](#)

Exemple avec Hello World

Nous souhaitons imprimer "Hello World" dans une cellule de la zone 4D View Pro en utilisant une méthode projet 4D :

- Créez une méthode projet "myMethod" avec le code suivant :

```

#DECLARE->$hw Text
$hw:="Hello World"

```

- Exécutez le code suivant avant d'ouvrir tout formulaire contenant une zone 4D View Pro :

```

Case of
:(Form event code=On Load)
  var $o : Object
  $o:=New object
// Définir la fonction "vpHello" à partir de la méthode "myMethod"
  $o.vpHello:=New object
  $o.vpHello.formula:=Formula(myMethod)
  VP SET CUSTOM FUNCTIONS("ViewProArea";$o)
End case

```

- Modifiez le contenu d'une cellule dans une zone 4D View Pro et saisissez :

"myMethod" est alors appelé par 4D et la cellule s'affiche :

Paramètres

Les paramètres peuvent être passés aux fonctions 4D qui appellent les méthodes projet en utilisant la syntaxe suivante :

```
=METHODNAME(param1,param2,...,paramN)
```

Ces paramètres sont reçus dans *methodName* en \$1, \$2...\$N.

A noter que les () sont obligatoires, même si aucun paramètre n'est passé :

```
=METHODWITHOUTNAME()
```

Vous pouvez déclarer le nom, le type et le nombre de paramètres via la collection de *paramètres* de la fonction que vous avez déclarée à l'aide de la méthode [VP SET CUSTOM FUNCTIONS](#). En option, vous pouvez contrôler le nombre de paramètres passés par l'utilisateur à l'aide des propriétés *minParams* et *maxParams*.

Pour plus d'informations sur les types de paramètres entrants pris en charge, veuillez vous reporter à la description de la méthode [VP SET CUSTOM FUNCTIONS](#).

Si vous ne déclarez pas de paramètres, les valeurs peuvent être passées séquentiellement aux méthodes (elles seront reçues dans \$1, \$2...) et leur type sera automatiquement converti. Les dates en *jstype* seront passées comme **objet** dans le code 4D avec deux propriétés :

|Property| Type| Description| ---|---|---| value| Date| Valeur de la date| time |Real| Temps en secondes|

Les méthodes projet 4D peuvent également retourner des valeurs dans la formule de la cellule 4D View Pro via \$0. Les types de données suivants sont pris en charge pour les paramètres retournés :

- **text** (converti en chaîne de caractères dans 4D View Pro)
- **real/longint** (converti en numérique dans 4D View Pro)
- **date** (converti en type JS Date dans 4D View Pro - heure, minute, seconde = 0)
- **time** (converti en type JS Date dans 4D View Pro - date en date de base, c'est-à-dire 30/12/1899)
- **boolean** (converti en bool dans 4D View Pro)
- **image** (jpg,png,gif,bmp,svg autres types convertis en png) crée un URI (data:image/png;base64,xxxx) et ensuite utilisé comme fond dans 4D View Pro dans la cellule où la formule est exécutée
- **objet** avec les deux propriétés suivantes (permettant de passer une date et une heure) :

Propriété	Type	Description
value	Date	Valeur date
time	Réel	Heure en secondes

Si la méthode 4D ne retourne rien, une chaîne vide est automatiquement retournée.

Une erreur est retournée dans la cellule 4D View Pro si :

- la méthode 4D retourne un autre type que ceux listés ci-dessus,
- une erreur s'est produite pendant l'exécution de la méthode 4D (lorsque l'utilisateur clique sur le bouton "abort").

Exemple

```

var $o : Object

$o.BIRTH_INFORMATION:=New object
$o.BIRTH_INFORMATION.formula:=Formula(BirthInformation)
$o.BIRTH_INFORMATION.parameters:=New collection
$o.BIRTH_INFORMATION.parameters.push(New object("name";"First name";"type";Is text))
$o.BIRTH_INFORMATION.parameters.push(New object("name";"Birthday";"type";Is date))
$o.BIRTH_INFORMATION.parameters.push(New object("name";"Time of birth";"type";Is time))
$o.BIRTH_INFORMATION.summary:="Returns a formatted string from given information"

VP SET CUSTOM FUNCTIONS("ViewProArea"; $o)

```

□

Compatibilité

Des solutions alternatives sont disponibles pour déclarer des champs ou des méthodes en tant que fonctions dans vos zones 4D View Pro. Ces solutions sont maintenues pour des raisons de compatibilité et peuvent être utilisées dans des cas spécifiques. Toutefois, il est recommandé d'utiliser la méthode `VP SET CUSTOM FUNCTIONS`.

Référencement de champs à l'aide de la structure virtuelle

4D View Pro vous permet de référencer des champs 4D en utilisant la structure virtuelle de la base de données, c'est-à-dire en les déclarant à l'aide des commandes `SET TABLE TITLES` et/ou `SET FIELD TITLES` avec le paramètre *. Cette solution alternative peut être utile si votre application s'appuie déjà sur une structure virtuelle (sinon, il est recommandé d'utiliser `VP SET CUSTOM FUNCTIONS`).

ATTENTION : Vous ne pouvez pas utiliser la structure virtuelle et `VP SET CUSTOM FUNCTIONS` simultanément. Dès que `VP SET CUSTOM FUNCTIONS` est appelé, les fonctions basées sur les commandes `SET TABLE TITLES` et `SET FIELD TITLES` sont ignorées dans la zone 4D View Pro.

Conditions requises

- Le champ doit appartenir à la structure virtuelle de la base de données, c'est-à-dire qu'il doit être déclaré par les commandes `SET TABLE TITLES` et/ou `SET FIELD TITLES` avec le paramètre * (voir exemple),
- Les noms des tables et des champs doivent être conformes à la norme ECMA (voir la [norme ECMA Script](#)),
- Le type de champ doit être pris en charge par 4D View Pro (voir ci-dessus).

Une erreur est retournée dans la cellule 4D View Pro si la formule appelle un champ qui n'est pas conforme.

Appel d'un champ virtuel dans une formule

Pour insérer une référence à un champ virtuel dans une formule, saisissez le champ avec la syntaxe suivante :

TABLENAME_FIELDNAME()

Par exemple, si vous avez déclaré le champ "Name" de la table "People" dans la structure virtuelle, vous pouvez appeler les fonctions suivantes :

```
=PEOPLE_NAME()
=LEN(PEOPLE_NAME())
```

Si un champ porte le même nom qu'une [méthode 4D], il a la priorité sur la méthode.

Exemple

Nous souhaitons imprimer le nom d'une personne dans une cellule de la zone 4D View Pro en utilisant un champ virtuel 4D :

1. Créez une table "Employee" avec un champ "L_Name" :

Employee	
ID	2 ³²
L_Name	A
F_Name	A

2. Exécuter le code suivant pour initialiser une structure virtuelle :

```
ARRAY TEXT($tableTitles;1)
ARRAY LONGINT($tableNum;1)
$tableTitles{1}:="Emp"
$tableNum{1}:=2
SET TABLE TITLES($tableTitles;$tableNum;*)

ARRAY TEXT($fieldTitles;1)
ARRAY LONGINT($fieldNum;1)
$fieldTitles{1}:="Name"
$fieldNum{1}:=2 //last name
SET FIELD TITLES([Employee];$fieldTitles;$fieldNum;*)
```

3. Modifiez le contenu d'une cellule de la zone 4D View Pro et saisir "=e" :

	A	B	C	D	E
1	=e				
2	EDATE				
3	EFFECT				
4	EMP_NAME				
5	ENCODEURL				

4. Sélectionnez EMP_NAME (utilisez la touche Tab) et saisissez la fermeture).

	A	B
1	=EMP_NAME()	

5. Validez le champ pour afficher le nom de l'employé courant :

	A	B
1	Smith	

La table [Employee] doit avoir un enregistrement en cours.

Déclarer des méthodes autorisées

Vous pouvez appeler directement des méthodes du projet 4D à partir de vos formules 4D View Pro. Pour des raisons de sécurité, vous devez déclarer explicitement les méthodes qui peuvent être appelées par l'utilisateur avec la méthode **VP SET ALLOWED METHODS**.

Conditions requises

Pour être appelée dans une formule 4D View Pro, une méthode projet doit être :

- Autorisée : elle a été déclarée explicitement avec la méthode **VP SET ALLOWED METHODS**.

- Exécutable : elle appartient au projet hôte ou à un composant chargé dont l'option "Partagé par les composants et le projet hôte" est activée (voir [Partage des méthodes projet](#)).
- Pas de conflit avec une fonction de tableau 4D View Pro existante : si vous appelez une méthode projet portant le même nom qu'une fonction intégrée 4D View Pro, la fonction est appelée.

Si ni la méthode **VP SET CUSTOM FUNCTIONS** ni la méthode **VP SET ALLOWED METHODS** n'ont été exécutées pendant la session, les fonctions personnalisées de 4D View Pro reposent sur les méthodes autorisées définies par la commande générique `SET ALLOWED METHODS` de 4D. Dans ce cas, les noms des méthodes projet doivent être conformes à la grammaire d'identification JavaScript (voir la [norme ECMA Script](#)). L'option de filtrage global de la boîte de dialogue Paramètres (voir *Accès aux données*) est ignorée dans tous les cas.

Liste des méthodes

Attention : Les commandes listées dans cette page ne sont pas thread-safe.

A - C - D - E - F - G - I - M - N - O - P - R - S

A

VP ADD FORMULA NAME

VP ADD FORMULA NAME (*vpAreaName* : Text ; *vpFormula* : Text ; *name* : Text { ; *options* : Object })

Paramètres	Type		Description
<i>vpAreaName</i>	Text	->	Nom d'objet formulaire zone 4D View Pro
<i>vpFormula</i>	Text	->	Formule 4D View Pro
<i>name</i>	Text	->	Nom de la formule
<i>options</i>	Object	->	Options de la formule nommée

Description

La commande `VP ADD FORMULA NAME` permet de créer ou de modifier une formule nommée dans le document courant.

Les formules nommées créées par cette commande sont stockées dans le document.

Dans *vpAreaName*, passez le nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est retournée.

Passez la formule 4D View Pro que vous souhaitez nommer dans *vpFormula*. Pour plus d'informations sur la syntaxe des formules, veuillez consulter la section [Formules et Fonctions](#).

Passez le nouveau nom de la formule dans *name*. Si le nom est déjà utilisé dans le même scope, la nouvelle formule nommée remplace la formule existante. A noter que vous pouvez utiliser le même nom pour plusieurs scopes (ci-dessous).

Vous pouvez passer un objet avec des propriétés additionnelles pour la plage nommée dans *options*. Les propriétés suivantes sont prises en charge :

Propriété	Type	Description
<i>scope</i>	Nombre	Scope de la formule. Vous pouvez passer l'indice de la feuille (la numérotation commence à zéro) ou utiliser les constantes suivantes : <ul style="list-style-type: none">• <code>vk current sheet</code>• <code>vk workbook</code> Le scope détermine si le nom d'une formule est propre à une feuille (<code>scope=sheet index</code> ou <code>vk current sheet</code>), ou s'il s'applique à l'ensemble du classeur (<code>scope= vk workbook</code>).
<i>comment</i>	Text	Commentaire associé à une formule nommée

Exemple

```
VP ADD FORMULA NAME("ViewProArea";"SUM($A$1:$A$10)";"Total2")
```

Voir aussi

[Cell references](#)

[VP ADD RANGE NAME](#)

[VP Get formula by name](#)

[VP Get names](#)

VP ADD RANGE NAME

VP ADD RANGE NAME (*rangeObj* : Object ; *name* : Text { ; *options* : Object })

Paramètres	Type		Description
<i>rangeObj</i>	Text	->	Objet plage
<i>name</i>	Text	->	Nom de la formule
<i>options</i>	Object	->	Options de la formule nommée

Description

La commande `VP ADD RANGE NAME` crée ou modifie une plage nommée dans le document ouvert.

Les plages nommées créées par cette commande sont stockées dans le document.

Dans *rangeObj*, passez la plage que vous souhaitez nommer, et passez le nouveau nom de la plage dans *name*. Si le nom est déjà utilisé dans le même scope, la nouvelle plage nommée remplace la plage existante. A noter que vous pouvez utiliser le même nom pour plusieurs scopes (ci-dessous).

In *options*, you can pass an object that specifies additional options. Les propriétés suivantes sont prises en charge :

Propriété	Type	Description
scope	Nombre	Scope de la plage. Vous pouvez passer l'indice de la feuille (la numérotation commence à zéro) ou utiliser les constantes suivantes : <ul style="list-style-type: none">• <code>vk current sheet</code>• <code>vk workbook</code> Le scope détermine si le nom d'une plage est propre à une feuille (<code>scope=sheet index</code> ou <code>vk current sheet</code>), ou s'il s'applique à l'ensemble du classeur (<code>scope= vk workbook</code>).
comment	Text	Commentaire associé à une plage nommée

- Une plage nommée est en réalité une formule nommée contenant des coordonnées. `VP ADD RANGE NAME` facilite la création de plages nommées, mais la commande `VP ADD FORMULA NAME` permet également d'en créer.
- Les formules définissant des plages nommées peuvent être récupérées à l'aide de la commande `VP Get formula by name`.

Exemple

Vous souhaitez créer une plage nommée à partir d'une plage contenant une cellule :

```
$range:=VP Cell("ViewProArea";2;10)
VP ADD RANGE NAME($range;"Total1")
```

Voir aussi

[VP ADD FORMULA NAME](#)

[VP Get formula by name](#)

[VP Get names](#)

[VP Name](#)

VP ADD SELECTION

VP ADD SELECTION (*rangeObj* : Object)

Paramètres	Type		Description
rangeObj	Text	->	Objet plage

Description

La commande `VP ADD SELECTION` ajoute les cellules spécifiées aux cellules sélectionnées.

Dans *rangeObj*, passez un objet plage de plusieurs cellules à ajouter à la sélection courante.

La cellule active n'est pas modifiée.

Exemple

Les cellules sélectionnées sont les suivantes :

	A	B	C	D	E	F	G
1							
2		Month	Area	Mr. Smith	Ms. Johnson	Ms. Williams	Mr. Jones
3	January	South		1898	1857	1651	1448
4		East		4859	4857	2548	4876
5		North		2458	1524	6150	4987
6		West		5787	1580	3975	4878
7		Total		15002	9818	14324	16189
8	February	South		6668	4374	17495	9999
9		East		5955	1677	7944	9400
10		North		1000	6722	2195	2777
11		West		6896	8355	7195	2058
12		Total		20519	21128	34829	24234
13	March	South		2577	2000	6185	2704
14		East		4859	4857	2548	4876
15		North		2458	1524	6150	4987
16		West		5787	1580	3975	4878
17		Total		15681	9961	18858	17445

Le code suivant ajoutera des cellules à votre sélection :

```
$currentSelection:=VP_Cells("myVPArea";3;4;2;3)  
VP_ADD_SELECTION($currentSelection)
```

Résultat :

A	B	C	D	E	F	G
1						
2	Month	Area	Mr. Smith	Ms. Johnson	Ms. Williams	Mr. Jones
3	January	South	1898	1857	1651	1448
4		East	4859	4857	2548	4876
5		North	2458	1524	6150	4987
6		West	5787	1580	3975	4878
7		Total	15002	9818	14324	16189
8	February	South	6668	4374	17495	9999
9		East	5955	1677	7944	9400
10		North	1000	6722	2195	2777
11		West	6896	8355	7195	2058
12		Total	20519	21128	34829	24234
13	March	South	2577	2000	6185	2704
14		East	4859	4857	2548	4876
15		North	2458	1524	6150	4987
16		West	5787	1580	3975	4878
17		Total	15681	9961	19858	17415

Voir aussi

[VP Get active cell](#)

[VP Get selection](#)

[VP RESET SELECTION](#)

[VP SET ACTIVE CELL](#)

[VP SET SELECTION](#)

[VP SHOW CELL](#)

VP ADD SHEET

VP ADD SHEET (*vpAreaName* : Text)

VP ADD SHEET (*vpAreaName* : Text ; *index* : Integer)

VP ADD SHEET (*vpAreaName* : Text ; *index* : Integer ; *name* : Text)

Paramètres	Type		Description
<i>vpAreaName</i>	Text	->	Nom d'objet formulaire zone 4D View Pro
<i>index</i>	Integer	->	Indice de la nouvelle feuille
<i>name</i>	Text	->	Nom de la feuille

Description

La commande `VP ADD SHEET` insère une feuille dans le document chargé dans *vpAreaName*. Dans *vpAreaName*, passez le nom de la zone 4D View Pro.

Dans *index*, vous pouvez passer un numéro pour la nouvelle feuille. In *sheet*, you can pass an index for the new sheet. Si l'*index* passé est inférieur ou égal à 0, la commande insère la nouvelle feuille au début.

La numérotation démarre à 0.

Dans *name*, vous pouvez passer un nom pour la nouvelle feuille. Le nouveau nom ne peut pas contenir les caractères suivants : *, :, [,], ?, \, /

Exemple

Le document comporte actuellement 3 feuilles :



Pour insérer une feuille à la troisième position (*index* 2) et la nommer "March" :

```
VP ADD SHEET("ViewProArea";2;"March")
```



Voir aussi

[VP REMOVE SHEET](#)

VP ADD SPAN

VP ADD SPAN (*rangeObj* : Object)

Paramètres	Type		Description
<i>rangeObj</i>	Object	->	Objet plage

Description

La commande `VP ADD SPAN` combine les cellules de *rangeObj* en une seule cellule fusionnée.

Dans *rangeObj*, passez une plage de cellules. Les cellules de la plage sont jointes, afin de créer une cellule plus large qui s'étend sur plusieurs colonnes et/ou lignes. Vous pouvez passer plusieurs plages de cellules pour créer plusieurs fusions de cellules en même temps. A noter que si les plages de cellules se chevauchent, seule la première plage est utilisée.

- Seules les données contenues dans la cellule supérieure gauche sont affichées. Les données des autres cellules combinées sont cachées jusqu'à ce que la fusion soit retirée.
- Les données masquées, contenues dans les cellules fusionnées, sont accessibles via des formules (commençant par la cellule supérieure gauche).

Exemple

Pour fusionner les cellules First quarter et Second quarter avec les deux cellules côté à côté, et de fusionner la cellule South area avec les deux lignes en-dessous :

		First quart			Second qu		
		Jan	Feb	Mar	Apr	May	Jun
South area	John						
	Sahra						
	Dixie						

```
// Plage First quarter
$q1:=VP Cells("ViewProArea";2;3;3;1)

// Plage Second quarter
$q2:=VP Cells("ViewProArea";5;3;3;1)

// Plage South area
$south:=VP Cells("ViewProArea";0;5;1;3)

VP ADD SPAN(VP Combine ranges($q1;$q2;$south))
```

	First quarter			Second quarter		
	Jan	Feb	Mar	Apr	May	Jun
South area	John					
	Sahra					
	Dixie					

Voir aussi

[4D View Pro Range Object Properties](#)

[VP Get spans](#)

[VP REMOVE SPAN](#)

VP ADD STYLESHEET

`VP ADD STYLESHEET (vpAreaName : Text ; styleName : Text ; styleObj : Object { ; scope : Integer })`

Paramètres	Type		Description
vpAreaName	Text	->	Nom d'objet formulaire zone 4D View Pro
styleName	Text	->	Nom du style
styleObj	Object	->	Objet définissant les propriétés de l'attribut
scope	Integer	->	Cible (par défaut = feuille courante)

Description

La commande `VP ADD STYLESHEET` crée ou modifie la feuille de style `styleName` basée sur la combinaison de propriétés indiquées dans `styleObj` dans le document courant. Si une feuille de style ayant le même nom et la même cible existe déjà dans le document, cette commande l'écrasera et le remplacera par les nouvelles valeurs.

Les feuilles de style créées par cette commande sont sauvegardées avec le document.

Dans `vpAreaName`, passez le nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est renvoyée.

Le paramètre `styleName` vous permet de nommer la feuille de style. Si le nom existe déjà dans le même scope, la nouvelle feuille de style remplace la feuille existante. A noter que vous pouvez utiliser le même nom pour plusieurs scopes (ci-dessous).

Dans `styleObj`, définissez les propriétés de la feuille de style (ex : police, alignement, bordures, etc.). Pour consulter la liste complète des propriétés, reportez-vous à la section [Propriétés des objets de style](#).

Vous pouvez désigner l'emplacement dans lequel vous souhaitez définir la feuille de style dans le paramètre optionnel `scope`, à l'aide de l'indice de la feuille (la numérotation commence à zéro) ou à l'aide des constantes suivantes :

- `vk current sheet`
- `vk workbook`

Si une feuille de style `styleName` est définie au niveau du workbook et de la feuille lors de son paramétrage, le niveau de la feuille est prioritaire par rapport à celui du workbook.

Vous pouvez appliquer la feuille de style à l'aide des commandes [VP SET DEFAULT STYLE](#) ou [VP SET CELL STYLE](#).

Exemple

Le code suivant :

```

$styles:=New object
$styles.backColor:="green"

//Objet Bordure de ligne
$borders:=New object("color";"green";"style";vk line style medium dash dot)

$styles.borderBottom:=$borders
$styles.borderLeft:=$borders
$styles.borderRight:=$borders
$styles.borderTop:=$borders

VP ADD STYLESHEET("ViewProArea";"GreenDashDotStyle";$styles)

//Pour appliquer le style
VP SET CELL STYLE(VP Cells("ViewProArea";1;1;2;2);New object("name";"GreenDashDotStyle"))

```

créera et appliquera l'objet style *GreenDashDotStyle* suivant :

```

{
  backColor:green,
  borderBottom:{color:green,style:10},
  borderLeft:{color:green,style:10},
  borderRight:{color:green,style:10},
  borderTop:{color:green,style:10}
}

```

Voir aussi

[4D View Pro Style Objects and Style Sheets](#)
[VP Get stylesheet](#)
[VP Get stylesheets](#)
[VP REMOVE STYLESHEET](#)
[VP SET CELL STYLE](#)
[VP SET DEFAULT STYLE](#)

VP All

VP All (*vpAreaName* : Text { ; *sheet* : Integer }) : Object

Paramètres	Type		Description
<i>vpAreaName</i>	Text	->	Nom d'objet formulaire zone 4D View Pro
<i>sheet</i>	Integer	->	Indice de la feuille (feuille courante si omis)
Résultat	Object	<-	Objet plage de toutes les cellules

Description

La commande `VP All` retourne une nouvelle plage référençant toutes les cellules.

Dans *vpAreaName*, passez le nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est retournée.

Dans le paramètre optionnel *sheet*, vous pouvez désigner une feuille spécifique dans laquelle sera définie la plage (la numérotation commence à zéro). Si le paramètre est omis ou si vous passez `vk current sheet`, la feuille courante est utilisée.

Exemple

Vous souhaitez définir une plage pour toutes les cellules de la feuille courante :

```
$all:=VP All("ViewProArea") // toutes les cellules de la feuille courante
```

Voir aussi

[VP Cell](#)
[VP Cells](#)
[VP Column](#)
[VP Combine ranges](#)
[VP Name](#)
[VP Row](#)

C

VP Cell

VP Cell (*vpAreaName* ; *column* : Integer ; *row* : Integer ; Text { ; *sheet* : Integer }) : Object

Paramètres	Type		Description
<i>vpAreaName</i>	Text	->	Nom d'objet formulaire zone 4D View Pro
<i>column</i>	Longint	->	Indice de la feuille (feuille courante si omis)
<i>row</i>	Longint	->	Indice de la feuille (feuille courante si omis)
<i>sheet</i>	Integer	->	Indice de la feuille (feuille courante si omis)
Résultat	Object	<-	Objet plage de toutes les cellules

Description

La commande `VP Cell` retourne une nouvelle plage référençant une cellule spécifique.

Cette commande s'applique aux plages d'une seule cellule. Pour créer une plage de plusieurs cellules, utilisez la commande [VP Cells](#).

Dans *vpAreaName*, passez le nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est retournée.

Le paramètre *column* définit la colonne de la plage. Passez l'indice de la colonne dans ce paramètre.

Le paramètre *row* définit la ligne de la plage. Passez l'indice de la ligne dans ce paramètre.

Dans le paramètre optionnel *sheet*, vous pouvez désigner l'indice de la feuille dans laquelle sera définie la plage. Si le paramètre est omis ou si vous passez `vk current sheet`, la feuille courante est utilisée par défaut.

l'indexation démarre à 0.

Exemple

Vous souhaitez définir une plage pour la cellule de la feuille courante (sur la feuille courante) :

	A	B	C	D	E	F
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						

Le code est le suivant :

```
$cell:=VP Cell("ViewProArea";2;4) // C5
```

Voir aussi

[VP All](#)
[VP Cells](#)
[VP Column](#)
[VP Combine ranges](#)
[VP Name](#)
[VP Row](#)

VP Cells

VP Cells (vpAreaName : Text ; column: Integer ; row: Integer ; columnCount : Integer ; rowCount : Integer { ; sheet : Integer }) : Object

► Historique

Paramètres	Type		Description
vpAreaName	Text	->	Nom d'objet formulaire zone 4D View Pro
column	Integer	->	Indice de la colonne
row	Integer	->	Indice de la ligne
columnCount	Integer	->	Nombre de colonnes
rowCount	Integer	->	Nombre de lignes
sheet	Integer	->	Indice de la feuille (feuille courante si omis)
Résultat	Object	<-	Objet plage de toutes les cellules

Description

La commande `VP Cells` retourne une nouvelle plage référençant des cellules spécifiques.

Dans `vpAreaName`, passez le nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est retournée.

Le paramètre `column` définit la première colonne de la plage de cellules. Passez l'indice de la colonne (la numérotation commence à zéro) dans ce paramètre. Si la plage contient plusieurs colonnes, vous devez également utiliser le paramètre `columnCount`.

Dans le paramètre `row`, vous pouvez définir l'emplacement de la ou des lignes de la plage de cellules. Passez l'indice de la ligne (la numérotation commence à zéro) dans ce paramètre. Si la plage contient plusieurs lignes, vous devez également utiliser le paramètre `rowCount`.

Le paramètre `columnCount` vous permet de définir le nombre total de colonnes comprises dans la plage. `columnCount` doit être supérieur à 0.

Le paramètre *rowCount* vous permet de définir le nombre total de lignes comprises dans la plage. *rowCount* doit être supérieur à 0.

Dans le paramètre optionnel *sheet*, vous pouvez désigner une feuille spécifique dans laquelle sera définie la plage (la numérotation commence à zéro). Si le paramètre est omis ou si vous passez `vk current sheet`, la feuille courante est utilisée par défaut.

Exemple

Vous souhaitez définir un objet plage pour les cellules suivantes (de la feuille courante) :

	A	B	C	D	E	F
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						

Le code est le suivant :

```
$cells:=VP Cells("ViewProArea";2;4;2;3) // de C5 à D7
```

Voir aussi

[VP All](#)
[VP Cells](#)
[VP Column](#)
[VP Combine ranges](#)
[VP Name](#)
[VP Row](#)

VP Column

`VP Column (vpAreaName : Text ; column: Integer ; columnCount : Integer { ; sheet : Integer }) : Object`

Paramètres	Type		Description
vpAreaName	Text	->	Nom d'objet formulaire zone 4D View Pro
column	Integer	->	Indice de la colonne
columnCount	Integer	->	Nombre de colonnes
sheet	Integer	->	Indice de la feuille (feuille courante si omis)
Résultat	Object	<-	Objet plage de toutes les cellules

Description

La commande `VP Column` retourne une nouvelle plage référençant une ou plusieurs colonnes.

Dans *vpAreaName*, passez le nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est retournée.

Le paramètre *column* définit la première colonne de la plage. Passez l'indice de la colonne (la numérotation commence à zéro) dans ce paramètre. Si la plage contient plusieurs colonnes, vous devez également utiliser le paramètre optionnel *columnCount*.

Le paramètre optionnel *columnCount* vous permet de définir le nombre total de colonnes comprises dans la plage. *columnCount* doit être supérieur à 0. Si le paramètre est omis, la valeur 1 sera définie par défaut et une plage de type colonne sera créée.

Dans le paramètre optionnel *sheet*, vous pouvez désigner une feuille spécifique dans laquelle sera définie la plage (la numérotation commence à zéro). Si le paramètre est omis ou si vous passez `vk current sheet`, la feuille courante est utilisée par défaut.

Exemple

Vous souhaitez définir une plage pour la colonne ci-dessous (dans la feuille courante) :

	A	B	C	D	E	F
1						
2						
3						
4						
5						
6						
7						
8						
9						

Le code est le suivant :

```
$column:=VP Column("ViewProArea";3) // colonne D
```

Voir aussi

[VP All](#)
[VP Cells](#)
[VP Column](#)
[VP Combine ranges](#)
[VP Name](#)
[VP Row](#)
[VP SET COLUMN ATTRIBUTES](#)

VP COLUMN AUTOFIT

VP COLUMN AUTOFIT (*rangeObj* : Object)

Paramètres	Type		Description
<i>rangeObj</i>	Object	->	Objet plage

Description

La commande `VP COLUMN AUTOFIT` dimensionne automatiquement dans *rangeObj* la ou les colonne(s) en fonction de leur contenu.

Dans *rangeObj*, passez un objet plage contenant une plage de colonnes dont la taille doit être gérée automatiquement.

Exemple

Les colonnes suivantes sont toutes de la même taille et n'affichent pas certaines parties du texte :

	A	B	C	D	E
1					
2					
3					
4					
5					

La sélection des colonnes et l'exécution du code suivant :

```
VP COLUMN AUTOFIT(VP Get selection("ViewProarea"))
```

redimensionne les colonnes pour correspondre à la taille du contenu :

	A	B	C	D	E
1		Hello World	The quick brown fox jumped over the lazy dog. TGI		
2					
3					
4					
c					

Voir aussi

[VP ROW AUTOFIT](#)

VP Combine Ranges

VP Combine Ranges (*rangeObj* : Object ; *otherRangeObj* : Object {;...*otherRangeObjN* : Object }) : Object

Paramètres	Type		Description
<i>rangeObj</i>	Object	->	Objet plage
<i>otherRangeObj</i>	Object	->	Objet plage
Résultat	Object	<-	Objet contenant une plage combinée

Description

La commande `VP Combine ranges` retourne une nouvelle plage qui incorpore au moins deux plages existantes. Toutes les plages doivent provenir de la même zone 4D View Pro.

Dans *rangeObj*, passez la première plage.

Dans *otherRangeObj*, passez une ou plusieurs autres plages à combiner avec *rangeObj*.

La commande incorpore *rangeObj* et *otherRangeObj* par référence.

Exemple

Vous souhaitez combiner des plages de type cellule, colonne et ligne dans une nouvelle plage distincte :

```
$cell:=VP Cell("ViewProArea";2;4) // C5
$column:=VP Column("ViewProArea";3) // colonne D
$row:=VP Row("ViewProArea";9) // ligne 10

$combine:=VP Combine ranges($cell;$column;$row)
```

Voir aussi

[VP All](#)

[VP Cells](#)

[VP Column](#)

[VP Combine ranges](#)

[VP Name](#)

[VP Row](#)

[VP SET COLUMN ATTRIBUTES](#)

VP Convert from 4D View

VP Convert from 4D View (*4DViewDocument* : Blob) : Object

Paramètres	Type		Description
4DViewDocument	Blob	->	Document 4D View
Résultat	Object	<-	Objet 4D View Pro

Description

La commande `VP Convert from 4D View` vous permet de convertir un document 4D View existant en un objet 4D View Pro.

Cette commande ne nécessite pas l'installation du plug-in 4D View dans votre environnement.

Dans le paramètre *4DViewDocument*, passez une variable ou un champ BLOB contenant le document 4D View à convertir. La commande retourne un objet 4D View Pro dans lequel toutes les informations stockées à l'origine dans le document 4D View sont converties en attributs 4D View Pro.

Exemple

Vous souhaitez obtenir un objet 4D View Pro à partir d'une zone 4D View stockée dans un BLOB :

```
C_OBJECT($vp0bj)
$vp0bj:=VP Convert from 4D View($pvcblob)
```

VP Convert to picture

`VP Convert to picture (vpObject : Object {; rangeObj : Object}) : Picture`

Paramètres	Type		Description
vpObject	Object	->	Objet 4D View Pro contenant la zone à convertir
rangeObj	Object	->	Objet plage
Résultat	Object	<-	Image SVG de la zone

Description

La commande `VP Convert to picture` convertit l'objet *vpObject* 4D View Pro (ou la plage *rangeObj* dans *vpObject*) en une image SVG.

Cette commande est utile, par exemple :

- pour intégrer un document 4D View Pro dans un autre document, tel qu'un document 4D Write Pro
- pour imprimer un document 4D View Pro sans le charger dans une zone 4D View Pro.

Dans *vpObject*, passez l'objet 4D View Pro que vous souhaitez convertir. Cet objet doit avoir été préalablement analysé à l'aide de [VP Export to object](#) ou enregistré avec [VP EXPORT DOCUMENT](#).

Le processus de conversion SVG nécessite que les expressions et les formats (cf. [Format de cellule 4D View Pro](#)) inclus dans la zone 4D View Pro soient évalués au moins une fois, afin d'être correctement exportés. Si vous convertissez un document qui n'a pas été évalué au préalable, les expressions ou les formats peuvent être rendus de manière inattendue.

Dans *rangeObj*, passez une plage de cellules à convertir. Par défaut, si ce paramètre est omis, tout le contenu du document est converti.

Les contenus de document sont convertis en tenant compte de leurs attributs d'affichage, y compris les formats (voir la note ci-dessus), la visibilité des en-têtes, des colonnes et des lignes. La conversion des éléments suivants est prise en charge :

- Texte : style / police / taille / alignement / rotation / format
- Arrière-plan de cellule : couleur / image
- Bordure de cellule : épaisseur / couleur / style
- Fusion de cellules
- Images
- Hauteur de lignes
- Largeur de colonnes
- Visibilité : colonnes / lignes cachées.

La numérotation démarre à 0.

Résultat

La commande retourne une image au format SVG.

Exemple

Vous souhaitez convertir une zone 4D View Pro en SVG, prévisualiser le résultat et l'envoyer dans une variable image :

```
C_OBJECT($vpAreaObj)
C_PICTURE($vPict)
$vpAreaObj:=VP Export to object("ViewProArea")
$vPict:=VP Convert to picture($vpAreaObj) //exporter toute la zone
```

Voir aussi

[VP EXPORT DOCUMENT](#)
[VP Export to object](#)
[VP SET PRINT INFO](#)

VP Copy to object

► Historique

VP Copy to object (*rangeObj* : Object {; *options* : Object}) : Object

Paramètres	Type		Description
<i>rangeObj</i>	Object	->	Objet plage
<i>options</i>	Object	->	Options supplémentaires
Résultat	Object	<-	Objet retourné. Contient les données copiées

Description

La commande `VP Copy to object` copie le contenu, le style et les formules de `rangeObj` vers un objet.

Dans `rangeObj`, passez la plage de cellules contenant les valeurs, formatages et formules à copier. Si `rangeObj` est une plage combinée, seule la première est utilisée.

Vous pouvez passer un paramètre facultatif `options` contenant les propriétés suivantes :

Propriété	Type	Description														
copy	Booléen	Vrai (par défaut) pour conserver les valeurs, formatages et formules après exécution de la commande. Faux pour les supprimer.														
copyOptions	Longint	Spécifie ce qui est copié ou déplacé. Valeurs possibles : <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Valeur</th> <th style="text-align: left;">Description</th> </tr> </thead> <tbody> <tr> <td><code>vk clipboard options all</code> (par défaut)</td> <td>Copie tous les objets de données, y compris les valeurs, formats et formules.</td> </tr> <tr> <td><code>vk clipboard options formatting</code></td> <td>Copie uniquement les formats.</td> </tr> <tr> <td><code>vk clipboard options formulas</code></td> <td>Copie uniquement les formules.</td> </tr> <tr> <td><code>vk clipboard options formulas and formatting</code></td> <td>Copie les formules et les formats.</td> </tr> <tr> <td><code>vk clipboard options values</code></td> <td>Copie uniquement les valeurs.</td> </tr> <tr> <td><code>vk clipboard options value and formatting</code></td> <td>Copie les valeurs et les formats.</td> </tr> </tbody> </table>	Valeur	Description	<code>vk clipboard options all</code> (par défaut)	Copie tous les objets de données, y compris les valeurs, formats et formules.	<code>vk clipboard options formatting</code>	Copie uniquement les formats.	<code>vk clipboard options formulas</code>	Copie uniquement les formules.	<code>vk clipboard options formulas and formatting</code>	Copie les formules et les formats.	<code>vk clipboard options values</code>	Copie uniquement les valeurs.	<code>vk clipboard options value and formatting</code>	Copie les valeurs et les formats.
Valeur	Description															
<code>vk clipboard options all</code> (par défaut)	Copie tous les objets de données, y compris les valeurs, formats et formules.															
<code>vk clipboard options formatting</code>	Copie uniquement les formats.															
<code>vk clipboard options formulas</code>	Copie uniquement les formules.															
<code>vk clipboard options formulas and formatting</code>	Copie les formules et les formats.															
<code>vk clipboard options values</code>	Copie uniquement les valeurs.															
<code>vk clipboard options value and formatting</code>	Copie les valeurs et les formats.															

Les options de collage définies dans les [options de workbook](#) sont prises en compte.

La commande retourne un objet qui contient les données copiées.

Exemple

Cet exemple de code copie d'abord le contenu, valeurs, formats et formules d'une plage dans un objet puis les colle dans une autre plage :

```

var $originRange; $targetRange; $dataObject; $options : Object

$originRange:=VP Cells("ViewProArea"; 0; 0; 2; 5)

$options:=New object
$options.copy:=True
$options.copyOptions:=vk clipboard options all

$dataObject:=VP Copy to object($originRange; $options)

$targetRange:=VP Cell("ViewProArea"; 4; 0)
VP PASTE FROM OBJECT($targetRange; $dataObject; vk clipboard options all)
  
```

Voir aussi

[VP PASTE FROM OBJECT](#)

[VP MOVE CELLS](#)

[VP Get workbook options](#)

[VP SET WORKBOOK OPTIONS](#)

D

VP DELETE COLUMNS

[VP DELETE COLUMNS \(*rangeObj* : Object \)](#)

Paramètres	Type		Description
rangeObj	Object	->	Objet plage

Description

La commande `VP DELETE COLUMNS` supprime les colonnes de *rangeObj*.

Dans `rangeObj`, passez un objet contenant les colonnes à supprimer. Si la plage qui est passée contient :

- des lignes et des colonnes, seules les colonnes sont supprimées.
 - uniquement des lignes, la commande ne fait rien.

La numérotation démarre à 0.

Exemple

Pour supprimer les colonnes sélectionnées par l'utilisateur (colonnes B, C et D de l'image ci-dessous) :

saisissez le code suivant :

```
VP DELETE COLUMNS(VP Get selection("ViewProArea"))
```

Voir aussi

VP All

VP Cells

VP Column

VP DELETE ROWS

VP DELETE ROWS (*rangeObj* : Object)

Paramètres	Type		Description
rangeObj	Object	->	Objet plage

Description

La commande `VP DELETE ROWS` supprime les lignes de *rangeObj*.

Dans `rangeObj`, passez un objet contenant les lignes à supprimer. Si la plage qui est passée contient :

- des lignes et des colonnes, seules les lignes sont supprimées.
 - uniquement des colonnes, la commande ne fait rien.

La numérotation démarre à 0.

Exemple

Pour supprimer les lignes sélectionnées par l'utilisateur (lignes 1, 2 et 3 de l'image ci-dessous) :

	A	B	C	D	E	F	G	H	I
1									
2									
3									
4	The	quick	brown	fox	jumped	over	the	lazy	dog

saisissez le code suivant :

```
VP DELETE ROWS(VP Get selection("ViewProArea"))
```

Voir aussi

[VP All](#)

[VP Cells](#)

[VP Column](#)

E

VP EXPORT DOCUMENT

VP EXPORT DOCUMENT (*vpAreaName* : Text ; *filePath* : Text {; *paramObj* : Object})

Paramètres	Type		Description
<i>vpAreaName</i>	Text	->	Nom d'objet formulaire zone 4D View Pro
<i>filePath</i>	Text	->	Chemin d'accès du document
<i>paramObj</i>	Object	->	Options d'export

Description

La commande **VP EXPORT DOCUMENT** exporte l'objet 4D View Pro attaché à la zone 4D View Pro *vpAreaName* vers un document sur disque en fonction des paramètres *filePath* et *paramObj*.

Dans *vpAreaName*, passez le nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est retournée.

Dans *filePath*, passez le chemin de destination et le nom du document que vous souhaitez exporter. Si vous ne spécifiez pas de chemin, le document sera sauvegardé au même niveau que le dossier Projet.

Vous pouvez préciser le format du document en incluant une extension après son nom :

- 4D View Pro ("4vp")
- Microsoft Excel ("xlsx")
- PDF ("pdf")
- CSV ("txt", or "csv")

Si l'extension n'est pas incluse mais que le format est spécifié dans *paramObj*, le fichier exporté aura l'extension qui correspond au format, à l'exception du format CSV (aucune extension n'est ajoutée dans ce cas).

Le paramètre optionnel *paramObj* vous permet de définir plusieurs propriétés de l'objet 4D View Pro exporté et de lancer une méthode callback (ou méthode de rétro-appel) lorsque l'export est terminé.

Propriété	Type	Description																		
format	Texte	(optionnel) Désigne le format du fichier exporté : ".4vp" (par défaut), ".xlsx", ".pdf", ".csv" ou ".txt". Vous pouvez passer une des constantes suivantes : <ul style="list-style-type: none"> • <code>vk 4D View Pro format</code> • <code>vk csv format</code> • <code>vk MS Excel format</code> • <code>vk pdf format</code> 4D ajoute l'extension appropriée au nom du fichier si nécessaire. Si le format défini ne correspond pas à l'extension dans le <code>filePath</code> , il sera ajouté à la fin du <code>filePath</code> . Si un format n'est pas précisé et si aucune extension n'est fournie dans <code>filePath</code> , le format de fichier par défaut est utilisé.																		
password	Texte	Microsoft Excel uniquement (optionnel) - Mot de passe utilisé pour protéger le document MS Excel																		
formula	object	Méthode callback à lancer lorsque l'export est terminé. L'utilisation d'une méthode callback est nécessaire lorsque l'export est asynchrone (ce qui est le cas pour les formats PDF et Excel) si vous avez besoin d'un code à exécuter après l'export. La méthode callback doit être utilisée avec la commande Formula (voir ci-dessous pour plus d'informations).																		
valuesOnly	boolean	Précise que seules les valeurs issues de formules (le cas échéant) seront exportées.																		
includeFormatInfo	boolean	True (default) to include formatting information, false otherwise. Les informations de formatage sont utiles dans certains cas, par exemple pour un export en SVG. On the other hand, setting this property to False allows reducing export time.																		
includeBindingSource	Booléen	4DVP uniquement. True (par défaut) pour exporter les valeurs du contexte de données courant en tant que valeurs de cellule dans le document exporté (les contextes de données eux-mêmes ne sont pas exportés). Sinon Faux. La liaison de cellule est toujours exportée. Pour la gestion des contextes de données et des fusions de cellules, voir VP SET DATA CONTEXT et VP SET BINDING PATH .																		
sheetIndex	number	PDF uniquement (optionnel) - Numéro de la feuille à exporter (débute à 0). - 2=toutes les feuilles visibles (par défaut), -1=feuille courante uniquement																		
pdfOptions	object	PDF uniquement (optionnel) - Options pour l'export en PDF <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Propriété</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>creator</td> <td>Texte</td> <td>nom de l'application qui a créé le document original à partir duquel il a été converti.</td> </tr> <tr> <td>title</td> <td>Texte</td> <td>titre du document.</td> </tr> <tr> <td>author</td> <td>Texte</td> <td>nom de la personne ayant créé ce document.</td> </tr> <tr> <td>keywords</td> <td>Texte</td> <td>mots-clés associés au document.</td> </tr> <tr> <td>subject</td> <td>Texte</td> <td>sujet du document.</td> </tr> </tbody> </table>	Propriété	Type	Description	creator	Texte	nom de l'application qui a créé le document original à partir duquel il a été converti.	title	Texte	titre du document.	author	Texte	nom de la personne ayant créé ce document.	keywords	Texte	mots-clés associés au document.	subject	Texte	sujet du document.
Propriété	Type	Description																		
creator	Texte	nom de l'application qui a créé le document original à partir duquel il a été converti.																		
title	Texte	titre du document.																		
author	Texte	nom de la personne ayant créé ce document.																		
keywords	Texte	mots-clés associés au document.																		
subject	Texte	sujet du document.																		
csvOptions	object	CSV uniquement (optionnel) - Options pour l'export en CSV <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Propriété</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>range</td> <td>object</td> <td>Objet plage de toutes les cellules</td> </tr> <tr> <td>rowDelimiter</td> <td>Texte</td> <td>Délimiteur de ligne. Par défaut : "%r%n"</td> </tr> <tr> <td>columnDelimiter</td> <td>Texte</td> <td>Délimiteur de colonne. Par défaut : ","</td> </tr> </tbody> </table>	Propriété	Type	Description	range	object	Objet plage de toutes les cellules	rowDelimiter	Texte	Délimiteur de ligne. Par défaut : "%r%n"	columnDelimiter	Texte	Délimiteur de colonne. Par défaut : ","						
Propriété	Type	Description																		
range	object	Objet plage de toutes les cellules																		
rowDelimiter	Texte	Délimiteur de ligne. Par défaut : "%r%n"																		
columnDelimiter	Texte	Délimiteur de colonne. Par défaut : ","																		
<customProperty>	any	Propriété personnalisée qui sera disponible via le paramètre \$3 dans la méthode de callback.																		

**Notes sur le format Excel **:

- Lors de l'export d'un document 4D View Pro en un fichier au format Microsoft Excel, certains paramètres peuvent être perdus. Par exemple, les méthodes et formules 4D ne sont pas prises en charge par Excel. Vous pouvez vérifier les autres paramètres avec [cette liste proposée par GrapeCity](#).
- L'export dans ce format est exécuté de manière asynchrone, utilisez la propriété `formula` de `paramObj` pour le code à exécuter après l'export.

Notes sur le format PDF :

- Lors de l'export d'un document 4D View Pro en un fichier au format PDF, les polices utilisées dans le document sont automatiquement intégrées dans le fichier PDF. Seules les polices OpenType (fichiers .OTF ou .TTF) ayant une table Unicode peuvent être intégrées. Si aucun fichier de polices valide n'est trouvé pour une police, une police par défaut est utilisée à sa place.
- L'export dans ce format est exécuté de manière asynchrone, utilisez la propriété `formula` de `paramObj` pour le code à exécuter après l'export.

Notes sur le format CSV :

- Lors de l'export d'un document 4D View Pro en un fichier au format PDF, certains paramètres peuvent être perdus, car seuls le texte et les valeurs sont sauvegardés.
- Toutes les valeurs sont enregistrées sous la forme de chaînes entre guillemets. Pour plus d'informations sur les valeurs séparées par des délimiteurs, consultez [cet article de Wikipedia](#).

Une fois que l'export est terminé, `VP EXPORT DOCUMENT` exécute automatiquement la méthode définie dans la propriété `formula` de `paramObj`, le cas échéant.

Passer une méthode callback (formula)

Lorsque vous passez le paramètre optionnel `paramObj`, la commande `VP EXPORT DOCUMENT` vous permet d'utiliser la commande `Formula` pour appeler une méthode 4D qui sera exécutée une fois que l'export sera terminé. La méthode callback recevra les valeurs suivantes dans des variables locales :

Variable		Type	Description
\$1		Texte	Nom de l'objet 4D View Pro
\$2		Texte	Chemin d'accès de l'objet 4D View Pro exporté
\$3		object	Référence au <code>paramObj</code> de la commande
\$4		object	Objet retourné par la méthode avec un message de statut
	.success	boolean	Vrai si l'export est réussi, Faux sinon.
	.errorCode	entier	Code d'erreur. Peut être retourné par 4D ou JavaScript.
	.errorMessage	Texte	Message d'erreur. Peut être retourné par 4D ou JavaScript.

Exemple 1

Vous souhaitez exporter le contenu de la zone "VPArea" vers un document 4D View Pro sur le disque :

```
var $docPath: Text  
  
$docPath:="C:\\Bases\\ViewProDocs\\MyExport.4VP"  
VP EXPORT DOCUMENT("VPArea";$docPath)  
//MyExport.4VP est sauvegardé sur votre disque
```

Exemple 2

Vous souhaitez exporter la feuille courante au format PDF :

```
var $params: Object  
$params:=New object  
$params.format:=vk pdf format  
$params.sheetIndex:=-1  
$params.pdfOptions:=New object("title";"Annual Report";"author";Current user)  
VP EXPORT DOCUMENT("VPArea";"report.pdf";$params)
```

Exemple 3

Vous souhaitez exporter un document 4D View Pro au format ".xlsx" et appeler une méthode qui lance Microsoft Excel avec le document ouvert une fois que l'export est terminé :

```
$params:=New object  
$params.formula:=Formula(AfterExport)  
$params.format:=vp MS Excel format //".xlsx"  
$params.valuesOnly:=True
```

Méthode *AfterExport* :

```
C_TEXT($1;$2)
C_OBJECT($3;$4)
$areaName:=$1
$filePath:=$2
$params:=$3
$status:=$4

If($status.success=False)
    ALERT($status.errorMessage)
Else
    LAUNCH EXTERNAL PROCESS("C:\\\\Program Files\\\\Microsoft Office\\\\Office15\\\\excel "+$filePath)
End if
```

Exemple 4

Vous souhaitez exporter la feuille courante dans un fichier .txt avec des valeurs séparées par des " | " :

The screenshot shows the Microsoft Excel ribbon with the 'HOME' tab selected. The 'Clipboard' group includes 'Paste' (with dropdown), 'Undo', and 'Clipboard' buttons. The 'Fonts' group includes 'Calibri' (font dropdown), '11' (size dropdown), bold ('B'), italic ('I'), underline ('U'), strikethrough ('D'), font color ('A'), alignment ('Alignment' dropdown), and number format ('Numbers' dropdown). The 'Number' group includes 'Cell Type' (dropdown), 'Styles' (dropdown), 'Cells' (dropdown), and 'Editing' (dropdown). The formula bar shows cell A1 selected, and the status bar shows the text 'Clark'.

```

var $params : Object
$params:=New object
$params.range:=VP Cells("ViewProArea";0;0;2;5)
$params.rowDelimiter:="\n"
$params.columnDelimiter:="|"
VP EXPORT DOCUMENT("ViewProArea";"c:\\tmp\\data.txt";New object("format";vk csv format;"csvOptions";$par

```

Here's the result:

```

"Clark"|"Kent"
"Bruce"|"Wayne"
"Barry"|"Allen"
"Peter"|"Parker"
"Tony"|"Stark"

```

Voir aussi

[VP Convert to picture](#)
[VP Export to object](#)
[VP Column](#)
[VP Print](#)

VP Export to object

VP Export to object (*vpAreaName* : Text {; *options* : Object}) : Object

Paramètres	Type	Description
<i>vpAreaName</i>	Text	-> Nom d'objet formulaire zone 4D View Pro
<i>options</i>	Object	-> Options d'export
Résultat	Object	<- Objet 4D View Pro

Description

La commande `VP Export to object` retourne l'objet 4D View Pro associé à la zone 4D View Pro *vpAreaName*. Vous pouvez utiliser cette commande par exemple pour stocker la zone 4D View Pro dans un champ objet de la base de données 4D.

Dans *vpAreaName*, passez le nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est renvoyée.

Dans le paramètre *options*, vous pouvez passer l'option d'export suivante, si nécessaire :

Propriété	Type	Description
<i>includeFormatInfo</i>	Booléen	Vrai pour inclure les informations de formatage, sinon faux (vrai par défaut). Les informations de formatage sont utiles dans certains cas, par exemple pour un export en SVG. D'un autre côté, mettre cette propriété à faux permet de réduire la durée de l'export.
<i>includeBindingSource</i>	Booléen	True (par défaut) pour exporter les valeurs du contexte de données courant en tant que valeurs de cellule dans l'objet exporté (les contextes de données eux-mêmes ne sont pas exportés). Sinon Faux. La liaison de cellule est toujours exportée.

Pour plus d'informations sur les objets 4D View Pro, veuillez vous référer au paragraphe [objet 4D View Pro](#).

Exemple 1

Vous souhaitez lire la propriété "version" de la zone 4D View Pro courante :

```

var $vpAreaObj : Object
var $vpVersion : Number
$vpAreaObj:=VP Export to object("vpArea")
// $vpVersion:=OB Get($vpAreaObj;"version")
$vpVersion:=$vpAreaObj.version

```

Exemple 2

Vous souhaitez exporter la zone, en excluant les informations de formatage :

```

var $vpObj : Object
$vpObj:=VP Export to object("vpArea";New object("includeFormatInfo";False))

```

Voir aussi

[VP Convert to picture](#)
[VP EXPORT DOCUMENT](#)
[VP IMPORT FROM OBJECT](#)

F

VP Find

VP Find (*rangeObj* : Object ; *searchValue* : Text) : Object
 VP Find (*rangeObj* : Object ; *searchValue* : Text ; *searchCondition* : Object }) : Object
 VP Find (*rangeObj* : Object ; *searchValue* : Text ; *searchCondition* : Object ; *replaceValue* : Text) : Object

Paramètres	Type		Description
<i>rangeObj</i>	Object	->	Objet plage
<i>searchValue</i>	Text	->	Valeur de recherche
<i>searchCondition</i>	Object	->	Objet contenant la/les condition(s) de recherche
<i>replaceValue</i>	Text	->	Valeur de remplacement
Résultat	Object	<-	Objet plage

Description

La commande `VP Find` recherche le paramètre *rangeObj* pour *searchValue*. Des paramètres facultatifs peuvent être utilisés pour affiner la recherche et/ou remplacer les résultats trouvés.

Dans le paramètre *rangeObj*, passez un objet contenant une plage à rechercher.

Le paramètre *searchValue* vous permet de passer le texte à rechercher dans le paramètre *rangeObj*.

Vous pouvez passer le paramètre optionnel *searchCondition* pour préciser le fonctionnement de la recherche. Les propriétés suivantes sont prises en charge :

Propriété	Type	Description
<i>afterColumn</i>	Integer	Le numéro de la colonne située juste avant la colonne de départ de la recherche. Si <i>rangeObj</i> est une plage combinée, le numéro de colonne indiqué doit provenir de la première plage. Valeur par défaut : -1 (début de <i>rangeObj</i>)
<i>afterRow</i>	Integer	Le numéro de la ligne située juste avant la ligne de départ de la recherche. Si <i>rangeObj</i> est une plage combinée, le numéro de ligne indiqué doit provenir de la première plage. Valeur par défaut : -1 (début de <i>rangeObj</i>)
<i>all</i>	Booléen	• True -Toutes les cellules de <i>rangeObj</i> correspondant à <i>searchValue</i> sont retournées

Propriété	Type	Description	(valeur par défaut) Seule la première cellule de <i>rangeObj</i> correspondant à <i>searchValue</i> est retournée
flags	Integer	<p><code>vk find flag exact match</code></p> <p><code>vk find flag ignore case</code></p> <p><code>vk find flag none</code></p> <p><code>vk find flag use wild cards</code></p>	<p>Tout le contenu de la cellule doit entièrement correspondre à la valeur de recherche</p> <p>Les majuscules et les minuscules sont considérées comme identiques. Ex : "a" est considérée comme identique à "A".</p> <p>Aucun indicateur de recherche n'est pris en compte (par défaut)</p> <p>Les caractères génériques (*,?) peuvent être utilisés dans la chaîne de recherche. Les caractères joker peuvent être utilisés dans n'importe quelle comparaison de chaînes pour correspondre à n'importe quel nombre de caractères :</p> <ul style="list-style-type: none"> • * - Pour le caractère zéro ou plusieurs caractères (par exemple, rechercher "bl*" peut donner comme résultat "bl", "black", "blue", et "blob") • ? pour un seul caractère (par exemple, rechercher "h?t" peut donner comme résultat "hot", "hat", et "hit")
		Ces marqueurs peuvent être combinés. Par exemple :	
		<code>\$search.flags:=vk find flag use wild cards+vk find flag ignore case</code>	
order	Integer	<p><code>vk find order by columns</code></p> <p><code>vk find order by rows</code></p>	<p>La recherche est effectuée par colonnes. Chaque ligne d'une colonne est recherchée avant que la recherche ne passe à la colonne suivante.</p> <p>La recherche est effectuée par lignes. Chaque colonne d'une ligne est recherchée avant que la recherche ne passe à la colonne suivante (par défaut)</p>
target	Integer	<p><code>vk find target formula</code></p> <p><code>vk find target tag</code></p> <p><code>vk find target text</code></p>	<p>La recherche est effectuée dans la formule de la cellule</p> <p>La recherche est effectuée dans la balise de la cellule</p> <p>La recherche est effectuée dans le texte de la cellule (par défaut)</p>
		Ces marqueurs peuvent être combinés. Par exemple :	
		<code>\$search.target:=vk find target formula+vk find target text</code>	

Dans le paramètre optionnel *replaceValue*, vous pouvez passer du texte pour remplacer toute instance du texte dans la *searchValue* trouvée dans *rangeObj*.

Objet retourné

La fonction retourne un objet de plage décrivant chaque valeur de recherche trouvée ou remplacée. Un objet de plage vide est retourné si aucun résultat n'est trouvé.

Exemple 1

Pour trouver la première cellule contenant le mot "Total" :

```
var $range;$result : Object  
  
$range:=VP All("ViewProArea")  
  
$result:=VP Find($range;"Total")
```

Exemple 2

Pour trouver "Total" et le remplacer par "Grand Total" :

```
var $range;$condition;$result : Object  
  
$range:=VP All("ViewProArea")  
  
$condition:=New object  
$condition.target:=vk find target text  
$condition.all:=True //Search entire document  
$condition.flags:=vk find flag exact match  
  
// Replace the cells containing only 'Total' in the current sheet with "Grand Total"  
$result:=VP Find($range;"Total";$condition;"Grand Total")  
  
// Check for empty range object  
If($result.ranges.length=0)  
    ALERT("No result found")  
Else  
    ALERT($result.ranges.length+" results found")  
End if
```

VP FLUSH COMMANDS

VP FLUSH COMMANDS (*vpAreaName* : Text)

Paramètres	Type		Description
vpAreaName	Text	->	Nom d'objet formulaire zone 4D View Pro

Description

La commande `VP FLUSH COMMANDS` exécute immédiatement les commandes empilées et vide le buffer de commandes.

Dans *vpAreaName*, passez le nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est retournée.

Afin d'améliorer la performance et de réduire le nombre de requêtes envoyées, les commandes 4D View Pro appelées par le développeur sont stockées temporairement dans un buffer de commandes. Lorsqu'elle est appelée, la commande `VP FLUSH COMMANDS` exécute les commandes en lot au moment de quitter la méthode et vide le buffer de commandes.

Exemple

Vous souhaitez tracer l'exécution des commandes et vider le buffer :

```

VP SET TEXT VALUE(VP Cell("ViewProArea1";10;1);"INVOICE")
VP SET TEXT VALUE(VP Cell("ViewProArea1";10;2);"Invoice date: ")
VP SET TEXT VALUE(VP Cell("ViewProArea1";10;3);"Due date: ")

VP FLUSH COMMANDS(("ViewProArea1")
TRACE

```

VP Font to object

VP Font to object (*font* : Text) : Object

Paramètres	Type		Description
font	Text	->	Chaîne raccourcie pour la police (shorthand)

Description

La commande utilitaire `VP Font to object` retourne un objet d'une chaîne raccourcie pour la police (shorthand). Cet objet peut être utilisé pour définir ou lire les propriétés de la police via la notation objet.

Dans le paramètre *font*, passez une chaîne raccourcie pour la police pour indiquer les différentes propriétés d'une police (ex : "12 pt Arial"). Cliquez [ici](#) pour en savoir plus sur les chaînes raccourcies pour la police.

L'objet retourné contient des attributs de police définis comme propriétés. Pour plus d'informations sur les propriétés disponibles, veuillez vous reporter à la commande [VP Object to font](#).

Exemple 1

Le code suivant :

```
$font:=VP Font to object("16pt arial")
```

retournera l'objet \$font suivant :

```
{
  family:arial
  size:16pt
}
```

Exemple 2

Voir l'exemple de [VP Object to font](#).

Voir aussi

[4D View Pro Style Objects and Style Sheets](#)
[VP Object to font](#)
[VP SET CELL STYLE](#)
[VP SET DEFAULT STYLE](#)

G

VP Get active cell

VP Get active cell (*vpAreaName* : Text { ; *sheet* : Integer }) : Object

Paramètres	Type		Description
vpAreaName	Text	->	Nom d'objet formulaire zone 4D View Pro
sheet	Integer	->	Indice de la feuille (feuille courante si omis)
Résultat	Object	<-	Objet plage d'une seule cellule

Description

La commande `VP Get active cell` retourne un nouvel objet plage référençant la cellule active, dans laquelle de nouvelles données seront saisies.

Dans `vpAreaName`, passez le nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est retournée.

Dans le paramètre optionnel `sheet`, vous pouvez désigner une feuille spécifique dans laquelle sera définie la plage (la numérotation commence à zéro). Si le paramètre est omis ou si vous passez `vk current sheet`, la feuille courante est utilisée.

Exemple

A	B	C	D	E	F	G
1						
2	Month	Area	Mr. Smith	Ms. Johnson	Ms. Williams	Mr. Jones
3	January	South	1898	1857	1651	1448
4		East	4859	4857	2548	4876
5		North	2458	1524	6150	4987
6		West	5787	1580	3975	4878
7	Total		15002	9818	14324	16189
8	February	South	6668	4374	17495	9999
9		East	5955	1677	7944	9400
10		North	1000	6722	2195	2777
11		West	6896	8355	7195	2058
12	Total		20519	21128	34829	24234
13	March	South	2577	2000	6185	2704
14		East	4859	4857	2548	4876
15		North	2458	1524	6150	4987
16		West	5787	1580	3975	4878
17	Total		15621	9961	18858	17445

Le code suivant récupérera les coordonnées de la cellule active :

```
$activeCell:=VP Get active cell("myVPArea")

//returns a range object containing:
//$activeCell.ranges[0].column=3
//$activeCell.ranges[0].row=4
//$activeCell.ranges[0].sheet=0
```

Voir aussi

[VP ADD SELECTION](#)
[VP Get selection](#)
[VP RESET SELECTION](#)
[VP SET ACTIVE CELL](#)
[VP SET SELECTION](#)
[VP SHOW CELL](#)

VP Get binding path

► Historique

`VP Get binding path (rangeObj : Object) : Text`

Paramètres	Type		Description
rangeObj	Object	->	Objet plage
Résultat	Text	<-	Name of the attribute bound to the cell

Description

The `VP Get binding path` command returns the name of the attribute bound to the cell specified in `rangeObj`.

In `rangeObj`, pass an object that is either a cell range or a combined range of cells. A noter que :

- If `rangeObj` is a range with several cells, the command returns the attribute name linked to the first cell in the range.
- If `rangeObj` contains several ranges of cells, the command returns the attribute name linked to the first cell of the first range.

Exemple

```
var $p; $options : Object
var $myAttribute : Text

$p:=New object
$p.firstName:="Freehafer"
$p.lastName:="Nancy"

VP SET DATA CONTEXT("ViewProArea"; $p)

VP SET BINDING PATH(VP Cell("ViewProArea"; 0; 0); "firstName")
VP SET BINDING PATH(VP Cell("ViewProArea"; 1; 0); "lastName")

$myAttribute:=VP Get binding path(VP Cell("ViewProArea"; 1; 0)) // "lastName"
```

Voir aussi

[VP SET BINDING PATH](#)
[VP Get data context](#)
[VP SET DATA CONTEXT](#)

VP Get cell style

`VP Get cell style (rangeObj : Object) : Object`

Paramètres	Type		Description
rangeObj	Object	->	Objet plage
Résultat	Object	<-	Objet style

Description

La commande `VP Get cell style` retourne un [objet style](#) pour la première cellule de `rangeObj`.

Dans `rangeObj`, passez une plage contenant le style à récupérer.

- Si `rangeObj` contient une plage cellule, le style de la cellule est retourné.
- Si `rangeObj` contient une plage qui n'est pas une plage cellule, le style de la première cellule de la plage est retourné.
- Si `rangeObj` contient plusieurs plages, seul le style de la première cellule de la première plage est retourné.

Exemple

Pour obtenir les détails concernant le style de la cellule sélectionnée (B2) :

	A	B	C
1			
2		H e l l o	
3		W o r l d	

Le code suivant :

```
$cellStyle:=VP Get cell style(VP Get selection("myDoc"))
```

... retournera cet objet :

```
{
  "backColor":"Azure",
  "borderBottom": {
    "color": "#800080",
    "style": 5
  },
  "font": "8pt Arial",
  "foreColor": "red",
  "hAlign": 1,
  "isVerticalText": "true",
  "vAlign": 0
}
```

Voir aussi

[VP GET DEFAULT STYLE](#)
[VP SET CELL STYLE](#)

VP Get column attributes

VP Get column attributes (*rangeObj* : Object) : Collection

Paramètres	Type		Description
rangeObj	Object	->	Objet plage
Résultat	Collection	<-	Collection de propriétés de colonnes

Description

La commande `VP Get column attributes` retourne une collection de propriétés pour les colonnes de *rangeObj*.

Dans *rangeObj*, passez un objet contenant une plage de colonnes dont les attributs seront récupérés.

La collection renvoyée contient les propriétés des colonnes, qu'elles aient ou non été définies par la commande [VP SET COLUMN ATTRIBUTES](#).

Exemple

Le code suivant :

```

C_OBJECT($range)
C_COLLECTION($attr)

$range:=VP Column("ViewProArea";1;2)
$attr:=VP Get column attributes($range)

```

... retournera une collection d'attributs de la plage donnée :

length	2
\$attr[0]	{"width":150,"pageBreak":false,"visible":true,"resizable":false,"header":"Hello World"}
\$attr[0].header	"Hello World"
\$attr[0].pageBreak	False
\$attr[0].resizable	False
\$attr[0].visible	True
\$attr[0].width	150
\$attr[1]	{"width":62,"pageBreak":false,"visible":true,"resizable":true,"header":"C"}
\$attr[1].header	"C"
\$attr[1].pageBreak	False
\$attr[1].resizable	True
\$attr[1].visible	True
\$attr[1].width	62

Voir aussi

[VP Get row attributes](#)
[VP SET COLUMN ATTRIBUTES](#)
[VP SET ROW ATTRIBUTES](#)

VP Get column count

VP Get column count (*vpAreaName* : Text { ; *sheet* : Integer }) : Integer

Paramètres	Type		Description
<i>vpAreaName</i>	Text	->	Nom de la zone 4D View Pro dans le formulaire
<i>sheet</i>	Integer	->	Indice de la feuille (feuille courante si omis)
Résultat	Integer	<-	Nombre total de colonnes

Description

La commande `VP Get column count` retourne le nombre total de colonnes de la feuille (*sheet*) désignée.

Dans *vpAreaName*, passez la propriété du nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est renvoyée.

Vous pouvez définir l'emplacement du nombre de colonnes dans le paramètre optionnel *sheet* à l'aide de l'indice de la feuille (la numérotation démarre à zéro). Si le paramètre est omis ou si vous passez `vk current sheet`, la feuille courante est utilisée.

Exemple

Le code suivant retourne le nombre de colonnes dans la zone 4D View Pro :

```

C_Integer($colCount)
$colCount:=VP Get column count("ViewProarea")

```

Voir aussi

[VP Get row count](#)

[VP SET COLUMN COUNT](#)

[VP SET ROW COUNT](#)

VP Get current sheet

VP Get current sheet (*vpAreaName* : Text)

Paramètres	Type		Description
<i>vpAreaName</i>	Text	->	Nom d'objet formulaire zone 4D View Pro
Résultat	Integer	<-	Indice de la feuille courante

Description

La commande `VP Get current sheet` retourne l'indice de la feuille courante dans *vpAreaName*. La feuille courante est la feuille sélectionnée dans le document.

Dans *vpAreaName*, passez le nom de la zone 4D View Pro.

La numérotation démarre à 0.

Exemple

Lorsque la troisième feuille est sélectionnée :



La commande retourne 2 :

```
$index:=VP Get current sheet("ViewProArea")
```

Voir aussi

[VP SET CURRENT SHEET](#)

VP Get data context

► Historique

[VP Get data context \(*vpAreaName* : Text {; *sheetIndex* : Integer } \) : Object](#)

[VP Get data context \(*vpAreaName* : Text {; *sheetIndex* : Integer } \) : Collection](#)

Paramètres	Type		Description
<i>vpAreaName</i>	Object	->	Nom d'objet formulaire zone 4D View Pro
<i>sheetIndex</i>	Integer	->	Index of the sheet to get the data context from
Résultat	Object Collection	<-	Data context

Description

The `VP Get data context` command returns the current data context of a worksheet. The returned context includes any modifications made to the contents of the data context.

In *sheetIndex*, pass the index of the sheet to get the data context from. If no index is passed, the command returns the data context of the current worksheet. If there is no context for the worksheet, the command returns `Null`.

The function returns an object or a collection depending on the type of data context set with [VP SET DATA CONTEXT](#).

Exemple

To get the data context bound to the following cells:

	0	1	2	3
1	Freehafer	Nancy		
2				
3				

```
var $dataContext : Object  
  
$dataContext:=VP Get data context("ViewProArea") // {firstName:Freehafer,lastName:Nancy}
```

Voir aussi

[VP SET DATA CONTEXT](#)

[VP Get binding path](#)

[VP SET BINDING PATH](#)

VP Get default style

VP Get default style (*vpAreaName* : Text { ; *sheet* : Integer }) : Integer

Paramètres	Type		Description
<i>vpAreaName</i>	Text	->	Nom de la zone 4D View Pro dans le formulaire
<i>sheet</i>	Integer	->	Indice de la feuille (feuille courante si omis)
Résultat	Integer	<-	Nombre total de colonnes

Description

La commande `VP Get default style` retourne un objet feuille de style par défaut pour une feuille. L'objet retourné contient des propriétés basiques de rendu d'un document ainsi que les propriétés du style par défaut (le cas échéant) définies préalablement à l'aide de la méthode [VP SET DEFAULT STYLE](#). Pour plus d'informations sur les propriétés de style, consultez [Objets style et feuilles de style](#).

Dans *vpAreaName*, passez la propriété du nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est retournée.

Vous pouvez définir l'emplacement du nombre de colonnes dans le paramètre optionnel *sheet* à l'aide de l'indice de la feuille (la numérotation démarre à zéro). Si le paramètre est omis ou si vous passez `vk current sheet`, la feuille courante est utilisée.

Exemple

Pour lire les détails du style par défaut de ce document :

	A	B	C
1			
2		Hello World!	
3			
4			
5			
6			
7			
8			

Le code suivant :

```
$defaultStyle:=VP Get default style("myDoc")
```

retournera les informations suivantes dans l'objet `$defaultStyle` :

```
{
    backColor:#E6E6FA,
    hAlign:0,
    vAlign:0,
    font:12pt papyrus
}
```

Voir aussi

[VP Get cell style](#)

[VP SET DEFAULT STYLE](#)

VP Get formula

VP Get formula (*rangeObj* : Object) : Text

Paramètres	Type		Description
<i>rangeObj</i>	Object	->	Objet plage
Résultat	Text	<-	Formula

Description

La commande `VP Get formula` récupère la formule depuis une plage de cellules désignée.

Dans *rangeObj*, passez la plage dont vous souhaitez récupérer la formule. Si *rangeObj* désigne plusieurs cellules ou plusieurs plages, la formule de la première cellule est renvoyée. Si *rangeObj* est une cellule qui ne contient pas de formule, la commande retourne une chaîne vide.

Exemple

```
//fixer une formule
VP SET FORMULA(VP Cell("ViewProArea";5;2);"SUM($A$1:$C$10)")

$result:=VP Get formula(VP Cell("ViewProArea";5;2)) // $result="SUM($A$1:$C$10)"
```

Voir aussi

[VP Get formulas](#)

[VP SET FORMULA](#)

[VP SET ROW COUNT](#)

VP Get formula by name

VP Get formula by name (*vpAreaName* : Text ; *name* : Text { ; *scope* : Number }) : Object

Paramètres	Type		Description
vpAreaName	Text	->	Nom d'objet formulaire zone 4D View Pro
name	Text	->	Nom de la plage nommée
scope	Nombre	->	"Scope" cible (par défaut=feuille courante)
Résultat	Text	<-	Définition de la formule nommée ou de la plage nommée

Description

La commande `VP Get formula by name` retourne la formule et le commentaire correspondant à la plage nommée ou à la formule nommée qui a été passée dans le paramètre `name`, ou retourne null si le nom n'existe pas dans le scope défini.

Dans `vpAreaName`, passez le nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est retournée.

Passez la plage nommée ou la formule nommée que vous souhaitez lire dans `name`. A noter que les plages nommées sont retournées sous forme de formules contenant des références absolues de cellules.

Le paramètre `scope` vous permet de définir l'espace de la zone dans lequel lire la formule, en passant l'indice de la feuille (la numérotation débute à 0) ou l'une des constantes suivantes :

- `vk current sheet`
- `vk workbook`

Objet retourné

L'objet retourné contient les propriétés suivantes :

Propriété	Type	Description
formula	Text	Texte de la formule correspondant à la formule nommée ou à la plage nommée. Pour les plages nommées, la formule est une séquence de coordonnées absolues.
comment	Text	Commentaire correspondant à la formule nommée ou à la plage nommée

Exemple

```
$range:=VP Cell("ViewProArea";0;0)
VP ADD RANGE NAME("Total1";$range)

$formula:=VP Get formula by name("ViewProArea";"Total1")
//$formula.formula=Sheet1!$A$1

$formula:=VP Get formula by name("ViewProArea";"Total")
//$formula=null (if not existing)
```

Voir aussi

[VP ADD FORMULA NAME](#)
[VP ADD RANGE NAME](#)
[VP Get names](#)

VP Get formulas

`VP Get formulas (rangeObj : Object) : Collection`

Paramètres	Type		Description
rangeObj	Object	->	Objet plage
Résultat	Collection	<-	Collection de valeurs d'une formule

Description

La commande `VP Get formulas` récupère les formules d'un `rangeObj`déterminé.

Dans `rangeObj`, passez une plage dont vous souhaitez récupérer les formules. Si `rangeObj` désigne plusieurs plages, la formule de la première plage est retournée. Si `rangeObj` ne contient pas de formules, la commande retourne une chaîne vide.

La collection retournée est bidimensionnelle :

- La collection de premier niveau contient des sous-collections de formules. Chaque sous-collection représente une ligne.
- Chaque sous-collection définit les valeurs des cellules de la ligne. Les valeurs sont des éléments textuels contenant les formules des cellules.

Exemple

Ous souhaitez récupérer les formules des colonnes Sum et Average de ce document :

A	B	C	D	E	F	G
1	Data			Sum	Average	
2	1	2	3	6	2	
3	4	5	6	15	5	
4	7	8	9	24	8.5	
5						
e						

Vous pouvez utiliser ce code :

```
$formulas:=VP Get formulas(VP Cells("ViewProArea";5;1;2;3))
//$formulas[0]=[Sum(B2:D2),Average(B2:D2)]
//$formulas[1]=[Sum(B3:D3),Average(B3:D3)]
//$formulas[2]=[Sum(B4:D4),Average(C4:D4)]
```

Voir aussi

[VP Get formula](#)

[VP Get values](#)

[VP SET FORMULAS](#)

[VP SET VALUES](#)

VP Get frozen panes

`VP Get frozen panes (vpAreaName : Text { ; sheet : Integer }) : Object`

Paramètres	Type		Description
vpAreaName	Text	->	Nom d'objet formulaire zone 4D View Pro
sheet	Integer	->	Indice de la feuille (feuille courante si omis)
Résultat	Object	<-	Objet contenant des informations sur les colonnes et lignes figées

Description

La commande `VP Get frozen panes` retourne un objet contenant des informations sur les lignes et colonnes figées de `vpAreaName`.

Dans `vpAreaName`, passez le nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est retournée.

Dans le paramètre optionnel `sheet`, vous pouvez désigner une feuille spécifique dans laquelle sera définie la plage (la numérotation commence à zéro). Si le paramètre est omis ou si vous passez `vk current sheet`, la feuille courante est utilisée.

Objet retourné

La commande retourne un objet décrivant les lignes et colonnes figées. Cet objet peut avoir les propriétés suivantes :

Propriété	Type	Description
columnCount	Integer	Le nombre de colonnes figées sur la gauche de la feuille
trailingColumnCount	Integer	Le nombre de colonnes figées sur la droite de la feuille
rowCount	Integer	Le nombre de lignes figées en haut de la feuille
trailingRowCount	Integer	Le nombre de lignes figées en bas de la feuille

Exemple

Vous souhaitez récupérer des informations sur le nombre de colonnes et de lignes figées :

```
var $panesObj : Object  
  
$panesObj:=VP Get frozen panes("ViewProArea")
```

L'objet retourné contient, par exemple :

Expression	Value
▀ \$paneObj	{"columnCount":3,"trailingColumnCount":0,"rowCount":1,"trailingRowCount":0}
▀ columnCount	3
▀ rowCount	1
▀ trailingColumnCount	0
▀ trailingRowCount	0

Voir aussi

[VP SET FROZEN PANES](#)

VP Get names

VP Get names (vpAreaName : Text { ; scope : Number }) : Collection

Paramètres	Type		Description
vpAreaName	Text	->	Nom d'objet formulaire zone 4D View Pro
scope	Nombre	->	"Scope" cible (par défaut=feuille courante)
Résultat	Collection	<-	Noms existant dans la zone définie

Description

La commande `VP Get names` retourne une collection de tous les "noms" définis dans la feuille courante ou dans la zone définie dans le paramètre `scope`.

Dans `vpAreaName`, passez le nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est retournée.

Le paramètre `scope` vous permet de définir l'espace de la zone dans lequel lire la formule, en passant l'indice de la feuille (la numérotation débute à 0) ou l'une des constantes suivantes :

- `vk current sheet`
- `vk workbook`

Collection retournée

La collection retournée contient un objet par nom. Les propriétés d'objets suivantes peuvent être retournées :

Propriété	Type	Description
result[].name	Text	nom de cellule ou de plage
result[].formula	Text	formula
result[].comment	Text	Commentaire associé au nom

Les propriétés disponibles dépendent du type d'élément nommé (cellule nommée, plage nommée ou formule nommée).

Exemple

```
var $list : Collection
$list:=VP Get names("ViewProArea";2) //noms de la 3e feuille
```

Voir aussi

[VP ADD FORMULA NAME](#)

[VP ADD RANGE NAME](#)

[VP Get formula by name](#)

[VP Name](#)

VP Get print info

VP Get print info (vpAreaName : Text { ; sheet : Integer }) : Object

Paramètres	Type		Description
vpAreaName	Text	->	Nom d'objet formulaire zone 4D View Pro
sheet	Integer	->	Indice de la feuille (feuille courante si omis)
Résultat	Object	<-	Objet contenant les informations d'impression

Description

La commande `VP Get print info` retourne un objet contenant les attributs d'impression de `vpAreaName`.

Passez le nom de la zone 4D View Pro dans `vpAreaName`. Si vous passez un nom inexistant, une erreur est retournée.

Dans le paramètre optionnel `sheet`, vous pouvez désigner une feuille spécifique (la numérotation commence à zéro) dont vous souhaitez retourner les attributs d'impression. Si le paramètre est omis ou si vous passez `vk current sheet`, la feuille courante est utilisée.

Exemple

Le code suivant :

```
$pinfo:=VP Get print info("ViewProArea")
```

Dès que la commande `VP SET CUSTOM FUNCTIONS` est appelée, les méthodes autorisées par [SET ALLOWED METHODS](#) sont ignorées dans la zone 4D View Pro.

```
{
bestFitColumns:false,
bestFitRows:false,
blackAndWhite:false,
centering:0,
columnEnd:8,
columnStart:0,
firstPageNumber:1,
fitPagesTall:1,
fitPagesWide:1,
footerCenter:"&BS.H.I.E.L.D. &A Sales Per Region",
footerCenterImage:, 
footerLeft:, 
footerLeftImage:, 
footerRight:"page &P of &N", 
footerRightImage:, 
headerCenter:, 
headerCenterImage:, 
headerLeft:&G, 
headerLeftImage:logo.png, 
headerRight:, 
headerRightImage:, 
margin:{top:75,bottom:75,left:70,right:70,header:30,footer:30}, 
orientation:2, 
pageOrder:0, 
pageRange:, 
paperSize:{width:850,height:1100,kind:1}, 
qualityFactor:2, 
repeatColumnEnd:-1, 
repeatColumnStart:-1, 
repeatRowEnd:-1, 
repeatRowStart:-1, 
rowEnd:24, 
rowStart:0, 
showBorder:false, 
showColumnHeader:0, 
showGridLine:false, 
showRowHeader:0, 
useMax:true, 
watermark:[], 
zoomFactor:1
}
```

Voir aussi

[4D View Pro Print Attributes](#)
[VP SET PRINT INFO](#)

VP Get row attributes

VP Get row attributes (rangeObj : Object) : Collection

Paramètres	Type		Description
rangeObj	Object	->	Objet plage
Résultat	Collection	<-	Collection de propriétés de lignes

Description

La commande `VP Get row attributes` retourne une collection de propriétés pour les lignes de `rangeObj`.

Dans `rangeObj`, passez un objet contenant une plage de lignes dont les attributs seront récupérés.

La collection renvoyée contient les propriétés des lignes, qu'elles aient été définies ou non par la méthode [VP SET ROW ATTRIBUTES](#).

Exemple

Le code suivant retourne une collection d'attributs de la plage donnée :

```
var $range : Object
var $attr : Collection

$range:=VP Column("ViewProArea";1;2)
$attr:=VP Get row attributes($range)
```

length	2
\$attr[0]	{"height":75,"pageBreak":false,"visible":true,"resizable":true,"header":"june"}
\$header	"june"
\$height	75
\$pageBreak	False
\$resizable	True
\$visible	True
\$attr[1]	{"height":20,"pageBreak":false,"visible":true,"resizable":true,"header":"3"}
\$header	"3"
\$height	20
\$pageBreak	False
\$resizable	True
\$visible	True

Voir aussi

[VP Get column attributes](#)
[VP SET COLUMN ATTRIBUTES](#)
[VP SET ROW ATTRIBUTES](#)

VP Get row count

VP Get row count (*vpAreaName* : Text {; *sheet* : Integer }) : Integer

Paramètres	Type		Description
<i>vpAreaName</i>	Text	->	Nom de la zone 4D View Pro dans le formulaire
<i>sheet</i>	Integer	->	Indice de la feuille (feuille courante si omis)
Résultat	Integer	<-	Nombre total de lignes

Description

La commande `VP Get row count` retourne le nombre total de lignes de la feuille (*sheet*) désignée.

Dans *vpAreaName*, passez la propriété du nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est renvoyée.

In the optional *sheet* parameter, you can designate a specific spreadsheet where the range will be defined (counting begins at 0). Si le paramètre est omis ou si vous passez `vk current sheet`, la feuille courante est utilisée.

Exemple

Le code suivant retourne le nombre de lignes dans la zone 4D View Pro :

```
var $rowCount : Integer
$rowCount:=VP Get row count("ViewProarea")
```

Voir aussi

[VP Get column count](#)
[VP SET COLUMN COUNT](#)
[VP SET ROW COUNT](#)

VP Get selection

VP Get selection (*vpAreaName* : Text {; *sheet* : Integer }) : Object

Paramètres	Type		Description
<i>vpAreaName</i>	Text	->	Nom de la zone 4D View Pro dans le formulaire
<i>sheet</i>	Integer	->	Indice de la feuille (feuille courante si omis)
Résultat	Object	<-	Objet plage de toutes les cellules

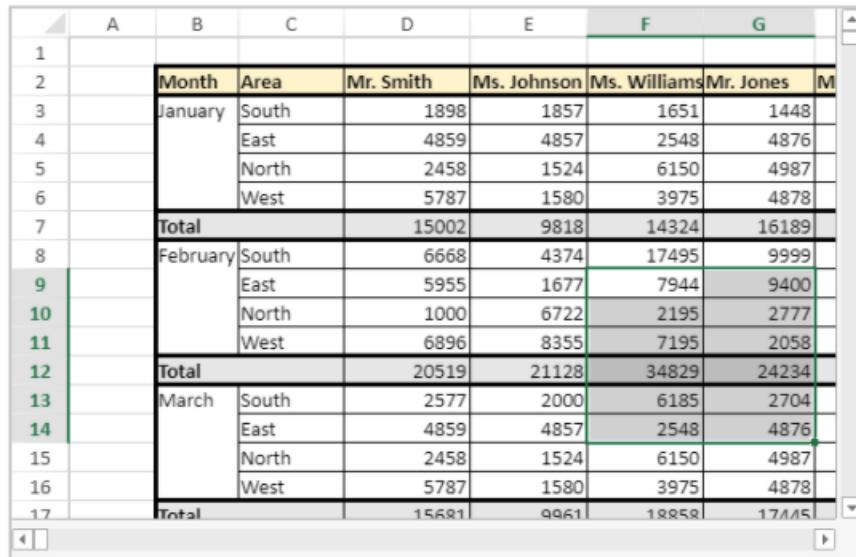
Description

La commande `VP Get selection` retourne un nouvel objet plage référençant les cellules courantes sélectionnées.

Dans *vpAreaName*, passez le nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est retournée.

Dans le paramètre optionnel *sheet*, vous pouvez désigner une feuille spécifique dans laquelle sera définie la plage (la numérotation commence à zéro). Si le paramètre est omis ou si vous passez `vk current sheet`, la feuille courante est utilisée.

Exemple



	A	B	C	D	E	F	G
1							
2		Month	Area	Mr. Smith	Ms. Johnson	Ms. Williams	Mr. Jones
3		January	South	1898	1857	1651	1448
4			East	4859	4857	2548	4876
5			North	2458	1524	6150	4987
6			West	5787	1580	3975	4878
7		Total		15002	9818	14324	16189
8		February	South	6668	4374	17495	9999
9			East	5955	1677	7944	9400
10			North	1000	6722	2195	2777
11			West	6896	8355	7195	2058
12		Total		20519	21128	34829	24234
13		March	South	2577	2000	6185	2704
14			East	4859	4857	2548	4876
15			North	2458	1524	6150	4987
16			West	5787	1580	3975	4878
17		Total		15681	9961	18852	17115

Le code suivant récupérera les coordonnées de la cellule active :

```
$currentSelection:=VP Get selection("myVPArea")

//retourne un objet plage contenant :
//$/currentSelection.ranges[0].column=5
//$/currentSelection.ranges[0].columnCount=2
//$/currentSelection.ranges[0].row=8
//$/currentSelection.ranges[0].rowCount=6
```

Voir aussi

[VP ADD SELECTION](#)
[VP Get active cell](#)
[VP SET ACTIVE CELL](#)
[VP SET SELECTION](#)
[VP SHOW CELL](#)

VP Get sheet count

VP Get sheet count (*vpAreaName* : Text) : Integer

Paramètres	Type		Description
<i>vpAreaName</i>	Text	->	Nom d'objet formulaire zone 4D View Pro
Résultat	Integer	<-	Nombre de feuilles

Description

La commande `VP Get sheet count` retourne le nombre de feuilles contenues dans le document chargé dans *vpAreaName*.

Dans *vpAreaName*, passez le nom de la zone 4D View Pro.

Exemple

Dans le document suivant :



Pour obtenir le nombre de feuilles et définir la feuille courante comme étant la dernière feuille :

```
$count:=VP Get sheet count("ViewProArea")
//définir la feuille courante comme étant la dernière feuille (l'indexation commence à 0) VP SET CURRE
```



Voir aussi

[VP Get sheet index](#)
[VP SET SHEET COUNT](#)

VP Get sheet index

VP Get sheet index (*vpAreaName* : Text ; *name* : Text) : Integer

Paramètres	Type		Description
<i>vpAreaName</i>	Text	->	Nom d'objet formulaire zone 4D View Pro
<i>name</i>	Text	->	Nom de la feuille
Résultat	Integer	<-	Indice de la feuille

Description

La commande `VP Get sheet index` retourne l'index d'une feuille en fonction de son nom dans *vpAreaName*.

Dans `vpAreaName`, passez le nom de la zone 4D View Pro.

Dans `name`, passez le nom de la feuille dont l'index sera retourné. Si aucune feuille nommée `name` n'est trouvée dans le document, la méthode retourne -1.

La numérotation démarre à 0.

Exemple

Dans le document suivant :



Lire l'index de la feuille appelée "Total premier trimester" :

```
$index:=VP Get sheet index("ViewProArea";"Total premier trimestre") //retourne 2
```

Voir aussi

[VP Get sheet count](#)

[VP Get sheet name](#)

VP Get sheet name

`VP Get sheet name (vpAreaName : Text ; sheet : Integer) : Text`

Paramètres	Type		Description
<code>vpAreaName</code>	Text	->	Nom d'objet formulaire zone 4D View Pro
<code>sheet</code>	Integer	->	Indice de la feuille
Résultat	Text	<-	Nom de la feuille

Description

La commande `VP Get sheet name` retourne le nom d'une feuille en fonction de son index dans `vpAreaName`.

Dans `vpAreaName`, passez le nom de la zone 4D View Pro.

Dans `sheet`, passez l'index de la feuille dont le nom sera retourné.

Si l'index passé n'existe pas, la méthode retourne un nom vide.

La numérotation démarre à 0.

Exemple

Lire le nom de la troisième feuille du document :

```
$sheetName:=VP Get sheet name("ViewProArea";2)
```

Voir aussi

[VP Get sheet index](#)

VP Get sheet options

VP Get sheet options (*vpAreaName* : Text {; *sheet* : Integer }) : Object

Paramètres	Type		Description
<i>vpAreaName</i>	Text	->	Nom de la zone 4D View Pro dans le formulaire
<i>sheet</i>	Integer	->	Indice de la feuille (feuille courante si omis)
Résultat	Object	<-	Objet options de la feuille

Description

La commande `VP Get sheet options` retourne un objet contenant les options relatives à la feuille courante de la zone *vpAreaName*.

Passez le nom de la zone 4D View Pro dans *vpAreaName*. Si vous passez un nom inexistant, une erreur est retournée.

Dans le paramètre optionnel *sheet*, vous pouvez désigner une feuille spécifique dans laquelle sera définie la plage (la numérotation commence à zéro). Si le paramètre est omis ou si vous passez `vk current sheet`, la feuille courante est utilisée.

Objet retourné

La commande retourne un objet contenant les valeurs courantes pour toutes les options de feuille disponibles. La valeur d'une option peut avoir été modifiée par l'utilisateur ou par la méthode [VP SET SHEET OPTIONS](#).

Pour visualiser la liste complète des options, voir [Options feuille](#).

Exemple

```
$options:=VP Get sheet options("ViewProArea")
If($options.colHeaderVisible) //les en-têtes des colonnes sont visibles
    ...
End if
```

Voir aussi

[4D VIEW PRO SHEET OPTIONS](#)

[VP SET SHEET OPTIONS](#)

VP Get show print lines

VP Get show print lines (*vpAreaName* : Text {; *sheet* : Integer }) : Boolean

Paramètres	Type		Description
<i>vpAreaName</i>	Text	->	Nom d'objet formulaire zone 4D View Pro
<i>sheet</i>	Integer	<-	Indice de la feuille
Résultat	Booléen	<-	True si les lignes d'impression sont visibles, sinon False

Description

La commande `VP Get show print lines` retourne `True` si les lignes d'aperçu avant impression sont visibles et `False` si elles sont masquées.

Dans *vpAreaName*, passez le nom de la zone 4D View Pro.

Dans *sheet*, passez l'index de la page cible. Si *sheet* est omis, la commande s'applique à la feuille courante.

La numérotation démarre à 0.

Exemple

Le code suivant permet de vérifier si les lignes d'aperçu sont visibles ou masquées dans le document :

```
var $result : Boolean  
$result:=VP Get show print lines("ViewProArea";1)
```

Voir aussi

[VP SET SHOW PRINT LINES](#)

VP Get spans

VP Get spans (*rangeObj* : Object) : Object

Paramètres	Type		Description
rangeObj	Object	->	Objet plage
Résultat	Object	<-	Objet de cellules fusionnées (cell spans) dans la plage définie

Description

La commande `VP Get spans` récupère les cellules fusionnées (cell spans) du paramètre *rangeObj* désigné.

Dans *rangeObj*, passez une plage de cellules fusionnées que vous souhaitez récupérer. Si *rangeObj* ne contient pas de cellules fusionnées, une plage vide est renournée.

Exemple

Pour centrer le texte des cellules fusionnées dans ce document :

		First quarter			Second quarter		
		Jan	Feb	Mar	Apr	May	Jun
South area	John						
	Sahra						
	Dixie						

```
// Search for all cell spans  
$range:=VP Get spans(VP All("ViewProArea"))  
  
//center text  
$style:=New object("vAlign";vk vertical align center;"hAlign";vk horizontal align center)  
VP SET CELL STYLE($range;$style)
```

Voir aussi

[VP ADD SPAN](#)

[VP REMOVE SPAN](#)

VP Get stylesheet

VP Get stylesheet (*vpAreaName* : Text ; *styleName* : Text { ; *scope* : Integer }) : Object

Paramètres	Type		Description
vpAreaName	Text	->	Nom d'objet formulaire zone 4D View Pro
styleName	Text	->	Nom du style
scope	Integer	->	Cible (par défaut = feuille courante)
Résultat	Object	<-	Objet feuille de style

Description

La commande `VP Get stylesheet` retourne l'objet feuille courante `styleName` qui contient les valeurs de propriété qui ont été définies.

Dans `vpAreaName`, passez le nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est retournée.

Dans `styleName`, passez le nom de la feuille de style que vous souhaitez lire.

Vous pouvez définir l'emplacement dans lequel vous souhaitez lire la feuille de style dans le paramètre `scope` à l'aide de l'indice de la feuille (la numérotation commence à partir de 0) ou à l'aide des constantes suivantes :

- `vk current sheet`
- `vk workbook`

Exemple

Le code suivant :

```
$style:=VP Get stylesheet("ViewProArea";"GreenDashDotStyle")
```

... retournera l'objet style `GreenDashDotStyle` de la feuille courante :

```
{
  backColor:green,
  borderBottom:{color:green,style:10},
  borderLeft:{color:green,style:10},
  borderRight:{color:green,style:10},
  borderTop:{color:green,style:10}
}
```

Voir aussi

[4D View Pro Style Objects and Style Sheets](#)

[VP ADD STYLESHEET](#)

[VP Get stylesheets](#)

[VP REMOVE STYLESHEET](#)

VP Get stylesheets

`VP Get stylesheets (vpAreaName : Text { ; scope : Integer }) : Collection`

Paramètres	Type		Description
vpAreaName	Text	->	Nom d'objet formulaire zone 4D View Pro
scope	Integer	->	Cible (par défaut = feuille courante)
Résultat	Collection	<-	Collection d'objets feuille de style

Description

La commande `VP Get stylesheets` retourne la collection d'objets feuille de style définis de la cible (`scope`) désignée.

Dans `vpAreaName`, passez la propriété du nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est retournée.

Vous pouvez définir l'emplacement dans lequel vous souhaitez lire les feuilles de style dans le paramètre `scope` à l'aide de l'indice de la feuille (la numérotation commence à partir de 0) ou à l'aide des constantes suivantes :

- `vk current sheet`
- `vk workbook`

Exemple

Le code suivant retournera une collection de tous les objets style de la feuille courante :

```
$styles:=VP Get stylesheets("ViewProArea")
```

Dans ce cas, la feuille courante utilise deux objets style :

```
[  
 {  
   backColor:green,  
   borderLeft:{color:green,style:10},  
   borderTop:{color:green,style:10},  
   borderRight:{color:green,style:10},  
   borderBottom:{color:green,style:10},  
   name:GreenDashDotStyle  
>,  
 {  
   backColor:red,  
   textIndent:10,  
   name:RedIndent  
>  
 ]
```

Voir aussi

[VP ADD STYLESHEET](#)

[VP Get stylesheet](#)

[VP REMOVE STYLESHEET](#)

VP Get value

`VP Get value (rangeObj : Object) : Object`

Paramètres	Type		Description
rangeObj	Object	->	Objet plage
Résultat	Object	<-	Objet contenant une valeur de cellule

Description

La commande `VP Get value` récupère la valeur d'une cellule depuis une plage de cellules désignée.

Dans `rangeObj`, passez la plage dont vous souhaitez récupérer la valeur.

Objet retourné

L'objet retourné contiendra la propriété `value`, et la propriété `time` dans le cas d'une valeur date :

Propriété	Type	Description
value	Entier long, Réel, Booléen, Texte, Date	Valeur de <i>rangeObj</i> (exceptée - time)
time	Réel	Valeur heure (en secondes) si la valeur est du type js

Si l'objet retourné inclut une date ou une heure, il est traité en tant que datetime et est complété comme suit :

- valeur heure - la date est complétée comme suit : December 30, 1899 au format dd/MM/yyyy (30/12/1899)
- valeur date - l'heure est complétée comme suit : minuit au format HH:mm:ss (00:00:00)

Si *rangeObj* contient plusieurs cellules ou plusieurs plages, la valeur de la première cellule est retournée. La commande retourne un objet null si la cellule est vide.

Exemple

```
$cell:=VP Cell("ViewProArea";5;2)
$value:=VP Get value($cell)
If(Value type($value.value)=Is text)
    VP SET TEXT VALUE($cell;New object("value";Uppercase($value.value)))
End if
```

Voir aussi

[VP Get values](#)
[VP SET VALUE](#)
[VP SET VALUES](#)

VP Get values

VP Get values (*rangeObj* : Object) : Collection

Paramètres	Type		Description
rangeObj	Object	->	Objet plage
Résultat	Collection	<-	Collection de valeurs

Description

La commande `VP Get values` récupère les valeurs du paramètre *rangeObj* désigné.

Dans *rangeObj*, passez une plage dont vous souhaitez récupérer les valeurs. Si *rangeObj* comprend plusieurs plages, seule la première plage est utilisée.

La collection retournée par `VP Get values` contient une collection bidimensionnelle :

- Chaque élément de la collection de premier niveau représente une ligne et contient une sous-collection de valeurs
- Chaque sous-collection contient des valeurs des cellule de la ligne. Les valeurs peuvent être de type entier, réel, booléen, texte, null. Si une valeur est de type date ou heure, elle est retournée en un objet dont les propriétés sont les suivantes :

Propriété	Type	Description
value	Date	Valeur de la cellule (excepté - time)
time	Réel	Valeur heure (en secondes) si la valeur est du type js

Les dates ou les heures sont considérées comme un datetime et sont complétées comme suit :

- valeur de type heure - la partie date est complétée comme étant le 30 décembre 1899
- valeur de type date - la partie heure est complétée comme étant minuit (00:00:00:000)

Exemple

Vous souhaitez lire les valeurs allant de C4 à G6 :

	A	B	C	D	E	F	G
1							
2			1	2	3	FALSE	
3							
4			4	5	hello	world	
5			6	7	8	9	
6			29/05/2019 0:00:42				
7							

```
$result:=VP Get values(VP Cells("ViewProArea";2;3;5;3))
// $result[0]=[4,5,null,hello,world]
// $result[1]=[6,7,8,9,null]
// $result[2]=[null,{time:42,value:2019-05-29T00:00:00.000Z},null,null,null]
```

Voir aussi

[VP Get formulas](#)

[VP Get value](#)

[VP SET FORMULAS](#)

[VP SET VALUES](#)

VP Get workbook options

VP Get workbook options (*vpAreaName* : Text) : Object

Paramètres	Type		Description
<i>vpAreaName</i>	Text	->	Nom d'objet formulaire zone 4D View Pro
Résultat	Object	<-	Objet contenant les options de workbook

Description

`VP Get workbook options` returns an object containing all the workbook options in *vpAreaName*

Dans *vpAreaName*, passez le nom de la zone 4D View Pro.

The returned object contains all the workbook options (default and modified ones), in the workbook.

The list of workbook options is referenced in [VP SET WORKBOOK OPTIONS](#)'s description.

Exemple

```
var $workbookOptions : Object
$workbookOptions:=VP Get workbook options("ViewProArea")
```

Voir aussi

[VP SET WORKBOOK OPTIONS](#)

VP IMPORT DOCUMENT

VP IMPORT DOCUMENT (*vpAreaName* : Text ; *filePath* : Text { ; *paramObj* : Object})

Paramètres	Type		Description
<i>vpAreaName</i>	Text	->	Nom d'objet formulaire zone 4D View Pro
<i>filePath</i>	Text	->	Chemin d'accès du document
<i>paramObj</i>	Object	->	Options d'import

Description

La commande `VP IMPORT DOCUMENT` importe et affiche le document désigné par le dont le chemin d'accès est indiqué dans *filePath* dans la zone *vpAreaName* de 4D View Pro. Le document importé remplace toutes les données déjà présentes dans la zone.

Dans *vpAreaName*, passez le nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est retournée.

Dans *filePath*, passez le chemin et le nom du document à importer. Les formats suivants sont pris en charge :

- Les documents 4D View Pro (extension ".4vp")
- Microsoft Excel (extension ".xlsx")
- documents texte (extension ".txt", ".csv", le document doit être en utf-8)

Si l'extension du document n'est pas une extension reconnue, telle que `.4vp` ou `.xlsx`, le document est considéré comme un document texte. Vous devez passer un chemin d'accès complet, sauf si le document est situé au même niveau que le dossier Project, auquel cas vous pouvez simplement passer son nom.

When importing a Microsoft Excel-formatted file into a 4D View Pro document, some settings may be lost. You can verify your settings with [this list from GrapeCity](#).

Une erreur est retournée si le paramètre `filePath` est invalide, ou si le fichier est manquant ou mal-formé.

Le paramètre optionnel *paramObj* vous permet de définir les propriétés du document importé :

Paramètres		Type	Description
<i>formula</i>		object	Nom d'une méthode callback (ou méthode rétro-rappel) à lancer lorsque l'import est terminé. La méthode doit utiliser la commande <code>Formula</code> . Voir Passing a callback method (formula) .
<i>password</i>		Texte	Microsoft Excel uniquement (optionnel) - Mot de passe utilisé pour protéger un document Microsoft Excel.
<i>csvOptions</i>		object	options d'import csv
	<i>range</i>	object	Plage de cellules contenant la première cellule dans laquelle les données seront saisies. Si la plage spécifiée n'est pas une plage de cellules, seule la première cellule de la plage est utilisée.
	<i>rowDelimiter</i>	Texte	Délimiteur de ligne. S'il n'est pas défini, le délimiteur est automatiquement déterminé par 4D.
	<i>columnDelimiter</i>	Texte	Délimiteur de colonne. Par défaut : ","

For more information on the CSV format and delimiter-separated values in general, see [this article on Wikipedia](#)

Exemple 1

Vous souhaitez importer un document 4D View Pro stocké sur le disque, à l'ouverture du formulaire :

```

C_TEXT($docPath)
If(Form event code=On VP Ready) //La zone 4D View Pro est chargée et prête
    $docPath:="C:\\Bases\\ViewProDocs\\MyExport.4VP"
    VP IMPORT DOCUMENT("VPArea";$docPath)
End if

```

Exemple 2

Vous souhaitez importer un document Microsoft Excel protégé par un mot de passe dans 4D View Pro :

```

$o:=New object
$o.password:="excel123"

VP IMPORT DOCUMENT("ViewProArea";"c:\\tmp\\excefile.xlsx";$o)

```

Exemple 3

Vous souhaitez importer un fichier `.txt` qui utilise une virgule (",") comme délimiteur :

```

"Clark","Kent"
"Bruce","Wayne"
"Barry","Allen"
"Peter","Parker"
"Tony","Stark"

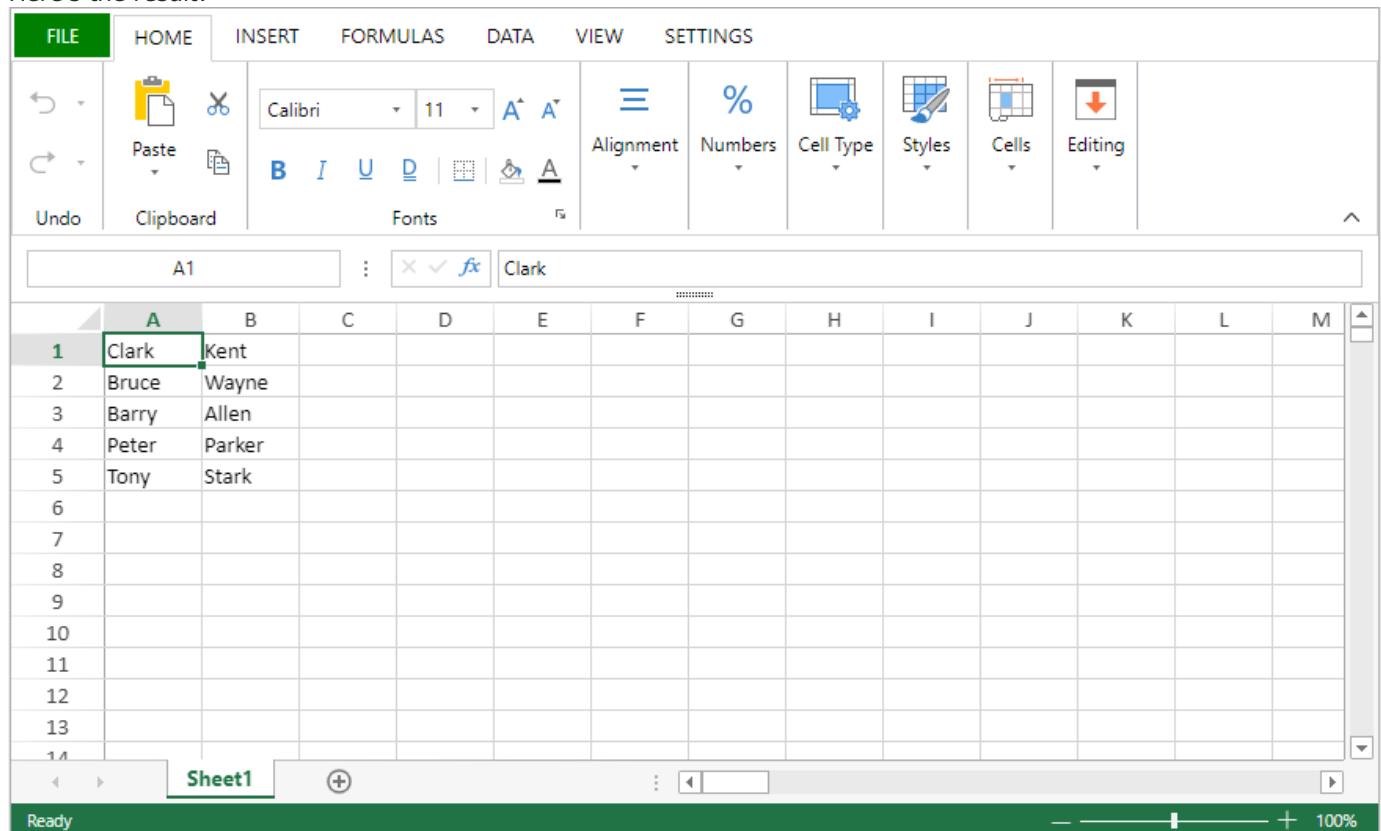
```

```

$params:=New object
$params.range:=VP Cells("ViewProArea";0;0;2;5)
VP IMPORT DOCUMENT("ViewProArea";"c:\\import\\my-file.txt";New object("csvOptions";$params))

```

Here's the result:



The screenshot shows a Microsoft Excel spreadsheet with the following data in the first five rows:

	A	B
1	Clark	Kent
2	Bruce	Wayne
3	Barry	Allen
4	Peter	Parker
5	Tony	Stark
6		
7		
8		
9		
10		
11		
12		
13		
14		

Voir aussi

[VP EXPORT DOCUMENT](#)
[VP NEW DOCUMENT](#)

VP IMPORT FROM OBJECT

VP IMPORT FROM OBJECT (*vpAreaName* : Text { ; *viewPro* : Object})

Paramètres	Type		Description
<i>vpAreaName</i>	Text	->	Nom d'objet formulaire zone 4D View Pro
<i>viewPro</i>	Object	->	Objet 4D View Pro

Description

La commande `VP IMPORT FROM OBJECT` importe l'objet 4D View Pro *viewPro* et l'affiche dans la zone 4D View Pro *vpAreaName*. Le contenu de l'objet importé remplace toutes les données insérées auparavant dans la zone.

Dans *vpAreaName*, passez le nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est retournée.

Dans *viewPro*, passez un objet 4D View Pro valide. Cet objet peut avoir été créé en utilisant [VP Export to object](#) ou manuellement. Pour plus d'informations sur les objets 4D View Pro, référez-vous à la section [Objet 4D View Pro](#).

Une erreur est retournée si l'objet *viewPro* est invalide.

Exemple

Vous souhaitez importer une feuille de calcul stockée dans un champ objet :

```
QUERY( [VPWorkBooks] ; [VPWorkBooks] ID=10 )
VP IMPORT FROM OBJECT("ViewProArea1"; [VPWorkBooks] SPBook)
```

Voir aussi

[VP Export to object](#)

VP INSERT COLUMNS

VP INSERT COLUMNS (*rangeObj* : Object)

Paramètres	Type		Description
<i>rangeObj</i>	Object	->	Objet plage

Description

La commande `VP INSERT COLUMNS` insère les colonnes définies par *rangeObj*.

Dans *rangeObj*, passez un objet plage indiquant la colonne de début (la colonne qui définit l'emplacement de la nouvelle colonne à insérer) et le nombre de colonnes à insérer. Si le nombre de colonnes à insérer est omis (non défini), une seule colonne est insérée.

De nouvelles colonnes sont insérées sur la gauche, directement avant la première colonne de *rangeObj*.

Exemple

Pour insérer 3 colonnes avant la deuxième colonne :

```
VP INSERT COLUMNS(VP Column("ViewProArea";1;3))
```

Le résultat est le suivant :

Before insertion

	A	B	C	D	E	F	G	H	I	J
1	The	quick	brown	fox	jumped	over	the	lazy	dog	
2										
3										

After insertion

	A	B	C	D	E	F	G	H	I	J	K	L
1	The				quick	brown	fox	jumped	over	the	lazy	dog
2												
3												

Voir aussi

[VP DELETE COLUMNS](#)

[VP DELETE ROWS](#)

[VP INSERT ROWS](#)

VP INSERT ROWS

VP INSERT ROWS (*rangeObj* : Object)

Paramètres	Type		Description
<i>rangeObj</i>	Object	->	Objet plage

Description

La commande `VP INSERT ROWS` insère les lignes définies par *rangeObj*.

Dans *rangeObj*, passez un objet plage indiquant la ligne de début (la ligne qui définit l'emplacement de la nouvelle ligne à insérer) et le nombre de lignes à insérer. Si le nombre de lignes à insérer est omis (non défini), une seule ligne est insérée.

De nouvelles lignes sont insérées directement avant la première ligne de *rangeObj*.

Exemple

Pour insérer 3 lignes avant la première ligne :

```
VP INSERT ROWS(VP Row("ViewProArea";0;3))
```

Le résultat est le suivant :

Before insertion

	A	B	C	D	E	F	G	H	I	J
1	The	quick	brown	fox	jumped	over	the	lazy	dog	
2										
3										
4										
e										

After insertion

	A	B	C	D	E	F	G	H	I	J
1										
2										
3										
4	The	quick	brown	fox	jumped	over	the	lazy	dog	
e										

Voir aussi

[VP DELETE COLUMNS](#)

[VP DELETE ROWS](#)

[VP INSERT COLUMNS](#)

M

VP MOVE CELLS

► Historique

VP MOVE CELLS (*originRange* : Object ; *targetRange* : Object ; *options* : Object)

Paramètres	Type		Description
<i>originRange</i>	Object	->	Cell range to copy from
<i>targetRange</i>	Object	->	Target range for the values, formatting and formulas
<i>options</i>	Object	->	Options supplémentaires

Description

The **VP MOVE CELLS** command moves or copies the values, style and formulas from *originRange* to *targetRange*.

originRange and *targetRange* can refer to different View Pro areas.

In *originRange*, pass a range object containing the values, style, and formula cells to copy or move. If *originRange* is a combined range, only the first one is used.

In *targetRange*, pass the range of cells where the cell values, style, and formulas will be copied or moved.

The *options* parameter has several properties:

Propriété	Type	Description														
copy	Booléen	Determines if the values, formatting and formulas of the cells in <i>originRange</i> are removed after the command executes: <ul style="list-style-type: none"> • <i>False</i> (default) to remove them • <i>True</i> to keep them 														
pasteOptions	Longint	Specifies what is pasted. Valeurs possibles : <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Valeur</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>vk clipboard options all</code> (par défaut)</td> <td>Pastes all data objects, including values, formatting, and formulas.</td> </tr> <tr> <td><code>vk clipboard options formatting</code></td> <td>Pastes only the formatting.</td> </tr> <tr> <td><code>vk clipboard options formulas</code></td> <td>Pastes only the formulas.</td> </tr> <tr> <td><code>vk clipboard options formulas and formatting</code></td> <td>Pastes the formulas and formatting.</td> </tr> <tr> <td><code>vk clipboard options values</code></td> <td>Pastes only the values.</td> </tr> <tr> <td><code>vk clipboard options value and formatting</code></td> <td>Pastes the values and formatting.</td> </tr> </tbody> </table>	Valeur	Description	<code>vk clipboard options all</code> (par défaut)	Pastes all data objects, including values, formatting, and formulas.	<code>vk clipboard options formatting</code>	Pastes only the formatting.	<code>vk clipboard options formulas</code>	Pastes only the formulas.	<code>vk clipboard options formulas and formatting</code>	Pastes the formulas and formatting.	<code>vk clipboard options values</code>	Pastes only the values.	<code>vk clipboard options value and formatting</code>	Pastes the values and formatting.
Valeur	Description															
<code>vk clipboard options all</code> (par défaut)	Pastes all data objects, including values, formatting, and formulas.															
<code>vk clipboard options formatting</code>	Pastes only the formatting.															
<code>vk clipboard options formulas</code>	Pastes only the formulas.															
<code>vk clipboard options formulas and formatting</code>	Pastes the formulas and formatting.															
<code>vk clipboard options values</code>	Pastes only the values.															
<code>vk clipboard options value and formatting</code>	Pastes the values and formatting.															

Les options de collage définies dans les [options de workbook](#) sont prises en compte.

Exemple

To copy the contents, values, formatting and formulas from an origin range:

```
var $originRange; $targetRange; $options : Object
$originRange:=VP Cells("ViewProArea"; 0; 0; 2; 5)
$targetRange:=VP Cells("ViewProArea"; 4; 0; 2; 5)
$options:=New object
$options.copy:=True
$options.pasteOptions:=vk clipboard options all
VP MOVE CELLS($originRange; $targetRange; $options)
```

Voir aussi

[VP Copy to object](#)
[VP PASTE FROM OBJECT](#)
[VP SET WORKBOOK OPTIONS](#)

N

VP Name

VP Name (*vpAreaName* : Text ; *rangeName* : Text { ; *scope* : Integer }) : Object

Paramètres	Type		Description
vpAreaName	Text	->	Nom d'objet formulaire zone 4D View Pro
rangeName	Text	->	Nom de plage existante
scope	Integer	->	Emplacement de la plage (si omis, feuille courante)
Résultat	Object	<-	Plage nommée

Description

La commande `VP Name` retourne une nouvelle plage référençant une plage nommée.

Dans `vpAreaName`, passez le nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est retournée.

Le paramètre `rangeName` indique une plage de cellule nommée existante.

Dans le paramètre optionnel `scope`, vous pouvez désigner une feuille spécifique dans laquelle est défini `rangeName`. Si le paramètre est omis, la feuille courante est utilisée par défaut. Vous pouvez sélectionner explicitement la feuille courante ou l'intégralité du classeur (workbook) à l'aide des constantes suivantes :

- `vk current sheet`
- `vk workbook`

Exemple

Vous souhaitez assigner une valeur à une plage nommée "Total".

```
// nommez la cellule B5 Total
VP ADD RANGE NAME(VP Cell("ViewProArea";1;4);"Total")
$name:=VP Name("ViewProArea";" Total")
VP SET NUM VALUE($name;285;"$#,###.00")
```

Voir aussi

[VP ADD RANGE NAME](#)
[VP ALL](#)
[VP Cell](#)
[VP Cells](#)
[VP Column](#)
[VP Combine ranges](#)
[VP Get names](#)
[VP REMOVE NAME](#)
[VP Row](#)

VP NEW DOCUMENT

`VP NEW DOCUMENT (vpAreaName : Text)`

Paramètres	Type		Description
vpAreaName	Text	->	Nom d'objet formulaire zone 4D View Pro

Description

La commande `VP NEW DOCUMENT` charge et affiche un nouveau document dans l'objet `vpAreaName` de la zone 4D View Pro sur le formulaire. Le nouveau document vide remplace toutes les données auparavant insérées dans la zone.

Dans `vpAreaName`, passez le nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est retournée.

Exemple

Vous souhaitez afficher un document vide dans l'objet formulaire "myVPArea" :

```
VP NEW DOCUMENT("myVPArea")
```

Voir aussi

[VP IMPORT DOCUMENT](#)

O

VP Object to font

VP Object to font (*fontObj* : Object) : Text

Paramètres	Type		Description
font object	Object	->	Objet police
Résultat	Text	<-	Police raccourcie (shorthand)

Description

La commande `VP Object to font` retourne une chaîne raccourcie pour la police (shorthand) de *fontObj*.

Dans *fontObj*, passez un objet contenant les propriétés de police. Les propriétés suivantes sont prises en charge :

Propriété	Type	Description	Possible values	Obligatoire
family	Texte	Définit la police.	tout type de famille de police standard ou générique. Ex : "Arial", "Helvetica", "serif", "arial,sans-serif"	Oui
size	Texte	Définit la taille de la police. Le line-height peut être ajouté au font-size : font-size/line-height: Ex : "15pt/20pt"	un chiffre avec l'une des unités suivantes : <ul style="list-style-type: none"> • "em", "ex", "%", "px", "cm", "mm", "in", "pt", "pc", "ch", "rem", "vh", "vw", "vmin", "vmax" ou l'une des constantes suivantes : <ul style="list-style-type: none"> • <code>vk font size large</code> • <code>vk font size larger</code> • <code>vk font size x large</code> • <code>vk font size xx large</code> • <code>vk font size small</code> • <code>vk font size smaller</code> • <code>vk font size x small</code> • <code>vk font size xx small</code> 	Oui
style	Texte	Style de police.	<ul style="list-style-type: none"> • <code>vk font style italic</code> • <code>vk font style oblique</code> 	Non
variant	Texte	Police en petites majuscules.	<ul style="list-style-type: none"> • <code>vk font variant small caps</code> 	Non
weight	Texte	Définit l'épaisseur de la police.	<ul style="list-style-type: none"> • <code>vk font weight 100</code> • <code>vk font weight 200</code> • <code>vk font weight 300</code> • <code>vk font weight 400</code> • <code>vk font weight 500</code> • <code>vk font weight 600</code> • <code>vk font weight 700</code> • <code>vk font weight 800</code> • <code>vk font weight 900</code> • <code>vk font weight bold</code> • <code>vk font weight bolder</code> • <code>vk font weight lighter</code> 	Non

Cet objet peut être créé à l'aide de la commande [VP Font to object](#).

La chaîne raccourcie renvoyée peut être affectée à la propriété "font" d'une cellule à l'aide, par exemple, de [VP SET CELL STYLE](#).

Exemple

```
$cellStyle:=VP Get cell style($range)

$font:=VP Font to object($cellStyle.font)
$font.style:=vk font style oblique
$font.variant:=vk font variant small caps
$font.weight:=vk font weight bolder

$cellStyle.font:=VP Object to font($font)
///$cellStyle.font contient "bolder oblique small-caps 16pt arial"
```

Voir aussi

[4D View Pro Style Objects and Style Sheets](#)

[VP Font to object](#)

[VP SET CELL STYLE](#)

[VP SET DEFAULT STYLE](#)

P

VP PASTE FROM OBJECT

► Historique

VP PASTE FROM OBJECT (*rangeObj* : Object ; *dataObject* : Object {; *options* : Longint})

Paramètres	Type		Description
<i>rangeObj</i>	Object	->	Cell range object
<i>dataObject</i>	Object	->	Object containing the data to be pasted
<i>options</i>	Longint	->	Specifies what is pasted

Description

The **VP PASTE FROM OBJECT** command pastes the contents, style and formulas stored in *dataObject* to the *rangeObj* object.

In *rangeObj*, pass the cell range object where the values, formatting, and/or formula cells will be pasted. If *rangeObj* refers to more than one cell, only the first one is used.

In *dataObject*, pass the object that contains the cell data, formatting, and formulas to be pasted.

In the optional *options* parameter, you can specify what to paste in the cell range. Valeurs possibles :

Constante	Description
<code>vk clipboard options all</code>	Pastes all data objects, including values, formatting, and formulas.
<code>vk clipboard options formatting</code>	Pastes only the formatting.
<code>vk clipboard options formulas</code>	Pastes only the formulas.
<code>vk clipboard options formulas and formatting</code>	Pastes formulas and formatting.
<code>vk clipboard options values</code>	Pastes only values.
<code>vk clipboard options value and formatting</code>	Pastes values and formatting.

Les options de collage définies dans les [options de workbook](#) sont prises en compte.

If *options* refers to a paste option not present in the copied object (e.g. formulas), the command does nothing.

Exemple

See example the example from [VP Copy to object](#)

Voir aussi

[VP Copy to object](#)

[VP MOVE CELLS](#)

[VP Get workbook options](#)

[VP SET WORKBOOK OPTIONS](#)

VP PRINT

VP PRINT (*vpAreaName* : Text { ; *sheet* : Integer })

Paramètres	Type		Description
<i>vpAreaName</i>	Text	->	Nom d'objet formulaire zone 4D View Pro
<i>sheet</i>	Integer	->	Indice de la feuille (feuille courante si omis)

Description

La commande `VP PRINT` ouvre une fenêtre de dialogue d'impression pour imprimer *vpAreaName*.

Passez la zone 4D View Pro à imprimer dans le paramètre *vpAreaName*. La commande ouvrira la fenêtre de dialogue d'impression permettant de définir l'imprimante et les propriétés de la page.

The properties defined in the print dialog window are for the printer paper, they are not the printing properties for the 4D View Pro area. Printing properties for 4D View Pro areas are defined using the [VP SET PRINT INFO](#) command. It is highly recommended that the properties for both the printer and the 4D View Pro area match, otherwise the printed document may not correspond to your expectations.

Dans le paramètre optionnel *sheet*, vous pouvez définir une feuille (sheet) spécifique à imprimer (la numérotation démarre à zéro). S'il est omis, la feuille courante est utilisée par défaut. Vous pouvez sélectionner explicitement la feuille courante ou le workbook entier à l'aide des constantes suivantes :

- `vk current sheet`
- `vk workbook`

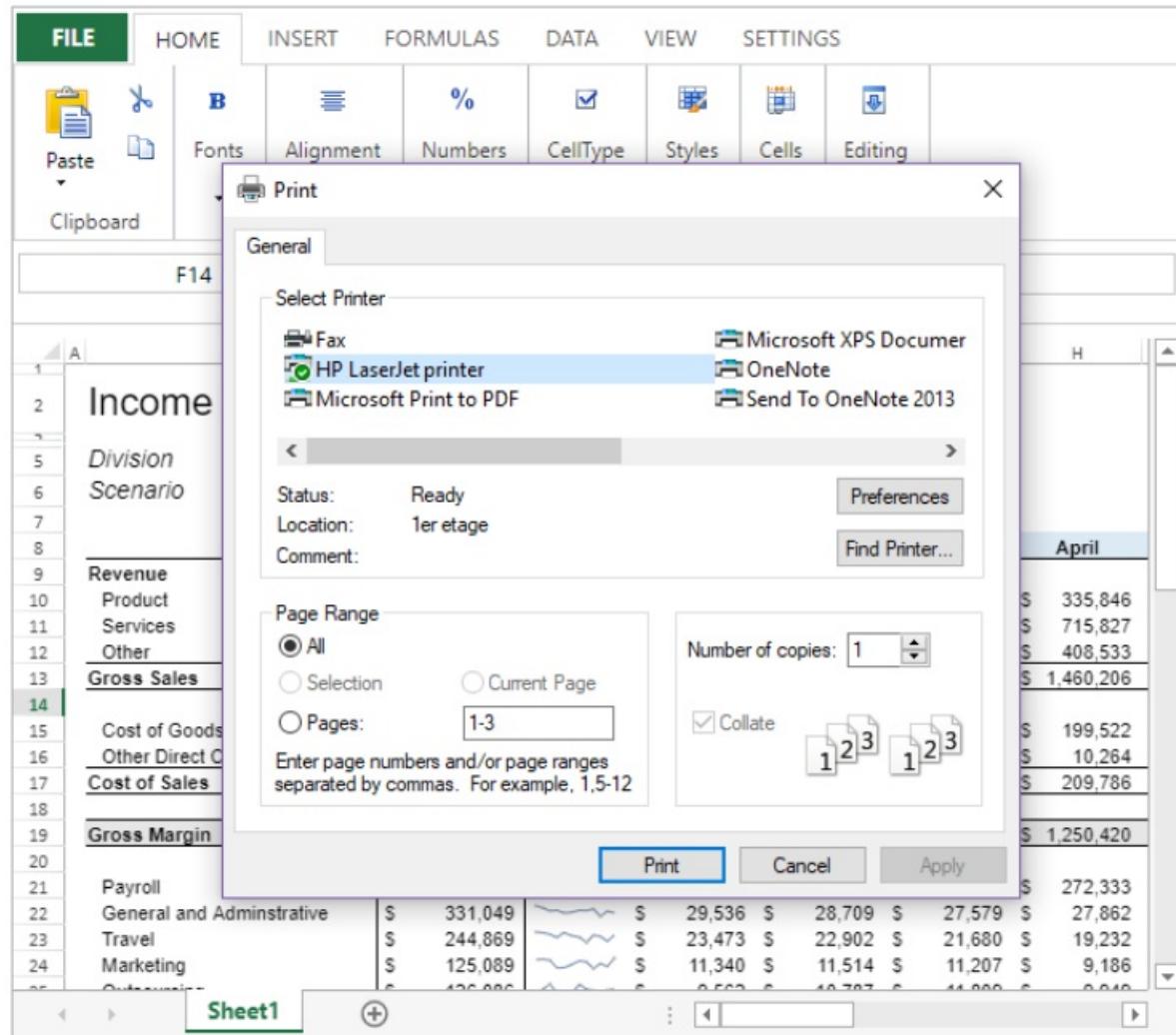
- 4D View Pro areas can only be printed with the `VP PRINT` command.
- Commands from the 4D Printing language theme are not supported by `VP PRINT`.
- This command is intended for individual printing by the final end user. For automated print jobs, it is advised to export the 4D View Pro area as a PDF with the [VP EXPORT DOCUMENT](#) method.

Exemple

Le code suivant :

```
VP PRINT("myVPArea")
```

Ouvrira une fenêtre de dialogue d'impression :



Voir aussi

[VP EXPORT DOCUMENT](#)
[VP SET PRINT INFO](#)

R

VP RECOMPUTE FORMULAS

VP RECOMPUTE FORMULAS (vpAreaName : Text)

Paramètres	Type	Description
vpAreaName	Text	-> Nom d'objet formulaire zone 4D View Pro

Description

La commande **VP RECOMPUTE FORMULAS** évalue immédiatement toutes les formules de *vpAreaName*. Par défaut, 4D calcule automatiquement les formules lorsqu'elles sont insérées, importées ou exportées. **VP RECOMPUTE FORMULAS** vous permet de forcer le calcul à tout moment (ex : si les formules sont modifiées ou si les formules contiennent des appels vers la base). La commande lance l'exécution de la commande **VP FLUSH COMMANDS** pour exécuter les commandes stockées et vider le tampon de commandes, puis calcule toutes les formules dans le workbook.

Dans *vpAreaName*, passez le nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est retournée.

Assurez-vous que la commande n'a pas été exécutée auparavant à l'aide de **VP RECOMPUTE FORMULAS**, sinon la commande ne fait rien.

Exemple

Pour actualiser toutes les forumules du workbook, saisissez le code suivant :

```
VP RECOMPUTE FORMULAS("ViewProArea")
```

Voir aussi

[VP RESUME COMPUTING](#)

[VP SUSPEND COMPUTING](#)

VP REMOVE NAME

VP REMOVE NAME (*vpAreaName* : Text ; *name* : Text { ; *scope* : Integer })

Paramètres	Type		Description
<i>vpAreaName</i>	Text	->	Nom d'objet formulaire zone 4D View Pro
<i>name</i>	Text	->	Nom de la plage nommée ou de la formule nommée à supprimer
<i>scope</i>	Integer	->	"Scope" cible (par défaut=feuille courante)

Description

La commande `VP REMOVE NAME` permet de supprimer la plage nommée ou la formule nommée que vous avez passée dans le paramètre *name* dans le *scope* défini.

Dans *vpAreaName*, passez le nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est retournée.

Passez la plage nommée ou la formule nommée que vous souhaitez supprimer dans *name*.

Le paramètre *scope* vous permet de définir l'emplacement dans lequel vous souhaitez supprimer le nom, en passant l'indice de la feuille (la numérotation débute à 0) ou l'une des constantes suivantes :

- `vk current sheet`
- `vk workbook`

Exemple

```
$range:=VP Cell("ViewProArea";0;0)
VP ADD RANGE NAME("Total1";$range)

VP REMOVE NAME("ViewProArea";"Total1")
$formula:=VP Get formula by name("ViewProArea";"Total1")
//$formula=null
```

Voir aussi

[VP Name](#)

VP REMOVE SHEET

VP REMOVE SHEET (*vpAreaName* : Text ; *index*: Integer)

Paramètres	Type		Description
<i>vpAreaName</i>	Text	->	Nom d'objet formulaire zone 4D View Pro
<i>index</i>	Integer	->	Index of the sheet to remove

Voir aussi

[VP ADD SHEET](#)

Description

The `VP REMOVE SHEET` command removes the sheet with the specified *index* from the document loaded in *vpAreaName*.

Dans *vpAreaName*, passez le nom de la zone 4D View Pro.

In *index*, pass the index of the sheet to remove. If the passed *index* does not exist, the command does nothing.

La numérotation démarre à 0.

Exemple

The document currently has three sheets:



Remove the third sheet:

```
VP REMOVE SHEET("ViewProArea";2)
```



VP REMOVE SPAN

`VP REMOVE SPAN (rangeObj : Object)`

Paramètres	Type		Description
rangeObj	Object	->	Objet plage

Description

The command `VP REMOVE SPAN` removes the fusion of the cells of *rangeObj*.

In *rangeObj*, pass a fusion range object. The fused cells of the range are divided into individual cells.

Exemple

To remove all fusions of cells in this document :

		First quarter			Second quarter		
		Jan	Feb	Mar	Apr	May	Jun
South area	John						
	Sahra						
	Dixie						

```
//identifier toutes les cellules fusionnées
$span:=VP Get spans(VP All("ViewProArea"))

//retirer les fusions
VP REMOVE SPAN($span)
```

Résultat :

		First quart		Second qu			
		Jan	Feb	Mar	Apr	May	Jun
South area	John						
	Sahra						
	Dixie						

Voir aussi

[VP ADD SPAN](#)
[VP Get spans](#)

VP REMOVE STYLESHEET

VP REMOVE STYLESHEET (*vpAreaName* : Text ; *styleName* : Text { ; *scope* : Integer })

Paramètres	Type		Description
<i>vpAreaName</i>	Text	->	Nom d'objet formulaire zone 4D View Pro
<i>styleName</i>	Text	->	Nom du style à supprimer
<i>scope</i>	Integer	->	Cible (par défaut = feuille courante)

Description

La commande `VP REMOVE STYLESHEET` supprime la feuille de style passée dans *styleName* de *vpAreaName*.

Dans *vpAreaName*, passez le nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est retournée.

Dans *styleName*, passez le nom de la feuille de style que vous souhaitez supprimer.

Vous pouvez définir, dans le paramètre optionnel *scope*, l'emplacement dans lequel vous souhaitez supprimer le style, à l'aide de l'indice de la feuille (la numérotation commence à partir de 0) ou à l'aide des constantes suivantes :

- `vk current sheet`
- `vk workbook`

Exemple

Pour supprimer l'objet style *GreenDashDotStyle* de la feuille courante :

```
VP REMOVE STYLESHEET("ViewProArea";"GreenDashDotStyle")
```

Voir aussi

[VP ADD STYLESHEET](#)
[VP Get stylesheet](#)
[VP Get stylesheets](#)

VP RESET SELECTION

VP RESET SELECTION (*vpAreaName* : Text { ; *sheet* : Integer })

Paramètres	Type		Description
<i>vpAreaName</i>	Text	->	Nom d'objet formulaire zone 4D View Pro
<i>sheet</i>	Integer	->	Indice de la feuille (feuille courante si omis)

Description

La commande **VP RESET SELECTION** désélectionne toutes les cellules, de telle façon qu'il n'existe plus aucune sélection courante ou cellule active visible.

Une cellule active par défaut (cellule A1) reste définie pour les commandes 4D View Pro.

Dans *vpAreaName*, passez le nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est retournée.

Dans le paramètre optionnel *sheet*, vous pouvez désigner une feuille spécifique dans laquelle sera définie la plage (la numérotation commence à zéro). Si le paramètre est omis, la feuille courante est utilisée par défaut. Vous pouvez sélectionner explicitement la feuille courante à l'aide de la constante suivante :

- `vk current sheet`

Exemple

Vous souhaitez désélectionner toutes les cellules (la cellule active et toute autre cellule sélectionnée) :

```
VP RESET SELECTION("myVPArea")
```

Voir aussi

[VP ADD SELECTION](#)
[VP Get active cell](#)
[VP Get selection](#)
[VP SET ACTIVE CELL](#)
[VP SET SELECTION](#)
[VP SHOW CELL](#)

VP RESUME COMPUTING

VP RESUME COMPUTING (*vpAreaName* : Text)

Paramètres	Type		Description
<i>vpAreaName</i>	Text	->	Nom d'objet formulaire zone 4D View Pro

Description

La commande **VP RESUME COMPUTING** redémarre le calcul des formules dans *vpAreaName*.

La commande réactive le service de calcul de 4D View Pro. Les formules impactées par des modifications apportées durant la suspension des calculs sont mises à jour, et les formules ajoutées après l'exécution de **VP RESUME COMPUTING** sont calculées.

Dans *vpAreaName*, passez le nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est retournée.

Le service de calcul de 4D View Pro maintient un compteur d'actions de suspension/reprise. Ainsi, chaque exécution de commande **VP RESUME COMPUTING** doit être compensée par une exécution correspondante de la commande **VP SUSPEND COMPUTING**.

Exemple

Voir l'exemple dans [VP SUSPEND COMPUTING](#).

Voir aussi

[VP RECOMPUTE FORMULAS](#)

[VP SUSPEND COMPUTING](#)

VP Row

`VP Row (vpAreaName : Text; row : Integer { ; rowCount : Integer { ; sheet : Integer } }) : Object`

Paramètres	Type		Description
vpAreaName	Text	->	Nom d'objet formulaire zone 4D View Pro
row	Integer	->	Indice de la ligne
rowCount	Integer	->	Nombre de lignes
sheet	Integer	->	Indice de la feuille (feuille courante si omis)
Résultat	Object	<-	Plage de ligne(s)

Description

La commande `VP Row` retourne une nouvelle plage référençant une ou plusieurs lignes spécifiques.

Dans `vpAreaName`, passez le nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est retournée.

Le paramètre `row` définit la première ligne de la plage. Passez l'indice de la ligne (la numérotation commence à zéro) dans ce paramètre. Passez l'indice de la ligne (la numérotation commence à zéro) dans ce paramètre.

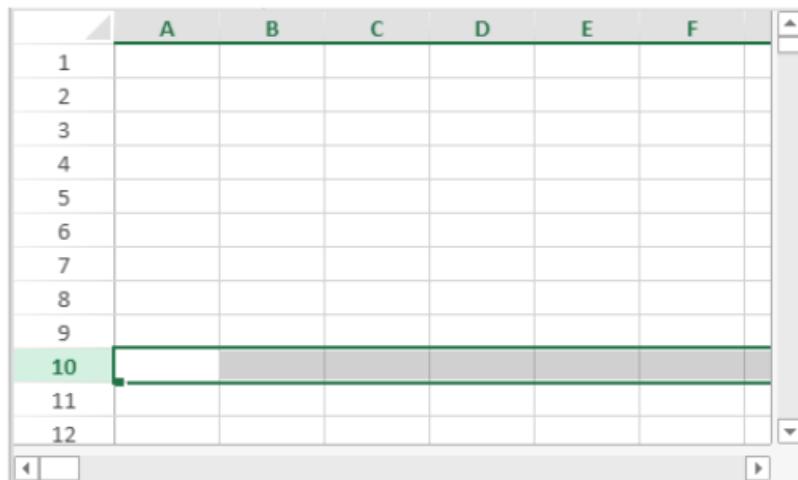
Dans le paramètre `row`, vous pouvez définir l'emplacement de la ou des lignes de la plage de cellules. `rowCount` doit être supérieur à 0. Passez l'indice de la ligne (la numérotation commence à zéro) dans ce paramètre.

Dans le paramètre optionnel `sheet`, vous pouvez désigner une feuille spécifique dans laquelle sera définie la plage (la numérotation commence à zéro). Si le paramètre est omis, la feuille courante est utilisée par défaut. Vous pouvez sélectionner explicitement la feuille courante à l'aide de la constante suivante :

- `vk current sheet`

Exemple

Vous souhaitez définir une plage pour la ligne ci-dessous (dans la feuille courante) :



Vous pouvez écrire :

```
$row:=VP Row("ViewProArea";9) // row 10
```

Voir aussi

VP All
VP Cell
VP Cells
VP Column
VP Combine ranges
VP Name

VP ROW AUTOFIT

VP ROW AUTOFIT (*rangeObj* : Object)

Paramètres	Type		Description
rangeObj	Object	->	Objet plage

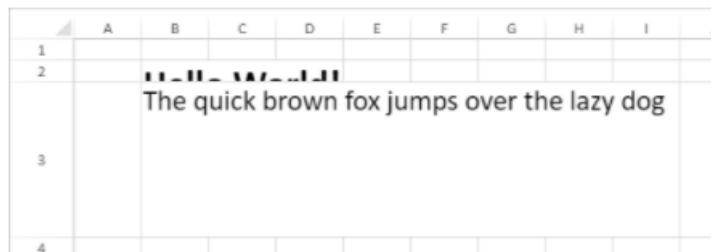
Description

La commande `VP ROW AUTOFIT` dimensionne automatiquement dans *rangeObj* la ou les ligne(s) en fonction de leur contenu.

Dans `rangeObj`, passez un objet plage contenant une plage de lignes dont la taille est gérée automatiquement.

Exemple

Les lignes suivantes n'affichent pas correctement le texte :



```
VP ROW AUTOFIT(VP Row("ViewProArea";1;2))
```

Résultat :



Voir aussi

VP Column autofit

VP Run offscreen area

VP Run offscreen area (parameters : Object) : Mixed

Paramètres	Type		Description
parameters	Object	->	Objet contenant les attributs de la zone hors écran
Résultat	Mixed	<-	Propriété <code>.result</code> de l'objet <code>.onEvent</code> , ou Null s'il ne retourne pas de valeur

Description

La commande `VP Run offscreen area` crée, dans la mémoire, une zone hors écran qui peut être utilisée pour traiter les commandes et fonctions d'une zone 4D View Pro.

Dans l'objet `parameters`, passez l'une des propriétés optionnelles suivantes. Ces propriétés seront disponibles grâce à la commande `This` dans la méthode `onEvent` et référencent l'instance :

Propriété	Type	Description
area	Texte	Le nom de la zone hors écran. S'il est omis ou null, un nom générique est assigné (ex : OffscreenArea1).
onEvent	objet (formula)	<p>Une méthode callback qui sera lancée lorsque la zone hors écran sera prête. Elle peut être soit :</p> <ul style="list-style-type: none"> • une fonction <code>onEvent</code> d'une classe, soit • un objet <code>Formula</code> <p>Par défaut, la méthode callback est appelée sur les événements <code>On VP Ready</code>, <code>On Load</code>, <code>On Unload</code>, <code>On End URL Loading</code>, <code>On URL Loading Error</code>, <code>On VP Range Changed</code>, or <code>On Timer</code>.</p> <p>La méthode callback peut être utilisée pour accéder à l'objet 4D View Pro.</p>
autoQuit	boolean	Vrai (valeur par défaut) si la commande doit stopper l'exécution de la formule lorsque les événements <code>On End URL Loading</code> ou <code>On URL Loading Error</code> se produisent. Si faux, vous devez utiliser les commandes <code>CANCEL</code> ou <code>ACCEPT</code> dans la méthode callback <code>onEvent</code> .
timeout	number	Durée maximale (exprimée en secondes) avant la fermeture de la zone si aucun événement n'est généré. Si elle est fixée à 0, aucune limitation n'est appliquée. Valeur par défaut : 60
result	mixte	Résultat du traitement (le cas échéant)
<customProperty>	mixte	Tout attribut personnalisé qui sera disponible dans la méthode callback <code>onEvent</code> .

La propriété suivante est automatiquement ajoutée par la commande, si nécessaire :

Propriété	Type	Description
timeoutReached	boolean	Ajouté avec la valeur vrai si le timeout a été dépassé

La zone hors écran est uniquement disponible durant l'exécution de la commande `VP Run offscreen area`. Elle sera automatiquement détruite à la fin de l'exécution.

Les commandes suivantes peuvent être utilisées dans la méthode callback (de rétro-appel) :

- `ACCEPT`
- `CANCEL`
- `SET TIMER`
- `WA Evaluate JavaScript`
- `WA EXECUTE JAVASCRIPT FUNCTION`

Exemple 1

Vous souhaitez créer une zone 4D View Pro hors écran et lire la valeur d'une cellule :

```
// cs.OffscreenArea class declaration
Class constructor ($path : Text)
  This.filePath:=$path

// This function will be called on each event of the offscreen area
Function onEvent()
  Case of
    :(FORM Event.code=On VP Ready)
      VP IMPORT DOCUMENT(This.area;This.filePath)
      This.result:=VP Get value(VP Cell(This.area;6;22))

      ALERT("The G23 cell contains the value: "+String(This.result))
  End case
```

La méthode callback (de rétro-appel) *OffscreenArea* :

```
$o:=cs.OffscreenArea.new()
$result:=VP Run offscreen area($o)
```

Exemple 2

Vous souhaitez charger un grand document hors écran, attendre que tous les calculs soient terminés et l'exporter au format PDF :

```

//Déclaration de la classe OffscreenArea
Class constructor($pdfPath : Text)
    This.pdfPath:=$pdfPath
    This.autoQuit:=False
    This.isWaiting:=False

Function onEvent()
    Case of
        :(FORM Event.code=On VP Ready)
        // Import du document
            VP IMPORT DOCUMENT(This.area;$largeDocument4VP)
            This.isWaiting:=True

        // Démarrer un minuteur pour vérifier si tous les calculs sont terminés.
        // Si pendant cette période le "On VP Range Changed" est lancé, le minuteur sera redémarré
        // L'heure doit être définie en fonction de la configuration de l'ordinateur.
        SET TIMER(60)

        :(FORM Event.code=On VP Range Changed)
        // Fin du calcul détectée. Restarts the timer
        If(This.isWaiting)
            SET TIMER(60)
        End if

        :(FORM Event.code=On Timer)
        // To be sure to not restart the timer if you call others 4D View command after this point
        This.isWaiting:=False

        // Stop the timer
        SET TIMER(0)

        // Start the PDF export
        VP EXPORT DOCUMENT(This.area;This.pdfPath;New object("formula";Formula(ACCEPT)))

        :(FORM Event.code=On URL Loading Error)
        CANCEL
    End case

```

La méthode callback (de rétro-appel) *OffscreenArea* :

```

$o:=cs.OffscreenArea.new()

$result:=VP Run offscreen area($o)

```

Voir aussi

[Blog post : End of document loading](#)

S

VP SET ACTIVE CELL

VP SET ACTIVE CELL (*rangeObj* : Object)

Paramètres	Type		Description
rangeObj	Object	->	Objet plage

Description

La commande `VP SET ACTIVE CELL` définit comme active une cellule spécifique.

Dans `rangeObj`, passez une plage contenant une seule cellule en tant qu'objet (reportez-vous à [VP Cell](#)). Si `rangeObj` n'est pas une plage de cellule ou contient plusieurs plages, seule la première cellule de la première plage est utilisée.

Exemple

Le code suivant déterminera comme active la cellule de la colonne D, ligne 5 :

```
$activeCell:=VP Cell("myVPArea";3;4)  
VP SET ACTIVE CELL($activeCell)
```

A	B	C	D	E	F	G
1						
2	Month	Area	Mr. Smith	Ms. Johnson	Ms. Williams	Mr. Jones
3	January	South	1898	1857	1651	1448
4		East	4859	4857	2548	4876
5		North	2458	1524	6150	4987
6		West	5787	1580	3975	4878
7	Total		15002	9818	14324	16189
8	February	South	6668	4374	17495	9999
9		East	5955	1677	7944	9400
10		North	1000	6722	2195	2777
11		West	6896	8355	7195	2058
12	Total		20519	21128	34829	24234
13	March	South	2577	2000	6185	2704
14		East	4859	4857	2548	4876
15		North	2458	1524	6150	4987
16		West	5787	1580	3975	4878
17	Total		15681	9961	18858	17115

Voir aussi

[VP ADD SELECTION](#)

[VP Get active cell](#)

[VP Get selection](#)

[VP RESET SELECTION](#)

[VP SET SELECTION](#)

[VP SHOW CELL](#)

VP SET ALLOWED METHODS

VP SET ALLOWED METHODS (`methodObj` : Object)

Paramètres	Type		Description
<code>methodObj</code>	Object	->	Méthodes autorisées dans les zones 4D View Pro

Compatibilité

Pour plus de flexibilité, il est recommandé d'utiliser la commande [VP SET CUSTOM FUNCTIONS](#) qui vous permet de désigner des formules 4D pouvant être appelées depuis des zones 4D View Pro. As soon as [VP SET CUSTOM FUNCTIONS](#) is called, [VP SET ALLOWED METHODS](#) calls are ignored. 4D View Pro prend également en charge la commande générique de 4D [SET ALLOWED METHODS](#) dans le cas où ni [VP SET ALLOWED METHODS](#) ni [VP SET ALLOWED METHODS](#) ne sont pas appelées.

Description

La commande `VP SET ALLOWED METHODS` désigne les méthodes projets qui peuvent être appelées dans des formules 4D View Pro. Cette commande s'applique à toutes les zones 4D View Pro qui ont été créées après l'appel de la commande durant la session. Elle peut être appelée à plusieurs reprises dans la même session pour créer différentes configurations.

Par défaut, à des fins de sécurité, si vous n'exécutez pas la commande `VP SET ALLOWED METHODS`, aucun appel à une méthode n'est autorisé dans les zones 4D View Pro -- sauf si la commande générique de 4D, `SET ALLOWED METHODS`, a été utilisée (voir la note de compatibilité). L'utilisation d'une méthode non autorisée dans une formule affiche une erreur `#NAME?` dans la zone 4D View Pro.

Dans le paramètre `methodObj`, passez un objet dans lequel chaque propriété porte le nom d'une fonction à définir dans les zones 4D View Pro :

Propriété		Type	Description
<code><functionName></code>		Object	Description de la fonction personnalisée. The <code><functionName></code> property name defines the name of the custom function to display in 4D View Pro formulas (no spaces allowed)
	<code>method</code>	Text	(obligatoire) Nom de la méthode projet 4D existante à autoriser
	<code>parameters</code>	Collection d'objets	Collection de paramètres (dans l'ordre dans lequel ils sont définis dans la méthode).
	<code>[].name</code>	Text	Nom d'un paramètre à afficher dans <code><functionName></code> . Note : Les noms de paramètres ne doivent pas contenir de caractères espace.
	<code>[].type</code>	Nombre	Type de paramètre. Supported types: <ul style="list-style-type: none"> • <code>Is Boolean</code> • <code>Is date</code> • <code>Is Integer</code> • <code>Is object</code> • <code>Is real</code> • <code>Is text</code> • <code>Is time</code> S'il est omis, par défaut la valeur est automatiquement envoyée avec son type, exceptées les valeurs date ou heure qui sont envoyées sous forme d'objet (voir la section Paramètres). Si type est défini sur <code>Is object</code> , l'objet possède la même structure que l'objet retourné par VP Get value .
	<code>summary</code>	Text	Description de la fonction à afficher dans 4D View Pro
	<code>minParams</code>	Nombre	Nombre minimum de paramètres
	<code>maxParams</code>	Nombre	Nombre maximum de paramètres. Si vous passez un nombre supérieur à la largeur de parameters, il est possible de déclarer des paramètres "optionnels" avec leur type par défaut

Exemple

Vous souhaitez autoriser deux méthodes dans vos zones 4D View Pro :

```

C_OBJECT($allowed)
$allowed:=New object //paramètre pour la commande

$allowed.Hello:=New object //crée une première fonction simple nommée "Hello"
$allowed.Hello.method:="My_Hello_Method" //définit la méthode 4D
$allowed.Hello.summary:="Hello prints hello world"

$allowed.Byebye:=New object //crée une deuxième fonction avec des paramètres nommée "Byebye"
$allowed.Byebye.method:="My_ByeBye_Method"
$allowed.Byebye.parameters:=New collection
$allowed.Byebye.parameters.push(New object("name";"Message";"type";Is text))
$allowed.Byebye.parameters.push(New object("name";"Date";"type";Is date))
$allowed.Byebye.parameters.push(New object("name";"Time";"type";Is time))
$allowed.Byebye.summary:="Byebye prints a custom timestamp"
$allowed.Byebye.minParams:=3
$allowed.Byebye.maxParams:=3

VP SET ALLOWED METHODS($allowed)

```

Une fois ce code exécuté, les fonctions définies peuvent être utilisées dans des formules 4D View Pro :

	A	B	C	D	E
1	=BYEBYE()				
2		BYEBYE(Message; Date; Time)			
3		Summary			
4		Byebye prints a custom timestamp			
5					

La numérotation démarre à 0.

Voir aussi

[4D functions](#)

[VP SET CUSTOM FUNCTIONS](#)

VP SET BINDING PATH

► Historique

VP SET BINDING PATH (*rangeObj* : Object ; *dataContextAttribute* : Text)

Paramètres	Type		Description
<i>rangeObj</i>	Object	->	Objet plage
<i>dataContextAttribute</i>	Text	->	Name of the attribute to bind to <i>rangeObj</i>

Description

The **VP SET BINDING PATH** command binds an attribute from a sheet's data context to *rangeObj*. After you set a data context using the [SET DATA CONTEXT](#) method. When loaded, if the data context contains the attribute, the value of *dataContextAttribute* is automatically displayed in the cells in *rangeObj*.

In *rangeObj*, pass an object that is either a cell range or a combined range of cells.

- If *rangeObj* is a range with several cells, the command binds the attribute to the first cell of the range.
- If *rangeObj* contains several ranges of cells, the command binds the attribute to the first cell of each range.

In *dataContextAttribute*, pass the name of the attribute to bind to *cellRange*. If *dataContextAttribute* is an empty string, the function removes the current binding.

Attributes of type collection are not supported. When you pass the name of a collection attribute, the command

does nothing.

Exemple

Set a data context and bind the `firstName` and `lastName` attribute to cells:

```
var $p : Object  
  
$p:=New object  
$p.firstName:="Freehafer"  
$p.lastName:="Nancy"  
  
VP SET DATA CONTEXT("ViewProArea"; $p)  
  
VP SET BINDING PATH(VP Cell("ViewProArea"; 0; 0); "firstName")  
VP SET BINDING PATH(VP Cell("ViewProArea"; 1; 0); "lastName")
```

	0	1	2	3
1	Freehafer	Nancy		
2				
3				

Voir aussi

[VP Get binding path](#)

[VP Get data context](#)

[VP SET DATA CONTEXT](#)

VP SET BOOLEAN VALUE

VP SET BOOLEAN VALUE (*rangeObj* : Object ; *boolValue* : Boolean)

Paramètres	Type		Description
<i>rangeObj</i>	Object	->	Objet plage
<i>boolValue</i>	Booléen	->	Valeur du booléen à fixer

Description

La commande `VP SET BOOLEAN VALUE` assigne une valeur booléenne spécifique à une plage de cellule désignée.

Dans *rangeObj*, passez la plage de cellule(s) (créée par exemple avec `VP Cell` ou `VP Column`) dont vous souhaitez indiquer la valeur. Si *rangeObj* comprend plusieurs cellules, la valeur indiquée sera répétée dans chaque cellule.

Le paramètre *boolValue* vous permet de passer la valeur booléenne (`True` ou `False`) qui sera assignée à *rangeObj*.

Exemple

```
//Fixez la valeur de la cellule à False  
VP SET BOOLEAN VALUE(VP Cell("ViewProArea";3;2);False)
```

Voir aussi

[VP SET VALUE](#)

VP SET BORDER

VP SET BORDER (*rangeObj* : Object ; *borderStyleObj* : Object ; *borderPosObj* : Object)

Paramètres	Type		Description
rangeObj	Object	->	Objet plage
borderStyleObj	Object	->	Objet contenant le style de bordure
borderPosObj	Object	->	Objet contenant la position de la bordure

Description

La commande `VP SET BORDER` applique le(s) style(s) de bordure défini(s) dans `borderStyleObj` et `borderPosObj` à la plage définie dans `rangeObj`.

Dans `rangeObj`, passez une plage de cellules à laquelle s'appliquera le style de bordure. Si `rangeObj` contient plusieurs cellules, les bordures appliquées avec `VP SET BORDER` seront appliquées à `rangeObj` dans son intégralité (contrairement à la commande `VP SET CELL STYLE` qui les applique à chaque cellule de `rangeObj`). Si une feuille de style a déjà été appliquée, `VP SET BORDER` remplaceront les paramètres de bordure appliqués antérieurement à `rangeObj`.

Le paramètre `borderStyleObj` vous permet de définir le style des lignes de la bordure. `borderStyleObj` prend en charge les propriétés suivantes :

Propriété	Type	Description	Possible values
color	Texte	Defines the color of the border. Default = black.	CSS color "#rrggbb" syntax (preferred syntax), CSS color "rgb(r,g,b)" syntax (alternate syntax), CSS color name (alternate syntax)
style	Integer	Defines the style of the border. Default = empty.	<ul style="list-style-type: none"> • <code>vk line style dash dot</code> • <code>vk line style dash dot dot</code> • <code>vk line style dashed</code> • <code>vk line style dotted</code> • <code>vk line style double</code> • <code>vk line style empty</code> • <code>vk line style hair</code> • <code>vk line style medium</code> • <code>vk line style medium dash dot</code> • <code>vk line style medium dash dot dot</code> • <code>vk line style medium dashed</code> • <code>vk line style slanted dash dot</code> • <code>vk line style thick</code> • <code>vk line style thin</code>

Les bordures appliquées à l'aide de `VP SET CELL STYLE` seront appliquées à chaque cellule de `rangeObj`, contrairement à la commande `VP SET BORDER` qui applique les bordures à l'ensemble de `rangeObj`.

Propriété	Type	Description
all	boolean	Style de la ligne de bordure appliquée à toutes les bordures.
left	boolean	Style de la ligne de bordure appliquée à la bordure de gauche.
top	boolean	Style de la ligne de bordure appliquée à la bordure supérieure.
right	boolean	Style de la ligne de bordure appliquée à la bordure de droite.
bottom	boolean	Style de la ligne de bordure appliquée à la bordure inférieure.
outline	boolean	Style de la ligne de bordure appliquée uniquement aux bordures extérieures.
inside	boolean	Style de la ligne de bordure appliquée uniquement aux bordures intérieures.
innerHorizontal	boolean	Style de la ligne de bordure appliquée uniquement aux bordures horizontales intérieures.
innerVertical	boolean	Style de la ligne de bordure appliquée uniquement aux bordures verticales intérieures.

Exemple 1

Ce code applique la bordure suivante autour de la plage :

```
$border:=New object("color";"red";"style";vk line style thick)
$option:=New object("outline";True)
VP SET BORDER(VP Cells("ViewProArea";1;1;3;3);$border;$option)
```

	A	B	C	D	E
1					
2					
3					
4					
5					
6					

Exemple 2

Ce code illustre, en termes de définition des bordures, la différence entre la commande `VP SET BORDER` et la commande `VP SET CELL STYLE` :

```
// Appliquer des bordure à l'aide de VP SET BORDER
$border:=New object("color";"red";"style";vk line style thick)
$option:=New object("outline";True)
VP SET BORDER(VP Cells("ViewProArea";1;1;3;3);$border;$option)

// // Appliquer des bordures à l'aide de VP SET CELL STYLE
$cellStyle:=New object
$cellStyle.borderBottom:=New object("color";"blue";"style";vk line style thick)
$cellStyle.borderRight:=New object("color";"blue";"style";vk line style thick)
VP SET CELL STYLE(VP Cells("ViewProArea";4;4;3;3);$cellStyle)
```

	A	B	C	D	E	F	G	H
1								
2								
3								
4								
5								
6								
7								
8								

Voir aussi

[VP SET CELL STYLE](#)

VP SET CELL STYLE

VP SET CELL STYLE (*rangeObj* : Object ; *styleObj* : Object)

Paramètres	Type		Description
<i>rangeObj</i>	Object	->	Objet plage
<i>styleObj</i>	Object	->	Objet style

Description

La commande `VP SET CELL STYLE` applique le(s) style(s) défini(s) dans *styleObj* aux cellules définies dans *rangeObj*.

Dans *rangeObj*, passez une plage de cellules à laquelle s'appliquera le style. Si *rangeObj* contient plusieurs cellules, le style s'applique à chaque cellule.

Si *rangeObj* n'est pas une plage cellule, seule la première cellule de la plage est utilisée.

Le paramètre *styleObj* vous permet de passer un objet contenant des propriétés de style. Vous pouvez utiliser une feuille de style existante ou créer un nouveau style. Si *styleObj* contient à la fois une feuille de style existante et des propriétés de style supplémentaires, la feuille de style existante s'applique, suivie des propriétés supplémentaires.

Attribuez la valeur NULL pour supprimer un style et rétablir les paramètres de style par défaut (le cas échéant) :

- attribuer la valeur NULL à *styleObj* supprimera toutes les propriétés de style de *rangeObj*,
- attribuer la valeur NULL à un attribut supprimera l'attribut spécifique de *rangeObj*.

Pour plus d'informations sur les objets style et les feuilles de style, consultez le paragraphe [Objets style](#).

Exemple

```
$style:=New object
$style.font:="8pt Arial"
$style.backColor:="Azure"
$style.foreColor:="red"
$style.hAlign:=1
$style.isVerticalText:=True
$style.borderBottom:=New object("color";"#800080";"style";vk line style thick)
$style.backgroundImage:=Null //supprimer un attribut spécifique

VP SET CELL STYLE(VP Cell("ViewProArea";1;1);$style)
```

	A	B	C
1	H e l o		
2		W o r d	
3			

Voir aussi

[VP ADD STYLESHEET](#)

[VP Font to object](#)

[VP Get cell style](#)

[VP Object to font](#)

[VP SET BORDER](#)

[VP SET DEFAULT STYLE](#)

VP SET COLUMN ATTRIBUTES

VP SET COLUMN ATTRIBUTES (*rangeObj* : Object ; *propertyObj* : Object)

Paramètres	Type		Description
rangeObj	Object	->	Objet plage
propertyObj	Object	->	Objet contenant des propriétés de colonnes

Description

La commande **VP SET COLUMN ATTRIBUTES** applique les attributs définis dans le paramètre *propertyObj* pour les colonnes de *rangeObj*.

Dans *rangeObj*, passez un objet contenant une plage. Si la plage contient des colonnes et des lignes, les attributs s'appliquent uniquement aux colonnes.

Le paramètre *propertyObj* vous permet de renseigner les attributs à appliquer aux colonnes de *rangeObj*. Ces attributs sont :

Propriété	Type	Description
width	number	Largeur de colonne exprimée en pixels
pageBreak	boolean	Vrai pour insérer un saut de page avant la première colonne de la plage, sinon faux
visible	boolean	Vrai si la colonne est visible, sinon faux
resizable	boolean	Vrai si la colonne peut être redimensionnée, sinon faux
header	Texte	Texte de l'en-tête de la colonne

Exemple

Pour modifier la taille de la deuxième colonne et définir un en-tête, le code suivant :

```
C_OBJECT($column;$properties)

$column:=VP Column("ViewProArea";1) //colonne B
$properties:=New object("width";100;"header";"Hello World")

VP SET COLUMN ATTRIBUTES($column;$properties)
```

	A	Hello World	C	D	E	F	G	H	I
1	The	quick	brown	fox	jumped	over	the	lazy	dog

Voir aussi

[VP Column](#)

[VP Get column attributes](#)

[VP Get row attributes](#)

[VP SET ROW ATTRIBUTES](#)

VP SET COLUMN COUNT

VP SET COLUMN COUNT (*vpAreaName* : Text , *columnCount* : Integer { , *sheet* : Integer })

Paramètres	Type		Description
<i>vpAreaName</i>	Text	->	Nom d'objet formulaire zone 4D View Pro
<i>columnCount</i>	Integer	->	Nombre de colonnes
<i>sheet</i>	Integer	->	Indice de la feuille (feuille courante si omis)

Description

La commande `VP SET COLUMN COUNT` définit le nombre total de colonnes dans *vpAreaName*.

Dans *vpAreaName*, passez le nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est retournée.

Passez le nombre total de colonnes dans le paramètre *columnCount*. *columnCount* doit être supérieur à 0.

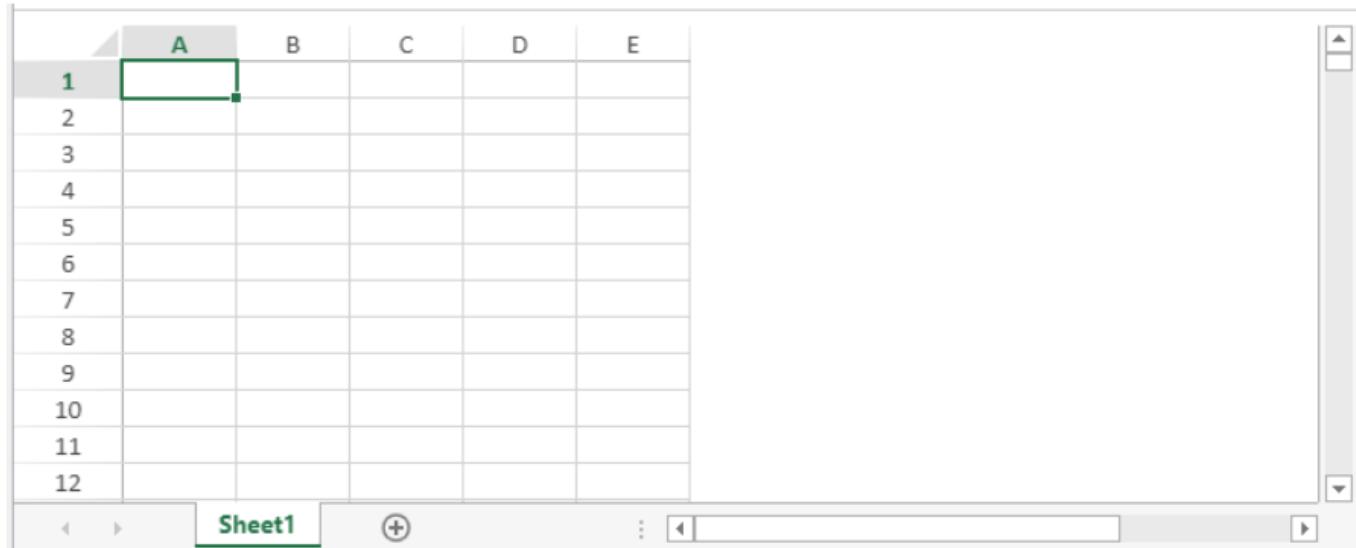
Dans le paramètre optionnel *sheet*, vous pouvez désigner une feuille de calcul spécifique dans laquelle s'appliquera *columnCount* (la numérotation démarre à zéro). Si le paramètre est omis, la feuille courante est utilisée par défaut. Vous pouvez sélectionner explicitement la feuille courante à l'aide de la constante suivante :

- `vk current sheet`

Exemple

Le code suivant définit cinq colonnes dans la zone 4D View Pro :

```
VP SET COLUMN COUNT("ViewProArea";5)
```



Voir aussi

[VP Get column count](#)

[VP Get row count](#)

[VP SET ROW COUNT](#)

VP SET CURRENT SHEET

VP SET CURRENT SHEET (*vpAreaName* : Text ; *index* : Integer)

Paramètres	Type		Description
<i>vpAreaName</i>	Text	->	Nom d'objet formulaire zone 4D View Pro
<i>index</i>	Integer	<-	Index of the new current sheet

Description

The `VP SET CURRENT SHEET` command sets the current sheet in `vpAreaName`. La feuille courante est la feuille sélectionnée dans le document.

Dans `vpAreaName`, passez le nom de la zone 4D View Pro.

In `index`, pass the index of the sheet to be set as current sheet. If the index passed is inferior to 0 or exceeds the number of sheets, the command does nothing.

La numérotation démarre à 0.

Exemple

The document's current sheet is the first sheet:



Set the current sheet to the third sheet:

```
VP SET CURRENT SHEET("ViewProArea";2)
```



Voir aussi

[VP Get current sheet](#)

VP SET CUSTOM FUNCTIONS

`VP SET CUSTOM FUNCTIONS (vpAreaName : Text ; formulaObj : Object)`

Paramètres	Type		Description
<code>vpAreaName</code>	Text	->	Nom d'objet formulaire zone 4D View Pro
<code>formulaObj</code>	Object	->	Objet formule

Description

La commande `VP SET CUSTOM FUNCTIONS` désigne les formules 4D qui peuvent être appelées directement à partir de formules 4D View Pro. Les fonctions personnalisées n'étant pas stockées dans le document, la commande `VP SET CUSTOM FUNCTIONS` doit être exécutée dans l'événement formulaire `On Load`.

Les formules spécifiées par `VP SET CUSTOM FUNCTIONS` apparaissent dans un menu pop-up lorsque la première lettre de leur nom est saisie. Voir la page [Formules et Fonctions](#).

La numérotation démarre à 0.

Passez le nom de la zone 4D View Pro dans `vpAreaName`. Si vous passez un nom inexistant, une erreur est retournée.

Dans le paramètre `formulaObj`, passez un objet contenant les formules 4D pouvant être appelées par des formules 4D View Pro et des propriétés supplémentaires. Chaque propriété `customFunction` passée dans `formulaObj` devient le nom d'une fonction dans la zone 4D View Pro.

Propriété		Type	Description
<customFunction>		Object	Description de la fonction personnalisée. <customFunction> defines the name of the custom function to display in 4D View Pro formulas (no spaces allowed)
	formula	Object	Objet formule 4D (obligatoire). Voir la commande Formula .
	parameters	Collection d'objets	Collection de paramètres (dans l'ordre dans lequel ils sont définis dans la formule)
	[].name	Text	Nom du paramètre à afficher dans 4D View Pro
	[].type	Nombre	<p>Type de paramètre. Supported types:</p> <ul style="list-style-type: none"> • Is Boolean • Is date • Is Integer • Is object • Is real • Is text • Is time <p>Si <code>type</code> est omis ou si la valeur par défaut (-1) est passée, la valeur est automatiquement envoyée avec son type, à l'exception des valeurs date ou heure qui sont envoyées en tant qu'objet (voir la section Paramètres).</p> <p>Si <code>type</code> est défini sur <code>Is object</code>, l'objet a la même structure que l'objet retourné par VP Get value.</p>
	summary	Text	Description de la formule à afficher dans 4D View Pro
	minParams	Nombre	Nombre minimum de paramètres
	maxParams	Nombre	Nombre maximum de paramètres. Passer un nombre supérieur à la longueur de <code>parameters</code> permet de déclarer les paramètres "optionnels" avec un type par défaut

ATTENTION

- Dès que `VP SET CUSTOM FUNCTIONS` est appelé, les fonctions basées sur les commandes `SET TABLE TITLES` et `SET FIELD TITLES` sont ignorées dans la zone 4D View Pro.
- Dès que `VP SET CUSTOM FUNCTIONS` est appelé, les fonctions basées sur les commandes `SET TABLE TITLES` et `SET FIELD TITLES` sont ignorées dans la zone 4D View Pro.

Exemple

Vous souhaitez utiliser des objets Formule dans une zone 4D View Pro pour ajouter des chiffres, récupérer le sexe et le nom de famille d'un client :

```

Case of
:(FORM Event.code=On Load)

var $o : Object
$o:=New object

// Définir la fonction "addnum" d'une méthode nommée "addnum"
$o.addnum:=New object
$o.addnum.formula:=Formula(addnum)
$o.addnum.parameters:=New collection
$o.addnum.parameters.push(New object("name";"num1";"type";Is Integer))
$o.addnum.parameters.push(New object("name";"num2";"type";Is Integer))

// Définir la fonction "ClientLastName" d'un champ de base de données
$o.ClientLastName:=New object
$o.ClientLastName.formula:=Formula([Customers]lastname)
$o.ClientLastName.summary:="Nom de famille du client courant"

// Définir la fonction "label" d'une expression 4D avec un paramètre
$o.label:=New object
$o.label.formula:=Formula(ds.Customers.getLabel($1).label)
$o.label.parameters:=New collection
$o.label.parameters.push(New object("name";"ID";"type";Is Integer))

// Définir la fonction "Title" d'une variable nommée "Title"
$o.Title:=New object
$o.Title.formula:=Formula(Title)

VP SET CUSTOM FUNCTIONS("ViewProArea";$o)

```

End case

Voir aussi

[VP SET ALLOWED METHODS](#)

VP SET DATA CONTEXT

► Historique

VP SET DATA CONTEXT (*vpAreaName* : Text ; *dataObj* : Object {; *options* : Object {; *sheetIndex* : Integer}})
VP SET DATA CONTEXT (*vpAreaName* : Text ; *dataColl* : Collection ; { *options* : Object {; *sheetIndex* : Integer}})

Paramètres	Type		Description
<i>vpAreaName</i>	Object	->	Nom d'objet formulaire zone 4D View Pro
<i>dataObj</i>	Object	->	Data object to load in the data context
<i>dataColl</i>	Object	->	Data collection to load in the data context
<i>options</i>	Object	->	Options supplémentaires
<i>sheetIndex</i>	Integer	->	Indice de la feuille

Description

The **VP SET DATA CONTEXT** command sets the data context of a sheet. A data context is an object or a collection bound to a worksheet, and whose contents can be used to automatically fill the sheet cells, either by using an autogenerate option or the **VP SET BINDING PATH** method. On the other hand, the **VP Get data context** command can return a context containing user modifications.

Dans *vpAreaName*, passez le nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est retournée.

In *dataObj* or *dataColl*, pass an object or a collection containing the data to load in the data context. Images are

converted to data URI schemes.

To pass a time value in *dataObj* or *dataColl*, encapsulate it in an object with the following properties (see [example 4](#)):

Propriété	Type	Description
value	Integer, Real, Boolean, Text, Date, Null	Value to put in the context
time	Réel	Time value (in seconds) to put in the context

In *options*, you can pass an object that specifies additional options. Possible properties are:

Propriété	Type	Description
reset	Object	True to reset the sheet's contents before loading the new context, False (default) otherwise.
autoGenerateColumns	Object	Only used when data is a collection. True (default) to specify that columns must be generated automatically when the data context is bound. In this case, the following rules apply: <ul style="list-style-type: none">If <i>dataColl</i> is a collection of objects, attribute names are used as column titles (see example 2).If <i>dataColl</i> contains subcollections of scalar values, each subcollection defines the values in a row (see example 3). The first subcollection determines how many columns are created.

In *sheetIndex*, pass the index of the sheet that will receive the data context. If no index is passed, the context is applied to the current sheet.

If you export your document to an object using [VP Export to object](#), or to a 4DVP document using [VP EXPORT DOCUMENT](#), the `includeBindingSource` option lets you copy the contents of the current contexts as cell values in the exported object or document. For more details, refer to the description of those methods.

Exemple

Pass an object and bind the context data to cells in the first row:

```
var $data : Object  
  
$data:=New object  
$data.firstName:="Freehafer"  
$data.lastName:="Nancy"  
  
VP SET DATA CONTEXT("ViewProArea"; $data)  
  
VP SET BINDING PATH(VP Cell("ViewProArea"; 0; 0); "firstName")  
VP SET BINDING PATH(VP Cell("ViewProArea"; 1; 0); "lastName")
```

	0	1	2	3
1	Freehafer	Nancy		
2				
3				

Exemple 2

Pass a collection of objects and generate columns automatically:

```

var $options : Object
var $data : Collection

$data:=New collection()
$data.push(New object("firstname"; "John"; "lastname"; "Smith"))
$data.push(New object("firstname"; "Mary"; "lastname"; "Poppins"))

$options:=New object("autoGenerateColumns"; True)

VP SET DATA CONTEXT("ViewProArea"; $data; $options)

```

	A1	:	X ✓ fx	
	firstname	lastname		
1	John	Smith		
2	Mary	Poppins		

Exemple 3

The *data* passed as a parameter is a collection that contains subcollections. Each subcollection defines the contents of a row:

```

var $data : Collection
var $options : Object

$data:=New collection
$data.push(New collection(1; 2; 3; False; ""))
$data.push(New collection) // Second row is empty
$data.push(New collection(4; 5; Null; "hello"; "world")) // Third row has 5 values
$data.push(New collection(6; 7; 8; 9)) // Fourth row has 4 values

$options:=New object("autoGenerateColumns"; True)

VP SET DATA CONTEXT("ViewProArea"; $data; $options)

```

	0	1	2	3	4
1	1	2	3	FALSE	
2					
3	4	5		hello	world
4	6	7	8	9	

Example 4 - Date and time syntax

```

var $data : Collection
var $options : Object

$data:= New collection()

// Dates can be passed as scalar values
$data.push(New collection("Date"; Current date))

// Time values must be passed as object attributes
$data.push(New collection("Time"; New object("time"; 5140)))

// Date + time example
$data.push(New collection("Date + Time"; New object("value"; Current date; "time"; 5140)))

$options:=New object("autoGenerateColumns"; True)

VP SET DATA CONTEXT("ViewProArea"; $data; $options)

```

Here's the result once the columns are generated:

	0	1
1	Date	2/22/2022
2	Time	1:25:40 AM
3	Date + Time	2/22/2022 01:25:40

Voir aussi

[VP SET BINDING PATH](#)

[VP Get binding path](#)

[VP Get data context](#)

VP SET DATE TIME VALUE

VP SET DATE TIME VALUE (*rangeObj* : Object ; *dateValue* : Date ; *timeValue* : Time {; *formatPattern* : Text })

Paramètres	Type		Description
<i>rangeObj</i>	Object	->	Objet plage
<i>dateValue</i>	Date	->	Valeur date à fixer
<i>timeValue</i>	Heure	->	Valeur heure à fixer
<i>formatPattern</i>	Text	->	Format de la valeur

Description

La commande `VP SET DATE TIME VALUE` assigne une valeur date et heure spécifique à une plage de cellule désignée.

Dans *rangeObj*, passez la plage de cellule(s) (créée par exemple avec `VP Cell` ou `VP Column`) dont vous souhaitez indiquer la valeur. Si *rangeObj* comprend plusieurs cellules, la valeur indiquée sera répétée dans chaque cellule.

Le paramètre *dateValue* indique une valeur date à assigner à *rangeObj*.

Le paramètre *timeValue* indique une valeur heure (exprimée en secondes) à assigner à *rangeObj*.

Le paramètre optionnel *formatPattern* définit un modèle pour les paramètres *dateValue* et *timeValue*. Pour plus d'informations sur les modèles et les caractères de formatage, veuillez consulter la section [Format date et heure](#).

Exemple

```

//Attribuer la date et l'heure locales à la valeur de la cellule
VP SET DATE TIME VALUE(VP Cell("ViewProArea";6;2);Current time;Current date;vk pattern full date time)

//Attribuer le 18 décembre à la valeur de la cellule
VP SET DATE TIME VALUE(VP Cell("ViewProArea";3;9);!2024-12-18!;?14:30:10?;vk pattern sortable date time

```

Voir aussi

[4D View Pro cell format](#)
[VP SET DATE VALUE](#)
[VP SET TIME VALUE](#)
[VP SET VALUE](#)

VP SET DATE VALUE

VP SET DATE VALUE (*rangeObj* : Object ; *dateValue* : Date { ; *formatPattern* : Text })

Paramètres	Type		Description
<i>rangeObj</i>	Object	->	Objet plage
<i>dateValue</i>	Date	->	Valeur date à fixer
<i>formatPattern</i>	Text	->	Format de la valeur

Description

La commande `VP SET DATE VALUE` assigne une valeur date à une plage de cellules désignée.

Dans *rangeObj*, passez la plage de cellule(s) dont vous souhaitez indiquer la valeur. Si *rangeObj* comprend plusieurs cellules, la valeur indiquée sera répétée dans chaque cellule.

Le paramètre *dateValue* indique une valeur date à assigner à *rangeObj*.

Le paramètre optionnel *formatPattern* définit un modèle pour le paramètre *dateValue*. Passez un format personnalisé ou utilisez ou utilisez l'une des constantes suivantes :

Constante	Description	Configuration par défaut des US
<code>vk pattern long date</code>	Format ISO 8601 long pour la date	"ddd, dd MMMM yyyy"
<code>vk pattern month day</code>	Format ISO 8601 pour le mois et le jour	"MMMM dd"
<code>vk pattern short date</code>	Format ISO 8601 court pour la date	"MM/dd/yyyy"
<code>vk pattern year month</code>	Format ISO 8601 pour le mois et l'année	"yyyy MMMM"

Pour plus d'informations sur les modèles et les caractères de formatage, veuillez consulter la section [Format date et heure](#).

Exemple

```

//Définir la valeur de la cellule à la date du jour
VP SET DATE VALUE(VP Cell("ViewProArea";4;2);Current date)

//Définir la valeur de la cellule à une date spécifique avec un format désigné
VP SET DATE VALUE(VP Cell("ViewProArea";4;4);Date("12/25/94");"d/m/yy ")
VP SET DATE VALUE(VP Cell("ViewProArea";4;6);!2005-01-15!;vk pattern month day)

```

Voir aussi

[4D View Pro cell format](#)

[VP SET DATE TIME VALUE](#)

[VP SET VALUE](#)

VP SET DEFAULT STYLE

VP SET DEFAULT STYLE (*vpAreaName* : Text ; *styleObj* : Object { ; *sheet* : Integer })

Paramètres	Type		Description
<i>vpAreaName</i>	Text	->	Nom d'objet formulaire zone 4D View Pro
<i>styleObj</i>	Object	->	Objet style
<i>sheet</i>	Integer	->	Indice de la feuille (par défaut = feuille courante)

Description

La commande `VP SET DEFAULT STYLE` définit, dans le paramètre *styleObj*, le style d'une feuille (*sheet*) en tant que style par défaut.

Dans *vpAreaName*, passez le nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est retournée.

Le paramètre *styleObj* vous permet de passer un objet contenant des paramètres de style. Vous pouvez utiliser une feuille de style existante ou créer un nouveau style. Pour plus d'informations, consultez [Objets style](#).

Dans le paramètre optionnel *sheet*, vous pouvez désigner une feuille spécifique dans laquelle le style sera défini. Si le paramètre est omis, la feuille courante est utilisée par défaut. Vous pouvez sélectionner explicitement la feuille courante à l'aide de la constante suivante :

- `vk current sheet`

Exemple

```
$style:=New object
$style.hAlign:=vk horizontal align left
$style.font:="12pt papyrus"
$style.backColor:="#E6E6FA" //couleur light purple

VP SET DEFAULT STYLE("myDoc";$style)
```

The screenshot shows a Microsoft Excel spreadsheet with a single cell selected. The cell contains the text "Hello World!". The row number is 2 and the column letter is A. The background color of the cell is light purple, matching the style defined in the example code. The rest of the table has white rows and columns.

	A	B	C
1			
2	Hello World!		
3			
4			
5			
6			
7			
8			

Voir aussi

[VP ADD STYLESHEET](#)

[VP Font to object](#)

[VP Get default style](#)

[VP Object to font](#)

[VP SET BORDER](#)

[VP SET CELL STYLE](#)

VP SET FIELD

VP SET FIELD (*rangeObj* : Object ; *field* : Pointer { ; *formatPattern* : Text })

Paramètres	Type		Description
<i>rangeObj</i>	Object	->	Objet plage
<i>field</i>	Pointeur	->	Référence au champ dans la structure virtuelle
<i>formatPattern</i>	Text	->	Format du champ

Description

La commande `VP SET FIELD` assigne un champ spécifique de la base 4D à une plage de cellules désignée.

Dans *rangeObj*, passez la plage de cellule(s) dont vous souhaitez indiquer la valeur. Dans *rangeObj*, passez la plage de cellule(s) dont vous souhaitez indiquer la valeur.

Le paramètre *field* indique un [champ virtuel](#) de la base 4D à assigner à *rangeObj*. Le nom de la structure virtuelle pour le *field* peut être visualisé dans la barre de formule. Si du contenu est déjà présent dans l'une des cellules de *rangeObj*, il sera remplacé par *rangeObj*.

Le paramètre optionnel *formatPattern* définit un modèle pour le paramètre *field*. You can pass any valid [custom format](#).

Exemple

```
VP SET FIELD(VP Cell("ViewProArea";5;2);->[TableName]Field)
```

Voir aussi

[VP SET VALUE](#)

VP SET FORMULA

VP SET FORMULA (*rangeObj* : Object ; *formula* : Text { ; *formatPattern* : Text })

Paramètres	Type		Description
<i>rangeObj</i>	Object	->	Objet plage
<i>formula</i>	Text	->	Formule ou méthode 4D
<i>formatPattern</i>	Text	->	Format du champ

Description

La commande `VP SET FORMULA` assigne une formule spécifique ou une méthode 4D à une plage de cellules désignée.

Dans *rangeObj*, passez la plage de cellule(s) (créeé par exemple avec [VP Cell](#) ou [VP Column](#)) dont vous souhaitez indiquer la valeur. Dans *rangeObj*, passez la plage de cellule(s) (créeé par exemple avec [VP Cell](#) ou [VP Column](#)) dont vous souhaitez indiquer la valeur.

Le paramètre *formula* indique un nom de formule ou de méthode 4D à assigner à *rangeObj*.

If the *formula* is a string, use the period `.` as numerical separator and the comma `,` as parameter separator.
Si une méthode 4D est utilisée, elle doit être autorisée à l'aide de la commande [VP SET ALLOWED METHODS](#).

Le paramètre optionnel *formatPattern* définit un [pattern](#) (modèle) pour le paramètre *formula*.

Vous pouvez supprimer la formule de *rangeObj* en la remplaçant par une chaîne vide ("").

Exemple 1

```
VP SET FORMULA(VP Cell("ViewProArea";5;2);"SUM($A$1:$C$10)")
```

Exemple 2

Pour supprimer la formule :

```
VP SET FORMULA(VP Cell("ViewProArea";5;2); "")
```

Exemple 3

```
VP SET FORMULA($range;"SUM(A1,B7,C11)") //"," to separate parameters
```

Voir aussi

[Cell format](#)
[VP Get Formula](#)
[VP SET FORMULAS](#)
[VP SET VALUE](#)

VP SET FORMULAS

VP SET FORMULAS (*rangeObj* : Object ; *formulasCol* : Collection)

Paramètres	Type		Description
<i>rangeObj</i>	Object	->	Cell range object
<i>formulasCol</i>	Collection	->	Collection de formules

Description

La commande `VP SET FORMULAS` affecte une collection de formules commençant par la plage de cellule spécifiée.

Dans *rangeObj*, passez une plage de la cellule (créée avec `VP Cell`) dont vous souhaitez indiquer la formule. Si *rangeObj* comprend plusieurs plages, seule la première plage est utilisée.

Le paramètre *formulasCol* est une collection bidimensionnelle :

- La collection de premier niveau contient des sous-collections de formules. Chaque sous-collection définit une ligne.
- Chaque sous-collection définit les valeurs des cellules de la ligne. Les valeurs doivent être des éléments textuels contenant les formules à associer aux cellules.

If the formula is a string, use the period `.` as numerical separator and the comma `,` as parameter separator.
Si une méthode 4D est utilisée, elle doit être autorisée à l'aide de la commande `VP SET ALLOWED METHODS`.

Vous pouvez supprimer les formules dans *rangeObj* en les remplaçant par une chaîne vide ("").

Exemple 1

```
$formulas:=New collection
$formulas.push(New collection("MAX(B11,C11,D11)";"myMethod(G4)")) // Première ligne
$formulas.push(New collection("SUM(B11:D11)";"AVERAGE(B11:D11)")) // Deuxième ligne

VP SET FORMULAS(VP Cell("ViewProArea";6;3);$formulas) // Associer les formules et les formules
```

myMethod :

$\$0:=\$1*3.33$

The screenshot shows a spreadsheet interface with a toolbar at the top. The formula bar displays the formula `=Mymethod(G4)`. Below the toolbar is a table with the following data:

Production Cost Analysis										
Retail Price Calculations										
	Product 1	Product 2	Product 3							
4	\$0.98	\$0.74	\$0.18		Highest Prod Cost	\$4.42	\$14.72	Retail Price		
5	\$0.23	\$0.83	\$0.46		Total Production Cost	\$11.38	\$3.79	Average Product Cost		
6	\$0.77	\$0.09	\$0.53							
7	\$0.74	\$0.11	\$0.47							
8	\$0.59	\$0.93	\$0.63							
9	\$0.85	\$0.65	\$0.34							
10	\$0.26	\$0.25	\$0.75							
11	Per Product Total	\$4.42	\$3.60	\$3.36						

Exemple 2

Pour supprimer des formules :

```

$formulas:=New collection
$formulas.push(New collection("", ""))
$formulas.push(New collection("", ""))
VP SET FORMULAS(VP Cell("ViewProArea";0;0);$formulas) // Assigner à des cellules

```

Voir aussi

[VP Get Formulas](#)
[VP GET VALUES](#)
[VP SET FORMULA](#)
[VP SET VALUES](#)

VP SET FROZEN PANES

`VP SET FROZEN PANES (vpAreaName : Text ; paneObj : Object { ; sheet : Integer })`

Paramètres	Type		Description
vpAreaName	Text	->	Nom d'objet formulaire zone 4D View Pro
paneObj	Object	->	Objet contenant des informations sur les colonnes et lignes figées
sheet	Integer	->	Indice de la feuille (feuille courante si omis)

Description

La commande `VP SET FROZEN PANES` commande fige les lignes et colonnes de `paneObj` afin qu'ils s'affichent constamment dans `vpAreaName`. Les lignes et colonnes figées sont statiques et ne bougent plus lorsque vous faites défiler le document. Un trait continu s'affiche pour indiquer que des lignes et colonnes sont figées. L'emplacement du trait dépend de l'emplacement de la ligne ou colonne figée dans la feuille :

- Colonnes de gauche ou de droite : Pour les colonnes situées à gauche de la feuille, le trait s'affiche sur le côté droit de la dernière colonne figée. Pour les colonnes situées à droite de la feuille, le trait s'affiche sur le côté gauche de la

première colonne figée.

- Lignes du haut ou du bas : Pour les lignes situées en haut de la feuille, le trait s'affiche en-dessous de la première ligne figée. Pour les lignes situées en bas de la feuille, le trait s'affiche au-dessus de la première ligne figée.

Dans `vpAreaName`, passez le nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est retournée.

Vous pouvez passer un objet définissant les lignes et colonnes à figer dans le paramètre `paneObj`. Si vous fixez la valeur d'une propriété de colonne ou de ligne à zéro, cela réinitialise (ne fige plus) la propriété. Si une propriété est définie sur une valeur inférieure à zéro, la commande ne fait rien. Vous pouvez passer :

Propriété	Type	Description
columnCount	Integer	Le nombre de colonnes figées sur la gauche de la feuille
trailingColumnCount	Integer	Le nombre de colonnes figées sur la droite de la feuille
rowCount	Integer	Le nombre de lignes figées en haut de la feuille
trailingRowCount	Integer	Le nombre de lignes figées en bas de la feuille

Dans le paramètre optionnel `sheet`, vous pouvez désigner une feuille spécifique dans laquelle sera définie la plage (la numérotation commence à zéro). Si le paramètre est omis, la feuille courante est utilisée par défaut. Vous pouvez sélectionner explicitement la feuille courante à l'aide de la constante suivante :

- `vk current sheet`

Exemple

Vous souhaitez figer les trois premières colonnes de gauche, deux colonnes de droite et la première ligne :

```
C_OBJECT($panes)  
  
$panes:=New object  
$panes.columnCount:=3  
$panes.trailingColumnCount:=2  
$panes.rowCount:=1  
  
VP SET FROZEN PANES("ViewProArea";$panes)
```



Voir aussi

[VP Get frozen panes](#)

VP SET NUM VALUE

`VP SET NUM VALUE (rangeObj : Object ; numberValue : Number { ; formatPattern : Text })`

Paramètres	Type		Description
rangeObj	Object	->	Objet plage
numberValue	Nombre	->	Valeur du nombre à fixer
formatPattern	Text	->	Format de la valeur

Description

La commande `VP SET NUM VALUE` assigne une valeur numérique spécifique à une plage de cellules désignée.

Dans `rangeObj`, passez la plage de cellule(s) (créée par exemple avec `VP Cell` ou `VP Column`) dont vous souhaitez indiquer la valeur. Si `rangeObj` comprend plusieurs cellules, la valeur indiquée sera répétée dans chaque cellule.

Le paramètre `numberValue` indique une valeur numérique à assigner à `rangeObj`.

Le paramètre optionnel `formatPattern` définit un `pattern` pour le paramètre `numberValue`.

Exemple

```
//Définir la valeur de la cellule à 2
VP SET NUM VALUE(VP Cell("ViewProArea";3;2);2)

//Définir la valeur de la cellule et la formater en dollars
VP SET NUM VALUE(VP Cell("ViewProArea";3;2);12.356;"_($* #,##0.00_)")
```

Voir aussi

[Cell format](#)

[VP SET VALUE](#)

VP SET PRINT INFO

`VP SET PRINT INFO (vpAreaName : Text ; printInfo : Object { ; sheet : Integer })`

Paramètres	Type		Description
vpAreaName	Text	->	Nom de la zone 4D View Pro
printInfo	Object	->	Objet contenant les attributs d'impression
sheet	Integer	->	Indice de la feuille (feuille courante si omis)

Description

La commande `VP SET PRINT INFO` définit les attributs à utiliser lors de l'impression de `vpAreaName`.

Passez le nom de la zone 4D View Pro que vous souhaitez imprimer dans `vpAreaName`. Si vous passez un nom inexistant, une erreur est renvoyée.

Vous pouvez passer un objet contenant les définitions de plusieurs attributs d'impression dans le paramètre `printInfo`. Pour consulter la liste complète des attributs disponibles, veuillez vous reporter à [Attributs d'impression](#).

Dans le paramètre optionnel `sheet`, vous pouvez définir une feuille (sheet) spécifique à imprimer (la numérotation démarre à zéro). Si le paramètre est omis, la feuille courante est utilisée par défaut. Vous pouvez sélectionner explicitement la feuille courante à l'aide de la constante suivante :

- `vk current sheet`

Exemple

Le code suivant imprimera une nouvelle zone 4D View Pro dans un document PDF :

```

var $printInfo : Object

//déclarer l'objet d'attributs d'impression
$printInfo:=New object

//definir les attributs d'impression
$printInfo.headerCenter:="&S.H.I.E.L.D. &A Sales Per Region"
$printInfo.firstPageNumber:=1
$printInfo.footerRight:="page &P of &N"
$printInfo.orientation:=vk print page orientation landscape
$printInfo.centering:=vk print centering horizontal
$printInfo.columnStart:=0
$printInfo.columnEnd:=8
$printInfo.rowStart:=0
$printInfo.rowEnd:=24

$printInfo.showGridLine:=True

//Ajouter un logo
$printInfo.headerLeftImage:=logo.png
$printInfo.headerLeft:="&G"

$printInfo.showRowHeader:=vk print visibility hide
$printInfo.showColumnHeader:=vk print visibility hide
$printInfo.fitPagesWide:=1
$printInfo.fitPagesTall:=1

//imprimer un document PDF
VP SET PRINT INFO ("ViewProArea";$printInfo)

//exporter le PDF
VP EXPORT DOCUMENT("ViewProArea";"Sales2018.pdf";New object("formula");Formula(ALERT("PDF ready!")))

```

Le PDF :



S.H.I.E.L.D. 2018 Sales Per Region

OrderDate	Region	Rep	Item	Units	UnitCost	Total
06-Jan-18	East	Stark	Pencil	95	\$1.99	\$189.05
23-Jan-18	Central	Rogers	Binder	50	\$19.99	\$999.50
09-Feb-18	Central	Banner	Pencil	36	\$4.99	\$179.64
26-Feb-18	Central	Romanoff	Pen	27	\$19.99	\$539.73
15-Mar-18	West	Barton	Pencil	56	\$2.99	\$167.44
01-Apr-18	East	Coulson	Binder	60	\$4.99	\$299.40
18-Apr-18	Central	Fury	Pencil	75	\$1.99	\$149.25
05-May-18	Central	Potts	Pencil	90	\$4.99	\$449.10
22-May-18	West	Jarvis	Pencil	32	\$1.99	\$63.68
08-Jun-18	East	Loki	Binder	60	\$8.99	\$539.40
25-Jun-18	Central	Odin	Pencil	90	\$4.99	\$449.10
06-Jul-18	East	Stark	Pencil	95	\$1.99	\$189.05
23-Jul-18	Central	Rogers	Binder	50	\$19.99	\$999.50
09-Aug-18	Central	Banner	Pencil	36	\$4.99	\$179.64
26-Aug-18	Central	Romanoff	Pen	27	\$19.99	\$539.73
15-Sep-18	West	Barton	Pencil	56	\$2.99	\$167.44
01-Oct-18	East	Coulson	Binder	60	\$4.99	\$299.40
18-Oct-18	Central	Fury	Pencil	75	\$1.99	\$149.25
05-Nov-18	Central	Potts	Pencil	90	\$4.99	\$449.10
22-Nov-18	West	Jarvis	Pencil	32	\$1.99	\$63.68
08-Dec-18	East	Loki	Binder	60	\$8.99	\$539.40
15-Dec-18	Central	Odin	Pencil	90	\$4.99	\$449.10
				1,342	\$155.78	\$8,050.58

page 1 of 1

Voir aussi

[4D View Pro print attributes](#)[VP Convert to picture](#)[VP Get print info](#)[VP PRINT](#)

VP SET ROW ATTRIBUTES

VP SET ROW ATTRIBUTES (*rangeObj* : Object ; *propertyObj* : Object)

Paramètres	Type		Description
<i>rangeObj</i>	Object	->	Plage de lignes
<i>propertyObj</i>	Object	->	Objet contenant des propriétés de lignes

Description

La commande `VP SET ROW ATTRIBUTES` applique les attributs définis dans le paramètre *propertyObj* pour les lignes de *rangeObj*.

Dans *rangeObj*, passez un objet contenant une plage. Si la plage contient des colonnes et des lignes, les attributs s'appliquent uniquement aux lignes.

Le paramètre *propertyObj* vous permet de renseigner les attributs à appliquer aux lignes de *rangeObj*. Ces attributs sont :

Propriété	Type	Description
height	number	Hauteur de la ligne exprimée en pixels
pageBreak	boolean	Vrai pour insérer un saut de page avant la première ligne de la plage, sinon faux
visible	boolean	Vrai si la ligne est visible, sinon faux
resizable	boolean	Vrai si la ligne peut être redimensionnée, sinon faux
header	Texte	Texte de l'en-tête de la ligne

Exemple

Pour modifier la taille de la deuxième ligne et définir l'en-tête, saisissez le code suivant :

```
var $row; $properties : Object

$row:=VP Row("ViewProArea";1)
$properties:=New object("height";75;"header";"June")

VP SET ROW ATTRIBUTES($row;$properties)
```

	A	B	C	D	E	F	G	H	I
1	The	quick	brown	fox	jumped	over	the	lazy	dog
June									

Voir aussi

[VP Get row attributes](#)
[VP get column attributes](#)
[VP SET ROW ATTRIBUTES](#)

VP SET ROW COUNT

VP SET ROW COUNT (*vpAreaName* : Text ; *rowCount* : Integer { ; *sheet* : Integer })

Paramètres	Type		Description
<i>vpAreaName</i>	Text	->	Nom d'objet formulaire zone 4D View Pro
<i>rowCount</i>	Integer	->	Nombre de lignes
<i>sheet</i>	Integer	->	Indice de la feuille (feuille courante si omis)

Description

La commande `VP SET ROW COUNT` définit le nombre total de lignes dans *vpAreaName*.

Dans *vpAreaName*, passez le nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est renvoyée.

Passez le nombre total de lignes dans le paramètre *rowCount*. *rowCount* doit être supérieur à 0.

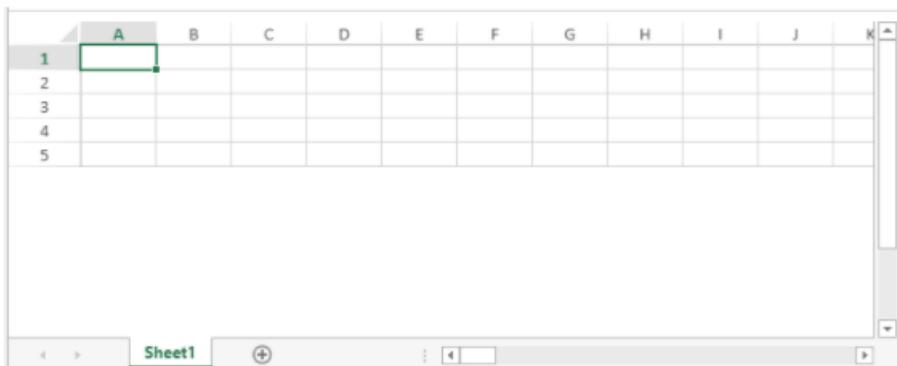
Dans le paramètre optionnel *sheet*, vous pouvez désigner une feuille de calcul spécifique dans laquelle s'appliquera *rowCount* (la numérotation démarre à zéro). Si le paramètre est omis, la feuille courante est utilisée par défaut. Vous pouvez sélectionner explicitement la feuille courante à l'aide de la constante suivante :

- `vk current sheet`

Exemple

Le code suivant définit cinq lignes dans la zone 4D View Pro :

```
VP SET ROW COUNT("ViewProArea";5)
```



Voir aussi

[VP Get column count](#)

[VP get row-count](#)

[VP SET COLUMN COUNT](#)

VP SET SELECTION

VP SET SELECTION (*rangeObj* : Object)

Paramètres	Type		Description
<i>rangeObj</i>	Object	->	Objet plage de toutes les cellules

Description

La commande `VP SET SELECTION` définit comme sélection les cellules spécifiées, et définit comme active la première cellule.

Dans *rangeObj*, passez un objet plage de cellule(s) à désigner comme sélection courante.

Exemple

```
$currentSelection:=VP Combine ranges(VP Cells("myVPArea";3;2;1;6);VP Cells("myVPArea";5;7;1;7))  
VP SET SELECTION($currentSelection)
```

	A	B	C	D	E	F	G
1							
2		Month	Area	Mr. Smith	Ms. Johnson	Ms. Williams	Mr. Jones
3		January	South	1898	1857	1651	1448
4			East	4859	4857	2548	4876
5			North	2458	1524	6150	4987
6			West	5787	1580	3975	4878
7		Total		15002	9818	14324	16189
8		February	South	6668	4374	17495	9999
9			East	5955	1677	7944	9400
10			North	1000	6722	2195	2777
11			West	6896	8355	7195	2058
12		Total		20519	21128	34829	24234
13		March	South	2577	2000	6185	2704
14			East	4859	4857	2548	4876
15			North	2458	1524	6150	4987
16			West	5787	1580	3975	4878
17		Total		15621	9961	18852	17115

Voir aussi

[VP Get active cell](#)

[VP Get selection](#)

[VP RESET SELECTION](#)

[VP SET ACTIVE CELL](#)

[VP ADD SELECTION](#)

[VP SHOW CELL](#)

VP SET SHEET COUNT

VP SET SHEET COUNT (*vpAreaName* : Text ; *number* : Integer)

Paramètres	Type		Description
<i>vpAreaName</i>	Text	->	Nom d'objet formulaire zone 4D View Pro
<i>number</i>	Integer	->	Nombre de feuilles

Description

The `VP SET SHEET COUNT` command sets the number of sheets in *vpAreaName*.

In *number*, pass a number corresponding to how many sheets the document will contain after the command is executed.

Warning: The command will delete sheets if the previous amount of sheets in your document is superior to the number passed. For example, if there are 5 sheets in your document and you set the sheet count to 3, the command will delete sheets number 4 and 5.

Exemple

The document currently has one sheet:



To set the number of sheets to 3:

```
VP SET SHEET COUNT("ViewProArea";3)
```



Voir aussi

[VP Get sheet count](#)

VP SET SHEET NAME

VP SET SHEET NAME (*vpAreaName* : Text ; *name* : Text {; *index*: Integer})

Paramètres	Type		Description
<i>vpAreaName</i>	Text	->	Nom d'objet formulaire zone 4D View Pro
<i>name</i>	Text	->	New name for the sheet
<i>index</i>	Integer	->	Index of the sheet to be renamed

Description

The `VP SET SHEET NAME` command renames a sheet in the document loaded in *vpAreaName*.

Dans *vpAreaName*, passez le nom de la zone 4D View Pro.

In *name*, pass a new name for the sheet.

In *index*, pass the index of the sheet to rename.

La numérotation démarre à 0.

If no *index* is passed, the command renames the current sheet.

Le nouveau nom ne peut pas contenir les caractères suivants : `* , :, [,] , ?, \, /`

The command does nothing if:

- the new name contains forbidden characters
- the new name's value is blank
- the new name already exists
- the passed *index* does not exist

Exemple

Set the third sheet's name to "Total first quarter":

```
VP SET SHEET NAME("ViewProArea";"Total first quarter";2)
```



VP SET SHEET OPTIONS

VP SET SHEET OPTIONS (*vpAreaName* : Text; *sheetOptions* : Object { ; *sheet* : Integer})

Paramètres	Type		Description
vpAreaName	Object	->	Nom de la zone 4D View Pro
sheetOptions	Object	->	Option(s) de la feuille à définir
sheet	Object	->	Indice de la feuille (feuille courante si omis)

Description

La commande `VP SET SHEET OPTIONS` permet de définir plusieurs options appliquées à une feuille de la zone `vpAreaName`.

Passez le nom de la zone 4D View Pro dans `vpAreaName`. Si vous passez un nom inexistant, une erreur est retournée.

Passez un objet contenant les définitions des options à définir dans le paramètre `sheetOptions`. Pour visualiser la liste complète des options disponibles, veuillez vous reporter aux [Sheet Options](#) de 4D View Pro.

Dans le paramètre optionnel `sheet`, vous pouvez désigner une feuille spécifique dans laquelle sera définie la plage (la numérotation commence à zéro). Si le paramètre est omis, la feuille courante est utilisée par défaut. Vous pouvez sélectionner explicitement la feuille courante à l'aide de la constante suivante :

- `vk current sheet`

Exemple 1

Vous souhaitez protéger toutes les cellules de la plage C5:D10 :

```
// Activer la protection sur la feuille courante
var $options : Object

$options:=New object
$options.isProtected:=True
VP SET SHEET OPTIONS("ViewProArea";$options)

// marquer les cellules C5:D10 comme 'déverrouillées'
VP SET CELL STYLE(VP Cells("ViewProArea";2;4;2;6);New object("locked";False))
```

Exemple 2

Vous souhaitez protéger votre document pendant que vos utilisateurs redimensionnent les lignes et colonnes :

```
var $options : Object

$options:=New object
// Activer la protection
$options.isProtected:=True
$options.protectionOptions:=New object
// Permettre à l'utilisateur de redimensionner les lignes
$options.protectionOptions.allowResizeRows=True;
// Permettre à l'utilisateur de redimensionner les colonnes
$options.protectionOptions.allowResizeColumns=True;

// Appliquer la protection à la feuille courante
VP SET SHEET OPTIONS("ViewProArea";$options)
```

Exemple 3

Vous souhaitez personnaliser la couleur des onglets, des lignes figées, du quadrillage, du fond de la sélection et de la bordure de la sélection :

```

var $options : Object

$options:=New object
// Personnaliser la couleur de l'onglet de la feuille 1
$options.sheetTabColor:="Black"
$options.gridline:=New object("color";"Purple")
$options.selectionBackColor:="rgb(255,128,0,0.4)"
$options.selectionBorderColor:="Yellow"
$options.frozenlineColor:="Gold"

VP SET SHEET OPTIONS("ViewProArea";$options;0)

// Personnaliser la couleur de l'onglet de la feuille 2
$options.sheetTabColor:="red"

VP SET SHEET OPTIONS("ViewProArea";$options;1)

// Personnaliser la couleur de l'onglet de la feuille 3
$options.sheetTabColor:="blue"

VP SET SHEET OPTIONS("ViewProArea";$options;2)

```

Résultat :

	A	B	E	F	G	H	I	J	
1									
8									
9									
10									
11									
12									
--									

Sheet1 Sheet2 Sheet3 + : ▲ ▼ ▶ ▷

Exemple 4

Vous souhaitez masquer le quadrillage et les en-têtes des lignes et colonnes.

```

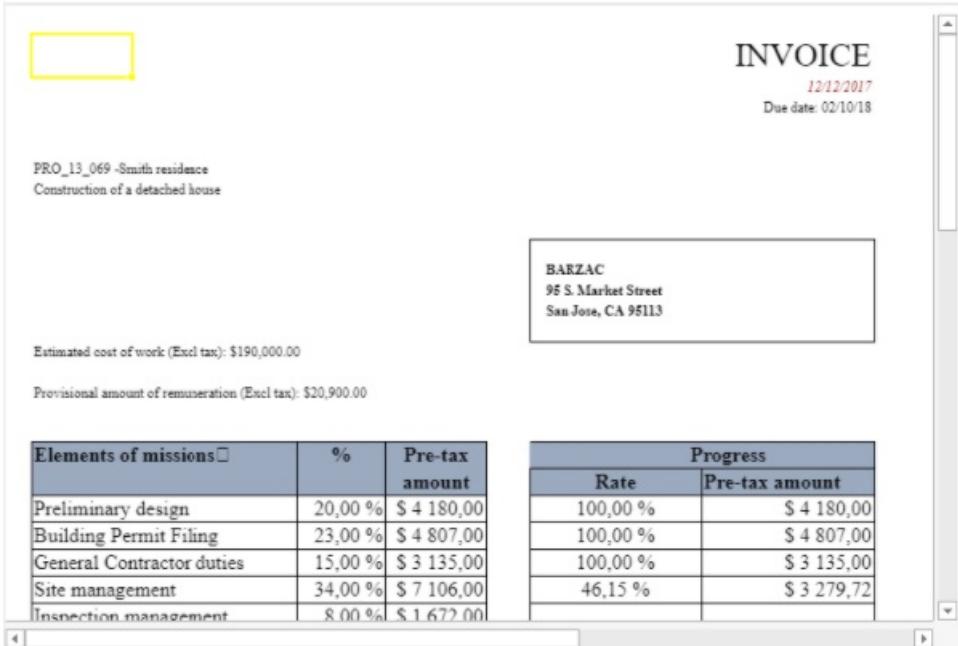
var $options : Object

$options:=New object
$options.gridline:=New object()
$options.gridline.showVerticalGridline:=False
$options.gridline.showHorizontalGridline:=False
$options.rowHeaderVisible:=False
$options.colHeaderVisible:=False

VP SET SHEET OPTIONS("ViewProArea";$options)

```

Résultat :



Voir aussi

[4D View Pro sheet options](#)

[VP Get sheet options](#)

VP SET SHOW PRINT LINES

VP SET SHOW PRINT LINES (vpAreaName : Text {; visible : Boolean}{; index : Integer})

Paramètres	Type		Description
vpAreaName	Text	->	Nom d'objet formulaire zone 4D View Pro
visible	Booléen	->	Print lines displayed if True (default), hidden if False
index	Integer	->	Indice de la feuille

Description

The **VP SET SHOW PRINT LINES** command sets whether to display print preview lines in a spreadsheet..

Dans *vpAreaName*, passez le nom de la zone 4D View Pro.

In *visible*, pass **True** to display the print lines, and **False** to hide them. **True** is passed by default.

In *index*, pass the index of the target sheet. If no index is specified, the command applies to the current sheet.

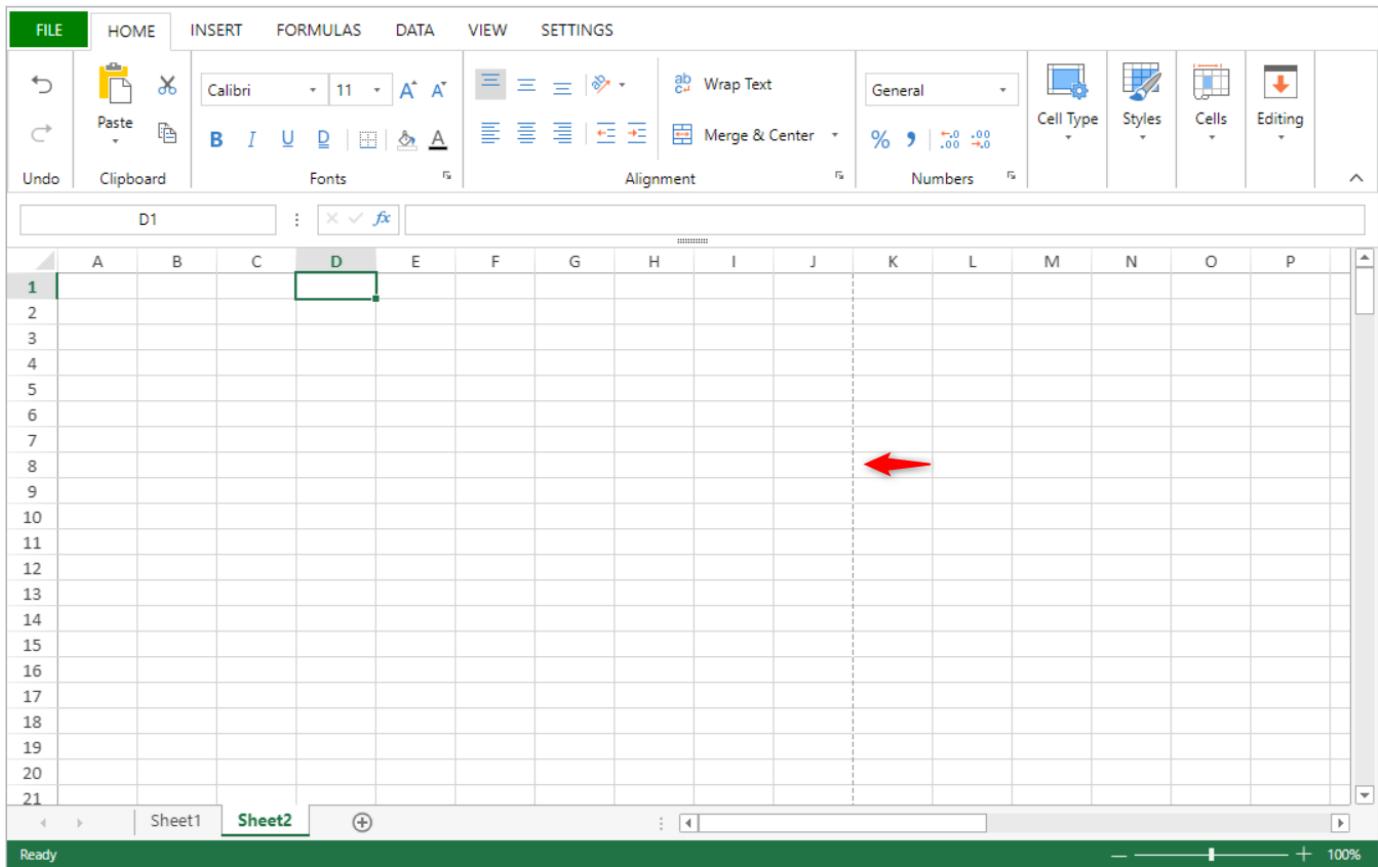
La numérotation démarre à 0.

The position of a spreadsheet's print lines varies according to that spreadsheet's page breaks.

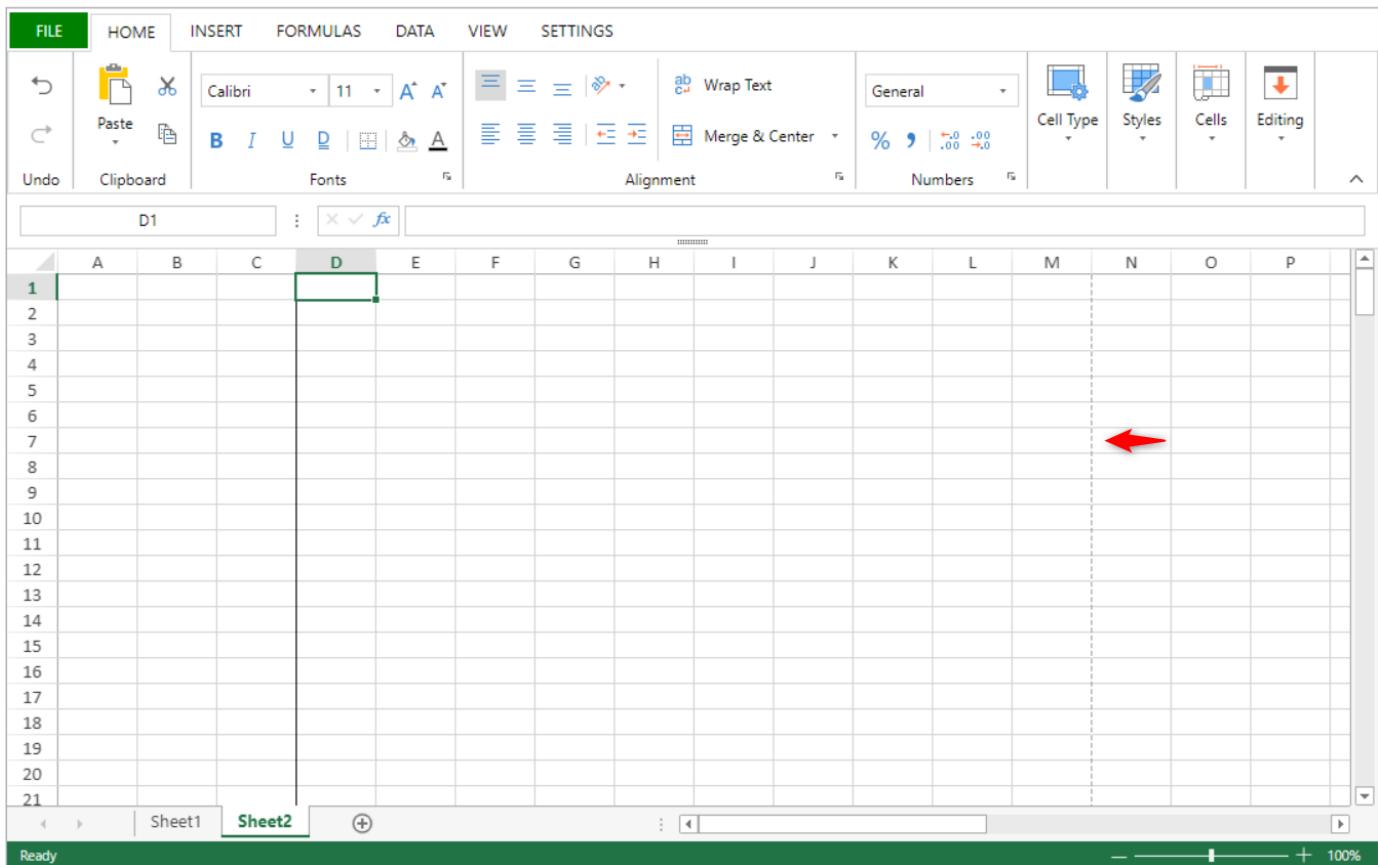
Exemple

The following code displays print lines in a document's second sheet:

```
VP SET SHOW PRINT LINES("ViewProArea";True;1)
```



With a page break:



Voir aussi

[4D Get show print lines](#)

VP SET TEXT VALUE

VP SET TEXT VALUE (*rangeObj* : Object ; *textValue* : Text { ; *formatPattern* : Text })

Paramètres	Type		Description
rangeObj	Object	->	Objet plage
textValue	Text	->	Valeur texte à fixer
formatPattern	Text	->	Format de la valeur

Description

La commande `VP SET TEXT VALUE` assigne une valeur texte spécifique à une plage de cellules désignée.

Dans *rangeObj*, passez la plage de cellule(s) (créeé par exemple avec `VP Cell` ou `VP Column`) dont vous souhaitez indiquer la valeur. Si *rangeObj* comprend plusieurs cellules, la valeur indiquée sera répétée dans chaque cellule.

Le paramètre *textValue* indique une valeur texte à assigner à *rangeObj*.

Le paramètre optionnel *formatPattern* définit un [pattern](#) pour le paramètre *textValue*.

Exemple

```
VP SET TEXT VALUE(VP Cell("ViewProArea";3;2);"Test 4D View Pro")
```

Voir aussi

[Cell Format](#)

[VP SET VALUE](#)

VP SET TIME VALUE

`VP SET TIME VALUE (rangeObj : Object ; timeValue : Text { ; formatPattern : Text })`

Paramètres	Type		Description
rangeObj	Object	->	Objet plage
timeValue	Text	->	Valeur heure à fixer
formatPattern	Text	->	Format de la valeur

Description

The `VP SET TIME VALUE` command assigns a specified time value to a designated cell range.

Dans *rangeObj*, passez la plage de cellule(s) (créeé par exemple avec `VP Cell` ou `VP Column`) dont vous souhaitez indiquer la valeur. Si *rangeObj* comprend plusieurs cellules, la valeur indiquée sera répétée dans chaque cellule.

The *timeValue* parameter specifies a time expressed in seconds to be assigned to the *rangeObj*.

The optional *formatPattern* defines a [pattern](#) for the *timeValue* parameter.

Exemple

```
//Set the value to the current time
VP SET TIME VALUE(VP Cell("ViewProArea";5;2);Current time)

//Set the value to a specific time with a designated format
VP SET TIME VALUE(VP Cell("ViewProArea";5;2);?12:15:06?vk pattern long time)
```

Voir aussi

[Cell Format](#)

[VP SET DATE TIME VALUE](#)

[VP SET VALUE](#)

VP SET VALUE

VP SET VALUE (*rangeObj* : Object ; *valueObj* : Object)

Paramètres	Type		Description
<i>rangeObj</i>	Object	->	Objet plage
<i>valueObj</i>	Object	->	Valeurs de la cellule et options de formatage

Description

La commande `VP SET VALUE` assigne une valeur spécifique à une plage de cellules désignée.

Cette commande vous permet d'utiliser du code générique pour fixer et formater les types de valeurs dans *rangeObj*, tandis que d'autres commandes, telles que [VP SET TEXT VALUE](#) et [VP SET NUM VALUE](#), limitent les valeurs à des types spécifiques.

Dans *rangeObj*, passez la plage de cellule(s) (créeé par exemple avec [VP Cell](#) ou [VP Column](#)) dont vous souhaitez indiquer la valeur. Si *rangeObj* comprend plusieurs cellules, la valeur indiquée sera répétée dans chaque cellule.

Le paramètre *valueObj* est un objet qui définit la valeur et le [format](#) à assigner à *rangeObj*. Il peut contenir les propriétés suivantes :

Propriété	Type	Description
<i>value</i>	Integer, Real, Boolean, Text, Date, Null	Valeur à assigner à <i>rangeObj</i> (exceptée heure). Passez null pour effacer le contenu de la cellule.
<i>time</i>	Réel	Valeur heure (en secondes) à assigner à <i>rangeObj</i>
<i>format</i>	Text	Pattern for value/time property. Pour plus d'informations sur les modèles et les caractères de formatage, veuillez consulter la section Format de cellule .

Exemple

```

//Fixer la valeur de la cellule à Faux
VP SET VALUE(VP Cell("ViewProArea";3;2);New object("value";False))

//Fixer la valeur de la cellule à 2
VP SET VALUE(VP Cell("ViewProArea";3;2);New object("value";2))

//Fixer la valeur de la cellule à $125,571.35
VP SET VALUE(VP Cell("ViewProArea";3;2);New object("value";125571.35;"format";"$_(* #,##0.00_)"))

//Fixer la valeur de la cellule à Hello World!
VP SET VALUE(VP Cell("ViewProArea";3;2);New object("value";"Hello World!"))

//Fixer la valeur de la cellule à date du jour
VP SET VALUE(VP Cell("ViewProArea";4;2);New object("value";Current date))

//Fixer la valeur de la cellule à heure courante
VP SET VALUE(VP Cell("ViewProArea";5;2);New object("time";Current time))

//Fixer la valeur de la cellule à une date et une heure spécifiques
VP SET VALUE(VP Cell("ViewProArea";3;9);New object("value";!2024-12-18!); "time";?14:30:10?;"format";vk p)

//Effacer le contenu de la cellule
VP SET VALUE(VP Cell("ViewProArea";3;9);New object("value";Null))!

VP SET VALUE(VP Cell("ViewProArea";3;2);New object("value";"Hello World!"))

VP SET VALUE(VP Cell("ViewProArea";3;2);New object("value";"Hello World!"))

VP SET VALUE(VP Cell("ViewProArea";3;2);New object("value";"Hello World!"))

//Set the cell value as current date
VP SET VALUE(VP Cell("ViewProArea";4;2);New object("value";Current date))

//Set the cell value as current hour
VP SET VALUE(VP Cell("ViewProArea";5;2);New object("time";Current hour))

//Set the cell value as specific date and time
VP SET VALUE(VP Cell("ViewProArea";3;9);New object("value";!2024-12-18!); "time";?14:30:10?;"format";vk p)

//Erase cell content
VP SET VALUE(VP Cell("ViewProArea";3;9);New object("value";Null))

```

Voir aussi

[Cell Format](#)
[VP Get values](#)
[VP SET VALUE](#)
[VP SET BOOLEAN VALUE](#)
[VP SET DATE TIME VALUE](#)
[VP SET FIELD](#)
[VP SET FORMULA](#)
[VP SET NUM VALUE](#)
[VP SET TEXT VALUE](#)
[VP SET TIME VALUE](#)

VP SET VALUES

`VP SET VALUES (rangeObj : Object ; valuesCol : Collection)`

Paramètres	Type		Description
rangeObj	Object	->	Objet plage
valuesCol	Collection	->	Collection de valeurs

Description

La commande `VP SET VALUES` affecte une collection de valeurs qui commence à partir de la plage de cellule spécifiée.

Dans `rangeObj`, passez une plage de cellule (créeée via `VP Cell`) dont vous souhaitez indiquer la valeur. La cellule définie dans `rangeObj` est utilisée pour déterminer le point de départ.

- Si `rangeObj` comprend plusieurs plages, seule la première cellule de la première plage est utilisée.
- Si `rangeObj` comprend plusieurs plages, seule la première cellule de la première plage est utilisée.

Le paramètre `valuesCol` est bidimensionnel :

- La collection de premier niveau contient des sous-collections de valeurs. Chaque sous-collection définit une ligne. Chaque sous-collection définit une ligne.
- Chaque sous-collection définit les valeurs des cellules de la ligne. Les valeurs peuvent être de type entier long, réel, booléen, texte, date, null ou objet. Si la valeur est un objet, elle peut avoir les propriétés suivantes :

Propriété	Type	Description
value	Integer, Real, Boolean, Text, Date, Null	Valeur de la cellule (excepté - time)
time	Réel	Valeur heure (en secondes)

Exemple

```
$param:=New collection
$param.push(Creer collection(1;2;3;False)) //première ligne, 4 valeurs
$param.push(Creer collection) //deuxième ligne, inchangé
$param.push(Creer collection(4;5;Null;"hello";"world")) // troisième ligne, 5 valeurs
$param.push(Creer collection(6;7;8;9)) // quatrième ligne, 4 valeurs
$param.push(Creer collection(Null;Creer objet("value";Current date;"time";42))) //cinquième ligne, 1 va
VP SET VALUES(VP Cell("ViewProArea";2;1);$param)
```

	A	B	C	D	E	F	G
1							
2			1	2	3	FALSE	
3							
4			4	5		hello	world
5			6	7	8		9
6			29/05/2019 0:00:42				
7							

Voir aussi

[VP Get formulas](#)
[VP Get value](#)
[VP Get Values](#)
[VP SET FORMULAS](#)
[VP SET VALUE](#)

VP SET WORKBOOK OPTIONS

VP SET WORKBOOK OPTIONS (*vpAreaName* : Text ; *optionObj* : Object)

Paramètres	Type		Description
<i>vpAreaName</i>	Text	->	Nom d'objet formulaire zone 4D View Pro
<i>optionObj</i>	Object	->	Object containing the workbook options to be set

Description

VP SET WORKBOOK OPTIONS sets the workbook options in *vpAreaName*.

Dans *vpAreaName*, passez le nom de la zone 4D View Pro.

In *optionObj*, pass the workbook options to apply to *vpAreaName*.

If *optionObj* is empty, the command does nothing.

Modified workbook options are saved with the document.

The following table lists the available workbook options:

Propriété	Type	Description									
allowUserDragMerge	boolean	The drag merge operation is allowed (select cells and drag the selection to merge cells)									
allowAutoCreateHyperlink	boolean	Enables automatic creation of hyperlinks in the spreadsheet.									
allowContextMenu	boolean	The built-in context menu can be opened.									
allowCopyPasteExcelStyle	boolean	Styles from a spreadsheet can be copied and pasted to Excel, and vice-versa.									
allowDynamicArray	boolean	Enables dynamic arrays in worksheets									
allowExtendPasteRange	boolean	Extends the pasted range if the pasted range is not enough for the pasted data									
allowSheetReorder	boolean	Sheet reordering is allowed									
allowUndo	boolean	Undoing edits is allowed.									
allowUserDeselect	boolean	Deselecting specific cells from a selection is allowed.									
allowUserDragDrop	boolean	Drag and drop of range data is allowed									
allowUserDragFill	boolean	Drag fill is allowed									
allowUserEditFormula	boolean	Formulas can be entered in cells									
allowUserResize	boolean	Columns and rows can be resized									
allowUserZoom	boolean	Zooming (ctrl + mouse wheel) is allowed									
autoFitType	number	Content is formatted to fit in cells, or cells and headers. Valeurs disponibles : <table border="1"> <thead> <tr> <th>Constante</th> <th>Valeur</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>vk auto fit type cell</td> <td>0</td> <td>The content autofits cells</td> </tr> <tr> <td>vk auto fit type cell with header</td> <td>1</td> <td>The content autofits cells and headers</td> </tr> </tbody> </table>	Constante	Valeur	Description	vk auto fit type cell	0	The content autofits cells	vk auto fit type cell with header	1	The content autofits cells and headers
Constante	Valeur	Description									
vk auto fit type cell	0	The content autofits cells									
vk auto fit type cell with header	1	The content autofits cells and headers									
backColor	string	A color string used to represent the background color of the area, such as "red", "#FFFF00", "rgb(255,0,0)", "Accent 5". The initial backgroundcolor is hidden when a <i>backgroundImage</i> is set									

			backgroundImage	Type string / picture / file	Description Background image for the area.															
			backgroundImageLayout	number	How the background image is displayed. Valeurs disponibles :															
					<table border="1"> <thead> <tr> <th>Constante</th> <th>Valeur</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>vk image layout center</td> <td>1</td> <td>In the center of the area.</td> </tr> <tr> <td>vk image layout none</td> <td>3</td> <td>In the upper left corner of the area with its original size.</td> </tr> <tr> <td>vk image layout stretch</td> <td>0</td> <td>Fills the area.</td> </tr> <tr> <td>vk image layout zoom</td> <td>2</td> <td>Displayed with its original aspect ratio.</td> </tr> </tbody> </table>	Constante	Valeur	Description	vk image layout center	1	In the center of the area.	vk image layout none	3	In the upper left corner of the area with its original size.	vk image layout stretch	0	Fills the area.	vk image layout zoom	2	Displayed with its original aspect ratio.
Constante	Valeur	Description																		
vk image layout center	1	In the center of the area.																		
vk image layout none	3	In the upper left corner of the area with its original size.																		
vk image layout stretch	0	Fills the area.																		
vk image layout zoom	2	Displayed with its original aspect ratio.																		
			calcOnDemand	boolean	Formulas are calculated only when they are demanded.															
			columnResizeMode	number	Resize mode for columns. Valeurs disponibles :															
					<table border="1"> <thead> <tr> <th>Constante</th> <th>Valeur</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>vk resize mode normal</td> <td>0</td> <td>Use normal resize mode (i.e remaining columns are affected)</td> </tr> <tr> <td>vk resize mode split</td> <td>1</td> <td>Use split mode (i.e remaining columns are not affected)</td> </tr> </tbody> </table>	Constante	Valeur	Description	vk resize mode normal	0	Use normal resize mode (i.e remaining columns are affected)	vk resize mode split	1	Use split mode (i.e remaining columns are not affected)						
Constante	Valeur	Description																		
vk resize mode normal	0	Use normal resize mode (i.e remaining columns are affected)																		
vk resize mode split	1	Use split mode (i.e remaining columns are not affected)																		
			copyPasteHeaderOptions	number	Headers to include when data is copied to or pasted. Valeurs disponibles :															
					<table border="1"> <thead> <tr> <th>Constante</th> <th>Valeur</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>vk copy paste header options all headers</td> <td>3</td> <td>Includes selected headers when data is copied; overwrites selected headers when data is pasted.</td> </tr> <tr> <td>vk copy paste header options column headers</td> <td>2</td> <td>Includes selected column headers when data is copied; overwrites selected column headers when data is pasted.</td> </tr> <tr> <td>vk copy paste header options no headers</td> <td>0</td> <td>Column and row headers are not included when data is copied; does not overwrite selected column or row headers when data is pasted.</td> </tr> <tr> <td>vk copy paste header options row headers</td> <td>1</td> <td>Includes selected row headers when data is copied; overwrites selected row headers when data is pasted.</td> </tr> </tbody> </table>	Constante	Valeur	Description	vk copy paste header options all headers	3	Includes selected headers when data is copied; overwrites selected headers when data is pasted.	vk copy paste header options column headers	2	Includes selected column headers when data is copied; overwrites selected column headers when data is pasted.	vk copy paste header options no headers	0	Column and row headers are not included when data is copied; does not overwrite selected column or row headers when data is pasted.	vk copy paste header options row headers	1	Includes selected row headers when data is copied; overwrites selected row headers when data is pasted.
Constante	Valeur	Description																		
vk copy paste header options all headers	3	Includes selected headers when data is copied; overwrites selected headers when data is pasted.																		
vk copy paste header options column headers	2	Includes selected column headers when data is copied; overwrites selected column headers when data is pasted.																		
vk copy paste header options no headers	0	Column and row headers are not included when data is copied; does not overwrite selected column or row headers when data is pasted.																		
vk copy paste header options row headers	1	Includes selected row headers when data is copied; overwrites selected row headers when data is pasted.																		
			customList	collection	The list for users to customize drag fill, prioritize matching this list in each fill. Each collection item is a collection of															

Propriété	Type	Strings. See on GrapeCity's website .																					
cutCopyIndicatorBorderColor	string	Border color for the indicator displayed when the user cuts or copies the selection.																					
cutCopyIndicatorVisible	boolean	Display an indicator when copying or cutting the selected item.																					
defaultDragFillType	number	<p>The default drag fill type. Available values :</p> <table border="1"> <thead> <tr> <th>Constante</th> <th>Valeur</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>vk auto fill type auto</td> <td>5</td> <td>Automatically fills cells.</td> </tr> <tr> <td>vk auto fill type clear values</td> <td>4</td> <td>Clears cell values.</td> </tr> <tr> <td>vk auto fill type copycells</td> <td>0</td> <td>Fills cells with all data objects, including values, formatting, and formulas.</td> </tr> <tr> <td>vk auto fill type fill formatting only</td> <td>2</td> <td>Fills cells only with formatting.</td> </tr> <tr> <td>vk auto fill type fill series</td> <td>1</td> <td>Fills cells with series.</td> </tr> <tr> <td>vk auto fill type fill without formatting</td> <td>3</td> <td>Fills cells with values and not formatting.</td> </tr> </tbody> </table>	Constante	Valeur	Description	vk auto fill type auto	5	Automatically fills cells.	vk auto fill type clear values	4	Clears cell values.	vk auto fill type copycells	0	Fills cells with all data objects, including values, formatting, and formulas.	vk auto fill type fill formatting only	2	Fills cells only with formatting.	vk auto fill type fill series	1	Fills cells with series.	vk auto fill type fill without formatting	3	Fills cells with values and not formatting.
Constante	Valeur	Description																					
vk auto fill type auto	5	Automatically fills cells.																					
vk auto fill type clear values	4	Clears cell values.																					
vk auto fill type copycells	0	Fills cells with all data objects, including values, formatting, and formulas.																					
vk auto fill type fill formatting only	2	Fills cells only with formatting.																					
vk auto fill type fill series	1	Fills cells with series.																					
vk auto fill type fill without formatting	3	Fills cells with values and not formatting.																					
enableAccessibility	boolean	Accessibility support is enabled in the spreadsheet.																					
enableFormulaTextbox	boolean	The formula text box is enabled.																					
grayAreaBackColor	string	A color string used to represent the background color of the gray area , such as "red", "#FFFF00", "rgb(255,0,0)", "Accent 5", and so on.																					
highlightInvalidData	boolean	Invalid data is highlighted.																					
iterativeCalculation	boolean	Enables iterative calculation. See on Grapecity's website .																					
iterativeCalculationMaximumChange	numeric	Maximum amount of change between two calculation values.																					
iterativeCalculationMaximumIterations	numeric	Number of times the formula should recalculate.																					
newTabVisible	boolean	Display a special tab to let users insert new sheets.																					
numbersFitMode	number	<p>Changes display mode when date/number data width is longer than column width. Valeurs disponibles :</p> <table border="1"> <thead> <tr> <th>Constante</th> <th>Valeur</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>vk numbers fit mode mask</td> <td>0</td> <td>Replace data content with "###" and shows tip</td> </tr> <tr> <td>vk numbers fit mode overflow</td> <td>1</td> <td>Display data content as a string. If next cell is empty, overflow the content.</td> </tr> </tbody> </table>	Constante	Valeur	Description	vk numbers fit mode mask	0	Replace data content with "###" and shows tip	vk numbers fit mode overflow	1	Display data content as a string. If next cell is empty, overflow the content.												
Constante	Valeur	Description																					
vk numbers fit mode mask	0	Replace data content with "###" and shows tip																					
vk numbers fit mode overflow	1	Display data content as a string. If next cell is empty, overflow the content.																					
pasteSkipInvisibleRange	boolean	<p>Paste or skip pasting data in invisible ranges:</p> <ul style="list-style-type: none"> • False (default): paste data • True: Skip pasting in invisible ranges <p>See Grapecity's docs for more information on invisible ranges.</p>																					

PreferenceStyle	number	Description : Style for cell and range references in cell formulas. Valeurs disponibles : <table border="1"> <thead> <tr> <th>Constante</th><th>Valeur</th><th>Description</th></tr> </thead> <tbody> <tr> <td>vk reference style A1</td><td>0</td><td>Use A1 style.</td></tr> <tr> <td>vk reference style R1C1</td><td>1</td><td>Use R1C1 style</td></tr> </tbody> </table>	Constante	Valeur	Description	vk reference style A1	0	Use A1 style.	vk reference style R1C1	1	Use R1C1 style			
Constante	Valeur	Description												
vk reference style A1	0	Use A1 style.												
vk reference style R1C1	1	Use R1C1 style												
resizeZeroIndicator	number	Drawing policy when the row or column is resized to zero. Valeurs disponibles : <table border="1"> <thead> <tr> <th>Constante</th><th>Valeur</th><th>Description</th></tr> </thead> <tbody> <tr> <td>vk resize zero indicator default</td><td>0</td><td>Uses the current drawing policy when the row or column is resized to zero.</td></tr> <tr> <td>vk resize zero indicator enhanced</td><td>1</td><td>Draws two short lines when the row or column is resized to zero.</td></tr> </tbody> </table>	Constante	Valeur	Description	vk resize zero indicator default	0	Uses the current drawing policy when the row or column is resized to zero.	vk resize zero indicator enhanced	1	Draws two short lines when the row or column is resized to zero.			
Constante	Valeur	Description												
vk resize zero indicator default	0	Uses the current drawing policy when the row or column is resized to zero.												
vk resize zero indicator enhanced	1	Draws two short lines when the row or column is resized to zero.												
rowResizeMode	number	The way rows are resized. Available values are the same as columnResizeMode												
scrollbarAppearance	number	Scrollbar appearance. Valeurs disponibles : <table border="1"> <thead> <tr> <th>Constante</th><th>Valeur</th><th>Description</th></tr> </thead> <tbody> <tr> <td>vk scrollbar appearance mobile</td><td>1</td><td>Mobile scrollbar appearance.</td></tr> <tr> <td>vk scrollbar appearance skin (default)</td><td>0</td><td>Excel-like classic scrollbar appearance.</td></tr> </tbody> </table>	Constante	Valeur	Description	vk scrollbar appearance mobile	1	Mobile scrollbar appearance.	vk scrollbar appearance skin (default)	0	Excel-like classic scrollbar appearance.			
Constante	Valeur	Description												
vk scrollbar appearance mobile	1	Mobile scrollbar appearance.												
vk scrollbar appearance skin (default)	0	Excel-like classic scrollbar appearance.												
scrollbarMaxAlign	boolean	The scroll bar aligns with the last row and column of the active sheet.												
scrollbarShowMax	boolean	The displayed scroll bars are based on the entire number of columns and rows in the sheet.												
scrollByPixel	boolean	Enable precision scrolling by pixel.												
scrollIgnoreHidden	boolean	The scroll bar ignores hidden rows or columns.												
scrollPixel	entier	Decides scrolling by that number of pixels at a time when scrollByPixel is true. The final scrolling pixels are the result of scrolling delta * scrollPixel . For example: scrolling delta is 3, scrollPixel is 5, the final scrolling pixels are 15.												
showDragDropTip	boolean	Display the drag-drop tip.												
showDragFillSmartTag	boolean	Display the drag fill dialog.												
showDragFillTip	boolean	Display the drag-fill tip.												
showHorizontalScrollbar	boolean	Display the horizontal scroll bar.												
showResizeTip	number	How to display the resize tip. Valeurs disponibles : <table border="1"> <thead> <tr> <th>Constante</th><th>Valeur</th><th>Description</th></tr> </thead> <tbody> <tr> <td>vk show resize tip both</td><td>3</td><td>Horizontal and vertical resize tips are displayed.</td></tr> <tr> <td>vk show resize tip column</td><td>1</td><td>Only the horizontal resize tip is displayed.</td></tr> <tr> <td>vk show resize</td><td>0</td><td>No resize tip is displayed.</td></tr> </tbody> </table>	Constante	Valeur	Description	vk show resize tip both	3	Horizontal and vertical resize tips are displayed.	vk show resize tip column	1	Only the horizontal resize tip is displayed.	vk show resize	0	No resize tip is displayed.
Constante	Valeur	Description												
vk show resize tip both	3	Horizontal and vertical resize tips are displayed.												
vk show resize tip column	1	Only the horizontal resize tip is displayed.												
vk show resize	0	No resize tip is displayed.												

Propriété	Type	Description			
		vk show resize tip row	2	Only the vertical resize tip is displayed.	
showScrollTip	number	How to display the scroll tip. Valeurs disponibles :			
		Constante	Valeur	Description	
		vk show scroll tip both	3	Horizontal and vertical scroll tips are displayed.	
		vk show scroll tip horizontal	1	Only the horizontal scroll tip is displayed.	
		vk show scroll tip none	No scroll tip is displayed.		
		vk show scroll tip vertical	2	Only the vertical scroll tip is displayed.	
showVerticalScrollbar	boolean	Display the vertical scroll bar.			
tabEditable	boolean	The sheet tab strip can be edited.			
tabNavigationVisible	boolean	Display the sheet tab navigation.			
tabStripPosition	number	Position of the tab strip. Valeurs disponibles :			
		Constante	Valeur	Description	
		vk tab strip position bottom	0	Tab strip position is relative to the bottom of the workbook.	
		vk tab strip position left	2	Tab strip position is relative to the left of the workbook.	
		vk tab strip position right	3	Tab strip position is relative to the right of the workbook.	
		vk tab strip position top	1	Tab strip position is relative to the top of the workbook.	
tabStripRatio	number	Percentage value (0.x) that specifies how much of the horizontal space will be allocated to the tab strip. The rest of the horizontal area (1 - 0.x) will be allocated to the horizontal scrollbar.			
tabStripVisible	boolean	Display the sheet tab strip.			
tabStripWidth	number	Width of the tab strip when position is left or right. Default and minimum is 80.			
useTouchLayout	boolean	Whether to use touch layout to present the Spread component.			

Exemple

To set the allowExtendPasteRange option in "ViewProArea":

```

var $workbookOptions : Object
$workbookOptions:= New Object
$workbookOptions.allowExtendPasteRange:=True
VP SET WORKBOOK OPTIONS("ViewProArea";$workbookOptions)

```

Voir aussi

[VP Get workbook options](#)

VP SHOW CELL

VP SHOW CELL (rangeObj : Object { ; vPos : Integer; hPos : Integer })

Paramètres	Type		Description
rangeObj	Object	->	Objet plage
vPos	Integer	->	Position verticale de la vue de la cellule ou de la ligne
hPos	Integer	->	Position horizontale de la vue de la cellule ou de la ligne

Description

La commande **VP SHOW CELL** repositionne verticalement et horizontalement la vue de *rangeObj*.

Dans *rangeObj*, passez, en tant qu'objet, une plage de cellules que vous souhaitez afficher. La vue de *rangeObj* sera positionnée verticalement ou horizontalement (i.e., là où *rangeObj* apparaît) en fonction des paramètres *vPos* et *hPos*. Le paramètre *vPos* définit la position verticale souhaitée pour afficher *rangeObj* et le paramètre *vPos* définit la position horizontale souhaitée pour afficher *rangeObj*.

Les sélecteurs suivants sont disponibles :

Sélecteur	Description	Disponible avec vPos	là où la ligne est appliquée) à l'aide de borderPosObj :
<code>vk position bottom</code>	Alignement vertical vers le bas de la cellule ou de la ligne.	X	
<code>vk position center</code>	Alignement vers le centre. L'alignement s'appliquera à la limite de la cellule, ligne ou colonne, en fonction de la position de la vue indiquée : <ul style="list-style-type: none">• Position verticale de la vue - cellule ou ligne• Position horizontale de la vue - cellule ou ligne	X	X
<code>vk position left</code>	Alignement horizontal vers la gauche de la cellule ou de la colonne		X
<code>vk position nearest</code>	Alignement vers la limite la plus proche (haut, bas, gauche, droite, centre). L'alignement s'appliquera à la limite de la cellule, ligne ou colonne, en fonction de la position de la vue indiquée : <ul style="list-style-type: none">• Position verticale de la vue (haut, centre, bas) - cellule ou ligne• Position horizontale de la vue (gauche, centre, droite) - cellule ou ligne	X	X
<code>vk position right</code>	Alignement horizontal vers la droite de la cellule ou de la colonne		X
<code>vk position top</code>	Alignement vertical vers le haut de la cellule ou de la ligne	X	

Cette commande n'est efficace que si le repositionnement de la vue est possible. Par exemple, si `rangeObj` est contenu dans la cellule A1 (la première colonne et la première ligne) de la feuille courante, le repositionnement de la vue n'apportera aucun changement, étant donné que les limites verticales et horizontales ont déjà été atteintes (i.e., il n'est pas possible de faire dérouler davantage vers le haut ou vers la gauche). De même si `rangeObj` est contenu dans la cellule C3 et que la vue est repositionnée au centre ou en bas à droite. La vue demeure inchangée.

Exemple

Vous souhaitez visualiser la cellule dans la colonne AY, ligne 51, au centre de la zone 4D View Pro.

```
$displayCell:=VP Cell("myVPArea";50;50)
// Déplacez la vue pour afficher la cellule
VP SHOW CELL($displayCell;vk position center;vk position center)
```

Résultat :

	AV	AW	AX	AY	AZ	BA	BB	BC
42								
43								
44								
45								
46								
47								
48								
49								
50								
51				Hello World				
52								
53								
54								
55								
56								
57								
58								
59								
60								
61								

Le même code ainsi que les sélecteurs verticaux et horizontaux ont été modifiés pour afficher la même cellule en haut à droite de la zone 4D View Pro :

```
$displayCell:=VP Cell("myVPArea";50;50)
// Déplacez la vue pour afficher la cellule
VP SHOW CELL($displayCell;vk position top;vk position right)
```

Résultat :

	AS	AT	AU	AV	AW	AX	AY	AZ
51							Hello World	
52								
53								
54								
55								
56								
57								
58								
59								
60								
61								
62								
63								
64								
65								
66								
67								
68								
69								
70								

Voir aussi

[VP ADD CELL](#)

[VP Get active cell](#)

VP Get selection
VP RESET SELECTION
VP SET ACTIVE CELL
VP SET SELECTION

VP SUSPEND COMPUTING

VP SUSPEND COMPUTING (*vpAreaName* : Text)

Paramètres	Type		Description
<i>vpAreaName</i>	Text	->	Nom d'objet formulaire zone 4D View Pro

Description

La commande **VP SUSPEND COMPUTING** stoppe le calcul de toutes les formules dans *vpAreaName*. Cette commande est utile lorsque vous souhaitez suspendre les calculs dans cette zone 4D View Pro, afin de modifier manuellement les formules sans générer d'erreurs avant la fin de vos modifications.

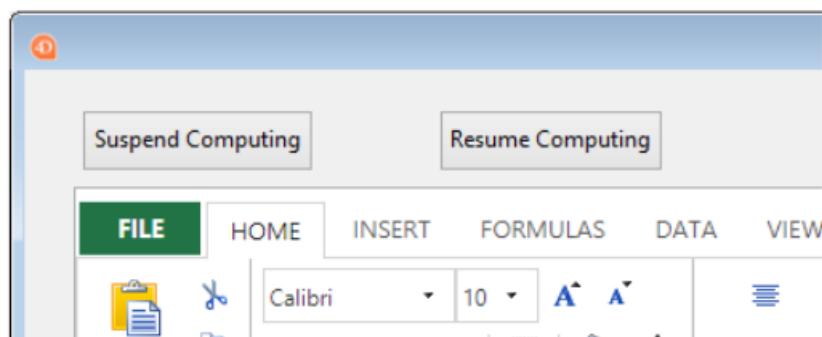
La commande met en pause les calculs dans 4D View Pro. Les formules déjà calculées restent inchangées, mais les formules ajoutées après **VP SUSPEND COMPUTING** et qui sont exécutées, ne sont pas calculées.

Dans *vpAreaName*, passez le nom de la zone 4D View Pro. Si vous passez un nom inexistant, une erreur est retournée.

Le service de calcul de 4D View Pro maintient un compteur d'actions de suspension/reprise. Ainsi, chaque exécution de **VP SUSPEND COMPUTING** doit être compensée par une exécution correspondante de la commande **VP RESUME COMPUTING**. Toute formule modifiée durant la phase de suspension des calculs sera recalculée lorsque la commande sera exécutée.

Exemple

Vous avez ajouté deux boutons au formulaire afin que l'utilisateur puisse suspendre/reprendre les calculs :



Le code du bouton Suspend Computing :

```
//mettre les calculs sur pause pendant que les utilisateurs saisissent les informations
If(FORM Event.code=On Clicked)

    VP SUSPEND COMPUTING("ViewProArea")

End if
```

```
If(FORM Event.code=On Clicked)

    VP RESUME COMPUTING("ViewProArea")

End if
```

Voir aussi

[VP RECOMUTE FORMULAS](#)

[VP RESUME COMPUTING](#)

Programmation avancée avec Javascript

Une zone 4D View Pro est un [objet de formulaire de zone Web](#) qui utilise le [moteur de rendu Web intégré](#). En tant que telle, elle se comporte comme n'importe quelle autre zone Web, à laquelle vous pouvez faire exécuter du code Javascript en appelant la commande 4D [WA Evaluate Javascript](#).

Etant donné que 4D View Pro est alimenté par la [solution de feuille de calcul SpreadJS](#), vous pouvez également appeler les méthodes Javascript de SpreadJS dans les zones 4D View Pro.

Exemple concret : masquer le ruban

4D View Pro étant une zone Web, vous pouvez sélectionner un élément de page Web et modifier son comportement à l'aide de Javascript. L'exemple suivant permet de masquer le [ruban](#) spreadJS :

```
//Méthode objet du bouton

var $js; $answer : Text

$js:="document.getElementsByClassName('ribbon')[0].setAttribute('style','d
$js+="window.dispatchEvent(new Event('resize'));""

$answer:=WA Evaluate JavaScript(*; "ViewProArea"; $js)
```

Appeler des méthodes JavaScript de SpreadJS

Vous pouvez exploiter la bibliothèque de méthodes Javascript de SpreadJS et les appeler directement pour gérer vos feuilles de calcul.

4D dispose d'une propriété [Utils.spread](#) intégrée qui pointe vers la feuille de calcul (également appelée [workbook](#)) dans la zone 4D View Pro, ce qui simplifie l'appel des [méthodes de Workbook](#) SpreadJS.

Exemple

Le code suivant annule la dernière action dans la feuille de calcul :

```
WA Evaluate JavaScript(*; "ViewProArea"; "Utils.spread.undoManager().undo()
```

Dépôt 4D View Pro Tips

[4D-View-Pro-Tips](#) est un dépôt GitHub qui contient un projet rempli de fonctions utiles pour vous permettre de gérer les images flottantes, de trier les colonnes ou les lignes, de créer une culture personnalisée, et bien plus encore ! N'hésitez pas à le cloner et à l'expérimenter !