4D Business Kit®

Langage
Windows®/Mac™OS

4D Business Kit
Langage

Sommaire

1 Introduction

4D Business Kit contient un métalangage de haut niveau, issu du langage de l'environnement de développement de 4e Dimension.

Le principe d'utilisation du langage de 4D Business Kit consiste à insérer des commandes au sein des pages HTML composant le site. Ces commandes, au moment de l'envoi des pages aux navigateurs, interagissent avec les données de la boutique. Ce fonctionnement permet de générer des pages dynamiques dont le contenu est entièrement personnalisé.

Les commandes de 4D Business Kit

Présentation

On nomme "commande 4D Business Kit" un mot commençant par **<!--#4DBK** et finissant par **-->**. Par exemple, **<!--#4DBKStoreSet/DEM1-->** est une commande 4DBK. Cela peut également être un lien (un URL), dans ce cas celui-ci contiendra une suite de commandes débutant par 4DBK et encadrées par **<a href="4DBKExecute:** et **">**.

Dans certains cas (conditions et boucles), une commande de fin doit toujours compléter une commande de départ (par exemple **<!--#4DBKIf-->** et **<!--#4DBKEndIf-->**).

En général, lorsqu'il s'agit d'inclure une commande 4D Business Kit au sein d'une autre instruction 4D Business Kit, il est inutile d'ajouter les caractères **<!--#** au début et **-->** pour chaque commande. Par exemple :

```
<a href="4DBKExecute:4DBKStoreSet/DEM1;4DBKQuantitySet/+1">
        //4DBKStoreSet et 4DBKQuantity ne sont pas encadrées ici.
```

Commandes et fonctions

Certaines commandes de 4D Business Kit retournent des valeurs en lieu et place de la commande dans la page HTML où elles sont insérées. Les commandes qui retournent une valeur sont appelées des *fonctions*.

Par exemple :

<!--#4DBKProcessingTime-->

Cette fonction est remplacée par une durée (exprimée en millisecondes) dans la page où elle est appelée.

Paramètres des commandes

La majorité des commandes 4D Business Kit admettent un ou plusieurs *paramètres*. Un paramètre permet de préciser l'objet, la portée ou la valeur de la commande.

Pour la plupart des commandes, la syntaxe est la suivante : **nom_de_la_commande** suivi d'une barre oblique "***/***", puis de la valeur du paramètre. Par exemple, l'instruction **<!--#4DBKStoreSet/DRES-->** comporte le paramètre **DRES** (nom de la boutique).

Les paramètres de toutes les commandes peuvent être des résultats de fonctions 4D Business Kit (telles que **4DBKVar** ou **4DBKSelectionSize**). Par exemple, vous pouvez écrire **<!--#4DBKRecordSet/4DBKSelectionSize-->**.

> *Note :* Ce principe ne s'applique pas aux paramètres de type "format d'affichage" (cf. chapitre "Formats d'affichage et fonctions", page 223).

Les paramètres des commandes effectuant des tests sur des chaînes de caractères *doivent* être encadrés par des guillemets. Par exemple, l'instruction **<!--#4DBKIf(4DBKField/T01=montexte)-->** retourne une erreur. La syntaxe correcte est dans ce cas **<!--#4DBKIf("4DBKField/T01"="montexte")**.
En revanche, cette syntaxe ne s'applique pas aux paramètres numériques, date, booléens, etc. Par exemple, la syntaxe de l'instruction **<!--#4DBKIf(4DBKField/VR02<3,00)** est correcte.

Certaines commandes acceptent plusieurs paramètres. Dans ce cas, les paramètres doivent être séparés par des virgules ou des points. Par exemple : **<!--#4DBKField/PrItemsINV.EUR,### ##0,00 EUR-->** indique que vous souhaitez afficher le champ **PrItemsINV** en euros et le format d'affichage "### ##0,00 EUR".

Enfin, d'autres commandes ont une syntaxe particulière : par exemple, les paramètres de la commande **4DBKExecute** doivent être insérés après deux-points (:).

Paramètres optionnels

Les paramètres des commandes sont parfois optionnels. Dans ce manuel, les paramètres optionnels sont indiqués entre crochets [].

Par exemple :

```
<!--#4DBKOrderValidate[/pageRéussite[,pageEchec]]-->
```

Les paramètres **pageRéussite** et **pageEchec** sont ici optionnels. A noter que cette commande comporte deux paramètres optionnels. Dans ce cas, vous ne pouvez utiliser le second paramètre optionnel que si vous avez utilisé le premier.

Accès aux champs 4D Business Kit

A l'aide des commandes 4D Business Kit et de leurs paramètres, vous pouvez accéder à toutes les valeurs présentes dans votre boutique : quantités, prix, noms des clients et des produits, etc.

Pour référencer une valeur dans les paramètres des commandes, il suffit de saisir son code 4D Business Kit (par exemple T01 ou C02) ou son nom (lorsqu'un nom a été associé au code dans le masque de la boutique).

Imaginons que vous souhaitiez afficher le nom d'un article, défini comme Texte 1 (T01). Ce code a été associé au nom "produit". Pour afficher ce champ dans votre page HTML, vous pouvez écrire :

```
<!--#4DBKField/produit-->
```

ou bien

```
<!--#4DBKField/T01-->
```

Référencer les variables par leur nom vous permet de visualiser plus facilement le contenu de vos modèles de pages, mais rend les pages moins interchangeables — le code n'est pas générique. En outre, il vous sera nécessaire de modifier les pages HTML modèles si vous changez des noms dans les masques.

Lorsque les variables sont référencées par leur code, une même page peut être plus facilement réutilisée dans différentes boutiques (si leurs structures sont proches) par un simple copier-coller du code. Celui-ci est en revanche moins lisible.

Intégration de Javascript

Vous pouvez parfaitement combiner les commandes 4D Business Kit avec du *JavaScript*. Le JavaScript est un langage destiné à compléter les possibilités du HTML.

Avec 4D Business Kit, vous pourrez utiliser vos propres librairies JavaScript. En outre, 4D Business Kit est livré avec plusieurs librairies JavaScript utiles, vous permettant d'inclure des fonctionnalités supplémentaires dans vos pages modèles : vérification des valeurs saisies, recherches, etc. Simples d'emploi, elles ne nécessitent pas l'apprentissage du JavaScript.

Ces librairies sont placées dans le dossier "Librairies" du dossier principal de l'application 4D Business Kit. Elles sont décrites dans le chapitre 1 du *Manuel de référence* de 4D Business Kit.

Le principe de combinaison des commandes 4D Business Kit avec le Javascript est identique à celui des pages HTML : les commandes 4D Business Kit sont insérées sous forme de commentaires HTML. Elles sont traitées au moment de l'envoi de la page et sont éventuellement remplacées par des valeurs issues du traitement.

Des exemples d'intégration de Javascript sont fournis dans les pages des boutiques modèles de 4D Business Kit.

Intégration des commandes dans les pages HTML

Principes

La plupart des commandes doivent être insérées sous forme de commentaire HTML, avec une syntaxe du type :

```
<!--#commande/paramètre(s)-->
```

Par exemple :

```
<!--#4DBKStoreSet/DEM1-->
```

Ces instructions sont exécutées en une seule opération, lors du chargement de chaque page, selon l'ordre dans lequel elles sont rencontrées dans la page lue.

D'autres commandes peuvent être insérées sous la forme d'URLs ou d'actions de formulaires, avec une syntaxe du type :

```
<a href="4DBKExecute:commande/paramètre(s);commande/
        paramètre(s)...">lien</a>
```

Par exemple :

```
<a href="4DBKExecute:4DBKStoreSet/DEM1;4DBKSelectionSet/1;
4DBKCursorNext/2;4DBKNoCache">Suivant</a>
```

Commentaires HTML

Les commentaires HTML sont utilisés pour indiquer au serveur 4D Business Kit les éléments à insérer dans une page avant de l'envoyer au navigateur. Ils n'apparaissent jamais sur les pages envoyées aux navigateurs Web. Les commandes 4D Business Kit insérées sous forme de commentaires provoquent un traitement dans le serveur Web et/ou sont éventuellement remplacées par des valeurs issues d'un traitement.

Par exemple, l'instruction **<!--#4DBKStoreSet/DRES-->** indique que nous souhaitons travailler avec la boutique dont le code est **DRES**. Nous insérons cette commande au début de la page HTML modèle. Au moment de l'envoi de la page au navigateur, 4D Business Kit tiendra compte de cette information et toutes les autres instructions présentes dans la page s'appliqueront à cette boutique. Une fois les instructions exécutées, 4D Business Kit envoie la page. Le libellé **<!--#4DBKStoreSet/DRES-->**, quant à lui, n'apparaîtra pas sur la page envoyée au navigateur.

D'autres types de commentaires sont en outre remplacés par la valeur issue du traitement. Par exemple, l'instruction **<!--#4DBKField/produit-->** sera remplacée, au moment de l'envoi de la page, par la valeur courante du champ "produit" dans la boutique précédemment définie. Le libellé **<!--#4DBKField/produit-->** ne s'affichera pas sur le navigateur, seule la valeur réelle du champ sera affichée.

Sur le même principe, les commentaires HTML 4D Business Kit peuvent être insérés à la place de toute valeur : dans les références d'images (**<img src="<!--#4DBKThumbnails/1-->...**), les champs de saisie (**<input type="text" value="<!--#4DBKField/CusCode-->"...**), etc.

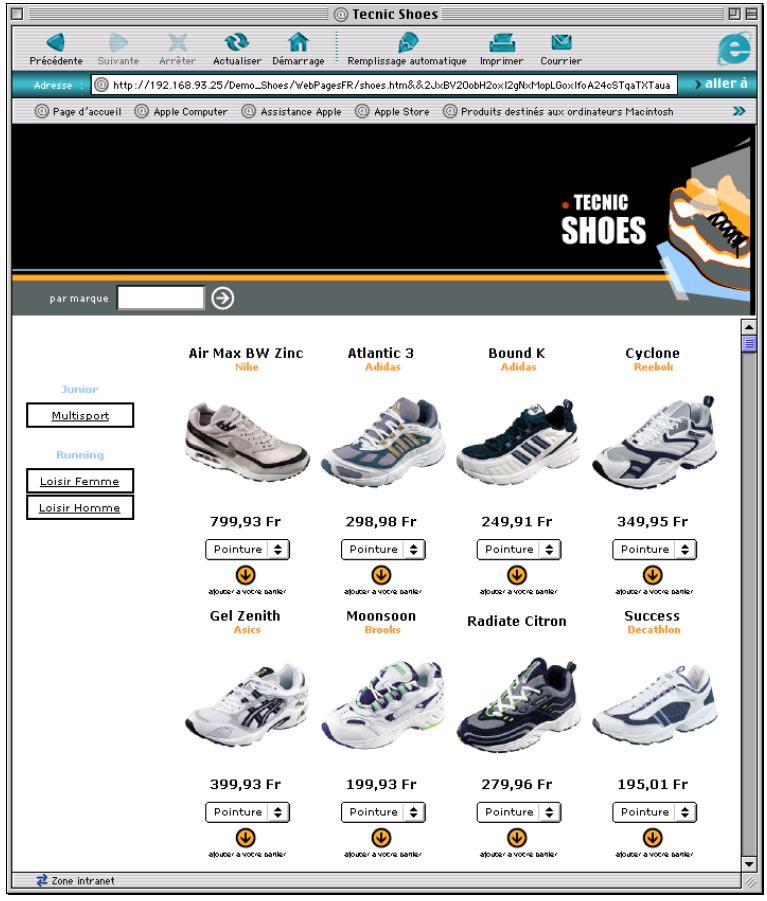
URLs et actions de formulaires

Les URLs et actions de formulaires, également placés dans les pages modèles, définissent les actions proposées au navigateur. Ils sont traités par le serveur 4D Business Kit au moment de la réception de la requête en provenance du navigateur.

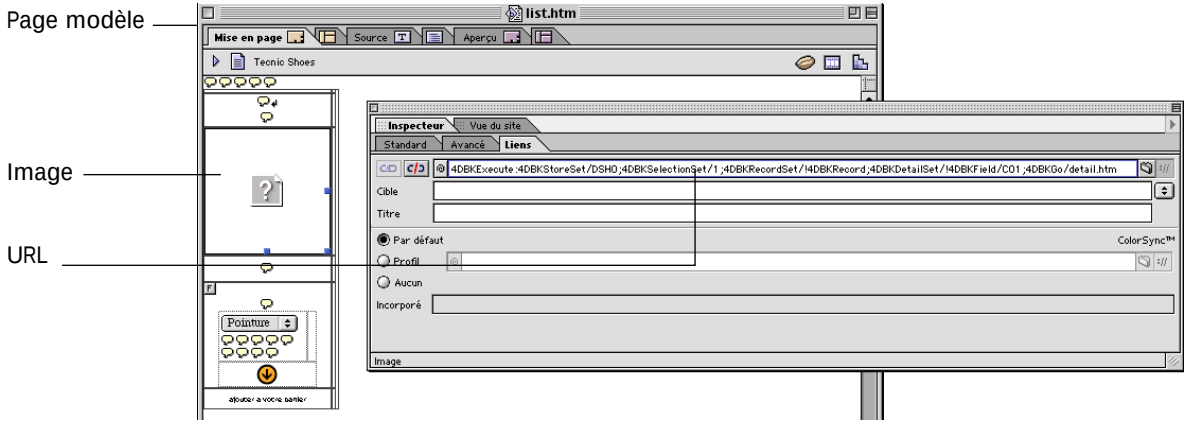
Par exemple, lorsqu'une page d'articles est affichée en mode liste, chaque élément de la liste comporte généralement un URL permettant d'afficher une page plus détaillée lorsque l'utilisateur clique dessus.

Dans ce cas, il suffit d'insérer dans la page modèle les commandes 4D Business Kit adéquates afin de définir la cible de l'URL. Lorsque l'utilisateur cliquera sur le lien, le serveur 4D Business Kit recevra et traitera la requête, puis retournera la page demandée.

Par exemple, la boutique suivante affiche sur la même page une liste de différents articles :



Pour chaque article, l'URL suivant est défini dans la page modèle :



Cet URL utilise plusieurs commandes 4D Business Kit afin de définir la boutique, la sélection, l'enregistrement et enfin la page à afficher.

Calculs des liens

Les liens sont codés à l'aide de chemins d'accès relatifs.

Dans le cas des liens, les commandes sont exécutées au moment où l'utilisateur clique sur le lien. Toutefois, pour les commandes retournant une valeur, il est possible de déclencher leur exécution au moment du chargement de la page. Pour cela, il suffit d'ajouter le symbole ! (point d'exclamation) devant le nom des commandes (par exemple **!4DBKField/T01** au lieu de **4DBKField/T01**).

Par exemple, le lien suivant :

```
<a href="4DBKExecute:4DBKStoreSet/DEM1;4DBKSelectionSet/1;
        4DBKRecordSet/!4DBKVar/VL01;4DBKNoCache">Lien</a>
```

... sera remplacé, au chargement de la page, par :

```
<a href="4DBKExecute:4DBKStoreSet/DEM1;4DBKSelectionSet/1;
        4DBKRecordSet/32;4DBKNoCache">Lien</a>
```

Note sur les liens vers des pages d'autres sites

Dans le cas où votre site contient des liens référençant des pages .htm, .html, .shtm ou .shtml extérieures au serveur 4D Business Kit, vous devez ajouter le signe "-" (moins) à la fin des URLs.

Par exemple :

```
<a href="http://www.4d.fr/htmlfiles/BKindex.shtml-">Lien</a>
```

Sinon, le mécanisme de gestion des sessions de 4D Business Kit empêchera le lien de fonctionner.

Note : Ce principe s'applique uniquement aux références de *pages* et non aux références de *sites* (vous pouvez écrire **Lien**).

Objets du langage 4DBK

Champs

Les champs de description d'articles peuvent être désignés soit par leur référence absolue (par exemple "T01"), soit par leur nom défini dans les masques (par exemple "Produit").

Structures de programmation

Il est possible d'utiliser des boucles de programmation (commande **#4DBKLoop**) comportant jusqu'à 20 niveaux de profondeur pour définir des catalogues dont les éléments se répètent, des tableaux à 2 dimensions, des listes alternées, etc.

Il est possible de créer des structures conditionnelles multi-niveaux à l'aide des opérateurs **#4DBKIf** et **#4DBKCaseof**.

Variables

Il est possible de définir des variables numériques ou entières (jusqu'à 100 de chaque type), les afficher dans des formats différents (par exemple ### ##0,00 ou ###,##0.00), tester leur valeur et les faire évoluer à l'aide des opérateurs numériques standard +,-,/,*.

Il est possible de définir des variables textes (jusqu'à 100), les afficher dans des formats différents (par exemple troncature à 10 caractères ou 2 mots, passage en majuscules, etc.) et tester leur valeur.

Sélections

Chaque utilisateur peut disposer de 9 sélections simultanées et d'une sélection particulière pour le panier. Ces 9 sélections sont disponibles par boutique/par utilisateur et ne dépendent pas des pages affichées. Elles sont accessibles pendant toute la session de navigation.

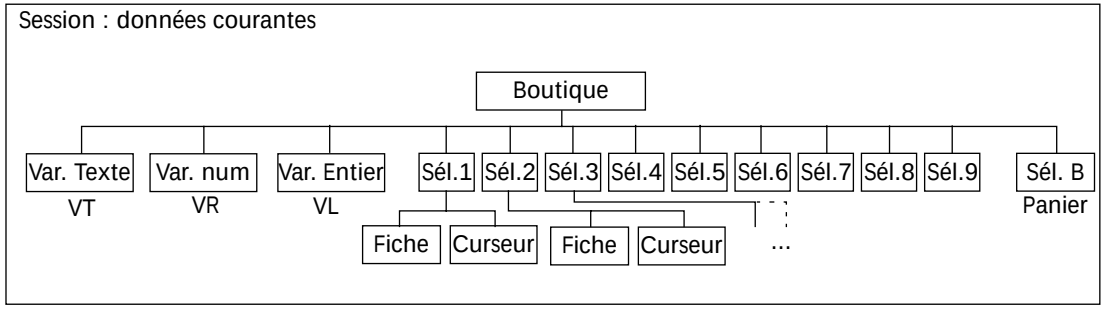
Concernant la gestion des sélections, 4D Business Kit émule les notions d'**ensembles** de 4e Dimension (possibilité d'effectuer des intersections, des unions, différence), ainsi que les fonctions de type **réduire sélection**, **valeurs distinctes** et **tris**.

Il est possible d'effectuer des recherches de type ET et OU multicritères.

Chaque sélection possède un pointeur courant de fiche (le **curseur**) qui permet de gérer les notions de pages d'articles et la navigation entre les articles.

Les sélections et les pointeurs, tout comme les variables, sont stockés dans une session mémoire pour chaque utilisateur et ne dépendent pas du chargement des pages.

Durée de vie des objets 4D Business Kit



2 Exemples de programmation

Ce chapitre a pour objectif de présenter l'utilisation des commandes 4D Business Kit au travers d'exemples simples qui vous permettront d'être rapidement opérationnel.

Voici les exemples de programmation qui seront traités dans ce chapitre :

- afficher une liste d'articles,
- créer un menu URL.

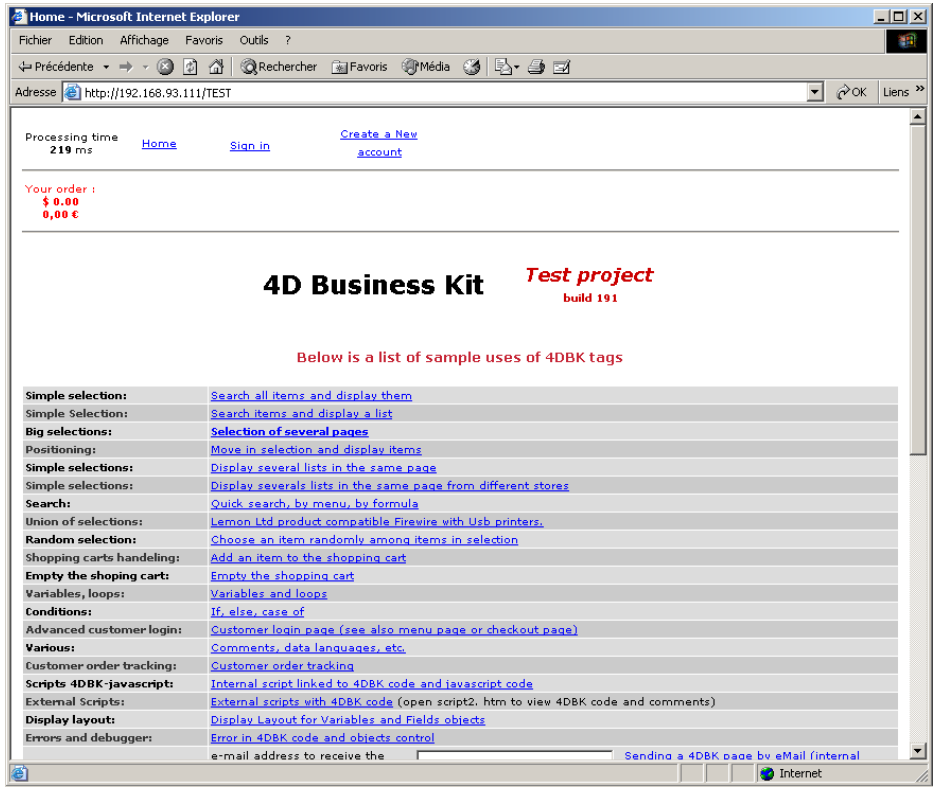
A propos de la boutique TEST

4D Business Kit est livré avec un espace multi-boutique nommé "Test", comportant la boutique "Projet Test 4D Business Kit" (code TEST).

Les pages de cette boutique (en anglais) contiennent de nombreux exemples commentés de programmation pour toutes les fonctions de 4D Business Kit (articles, clients, commandes, administration, utilisation de JavaScripts, etc.).

Pour accéder à chaque exemple de fonctionnement, il suffit de cliquer sur le lien correspondant dans la page d'accueil :

Page d'accueil du site TEST



Vous pouvez également observer le code HTML des pages du site, situées dans le dossier [Test_Site].

Nous vous conseillons d'examiner attentivement les exemples proposés par cette boutique, permettant de bien comprendre les mécanismes mis en jeu.

Création d'une page simple affichant des articles

Le but de cet exercice est d'afficher une liste telle que celle-ci :

Pour cela, nous aurons besoin d'effectuer les opérations suivantes :

- créer une sélection d'articles,
- afficher ces articles selon un gabarit.

Sélectionner des articles

Nous devrons utiliser les instructions suivantes au sein de la page :

```
<!--#4DBKStoreSet/MCOM-->
<!--#4DBKSelectionSet/1-->
<!--#4DBKQuerySet/couleur=noir-->
<!--#4DBKSortSet/type,produit-->
<!--#4DBKSelectionSort-->
<!--#4DBKRecordSet/1-->
```

Examinons ces instructions en détail :

- **<!--#4DBKStoreSet/MCOM-->**
  Nous passons sur la boutique définie par le code MCOM. La boutique courante ne changera que si le même utilisateur charge une page ou clique sur un lien qui appelle la commande 4DBKStore avec un autre code boutique.
- **<!--#4DBKSelectionSet/1-->**
  Nous choisissons d'utiliser la sélection numéro 1 pour stocker le résultat de notre recherche.
- **<!--#4DBKQuerySet/couleur=noir-->**
  "couleur" est le nom du champ qui désigne la couleur de nos articles. Nous l'avons défini ainsi au sein du masque principal. Nous recherchons donc tous les articles de couleur noire. La liste des articles correspondants sera stockée dans la sélection numéro 1.
- **<!--#4DBKSortSet/type,produit-->**
  Fixe le critère de tri de la sélection sur les champs "type" puis "produit".
- **<!--#4DBKSelectionSort-->**
  Effectue le tri de la sélection numéro 1, les articles sont maintenant triés par type puis par nom.
- **<!--#4DBKRecordSet/1-->**
  Nous nous plaçons maintenant sur la 1re fiche de la sélection.

Afficher les articles

Afin d'afficher les articles, nous utiliserons les instructions suivantes au sein de la page :

```
<!--#4DBKLoop/VL01,1,4DBKSelectionSize-->
<table border="0"> <tr>
<td> <img src="<!--#4DBKThumbnail/1-->"> </td>
<td> <!--#4DBKField/Product--> </td>
<td> <!--#4DBKField/Manufacturer--> </td> </tr>
</table>
<!--#4DBKRecordNext-->
<!--#4DBKEndLoop/VL01-->
```

Examinons ces instructions en détail :

- **<!--#4DBKLoop/VL01,1,4DBKSelectionSize-->**
  Nous déclarons le début de la boucle qui va afficher la liste. Notez qu'une instruction 4DBKLoop doit toujours avoir son pendant 4DBKEndLoop. Le code HTML compris entre 4DBKLoop et 4DBKEndLoop se répète ici 4DBKSelectionSize fois, c'est-à-dire autant de fois qu'il y a d'articles dans la sélection numéro 1.

- **<table border="0"> <tr> ... </tr> </table>**
  Nous créons un tableau de 3 colonnes.
- **<td> <img src="<-- #4DBKThumbnail/1-->"> </td>**
  Nous affichons le champ de l'article qui correspond à l'image réduite.
- **<td> <!--#4DBKField/Produit--> </td>**
  Nous affichons le champ de l'article qui correspond au nom du produit.
- **<td> <!--#4DBKField/fabricant--> </td> </tr>**
  Nous affichons le champ de l'article qui correspond à la famille définie comme "fabricant" et nous passons à la ligne suivante du tableau
- **</table>**
  Nous finissons le tableau.
- **<!--#4DBKRecordNext-->**
  Nous passons à la fiche suivante.
- **<!--#4DBKEndLoop/VL01-->**
  Nous fermons la boucle.

Création d'un menu URL

Le but de cet exercice est de créer un menu URL dans une page Web.

Cette page Web contiendra l'ensemble des catégories d'articles de votre fichier et vous permettra de visualiser le catalogue correspondant aux articles grâce à une sélection.

Pour cela, nous aurons besoin d'effectuer les opérations suivantes :

- obtenir la liste des différentes catégories du fichier d'articles,
- boucler sur cette liste pour générer chaque ligne comprise entre la balise <select> et la balise </select>.

Rappel de notions de HTML

Un menu en HTML se présente toujours sous la forme suivante :

```
<select name="select">
    <option value="data 1">choix 1</option>
    <option value="data 2">choix 2</option>
    <option value="data 3">choix 3</option>
    <option value="data 4">choix 4</option>
</select>
```

- Balises
  Ce menu doit être situé dans des balises <form> pour être parfaitement compatible avec tous les navigateurs Web.
- “choix n” (valeur affichée) et “donnée n” (valeur retournée) : “choix n” représente le n^{ième} élément affiché, tandis que “donnée n” représente, la n^{ième} donnée de la liste, c’est-à-dire l’élément qui nous sera renvoyé lorsqu’un utilisateur cliquera sur ce menu.

Obtenir la liste des différentes catégories du fichier d’articles

Pour obtenir cette liste, nous allons lancer une recherche sur tous les articles et ne garderons que ceux dont la catégorie n’a pas déjà été retenue. Pour les utilisateurs du logiciel 4^{e} Dimension, cela correspond à la commande VALEURS DISTINCTES.

```
<!--#4DBKStoreSet/MCOM-->
<!--#4DBKSelectionSet/1-->
<!--#4DBKQuerySet/all-->
<!--#4DBKSelectionDistinct/categorie-->
<!--#4DBKSortSet/categorie-->
<!--#4DBKSelectionSort-->
<!--#4DBKRecordSet/1-->
```

Examinons ces instructions en détail :

- **<!--#4DBKStoreSet/MCOM-->**
  Nous utilisons cette instruction pour passer sur la boutique concernée. Cette instruction n’est pas nécessaire si plus haut dans la page HTML vous l’avez déjà utilisée et qu’entre-temps vous n’avez pas changé de boutique.
- **<!--#4DBKSelectionSet/1-->**
  Nous choisissons d’utiliser la sélection numéro 1 pour effectuer notre recherche.
- **<!--#4DBKQuerySet/all-->**
  "all" signifie “tous”. Nous recherchons tous les articles du fichier.

Nous allons donc obtenir dans la sélection numéro 1, une série de données comme ci-dessous :

Produit	Catégorie	Marque	Couleur	Prix HT
Imac DV 400	Ordinateurs de bureau	Apple	Bleu	9 000,00
Souris USB 2 boutons	Accessoires	MacAlly	Orange	200,00
Imac DV 450	Ordinateurs de bureau	Apple	Gris	10 000,00
HP 1240C	Imprimante encre	HP	Blanc	1 400,00
HP 2100TN	Imprimante laser	HP	Blanc	4 500,00
Cable réseau 10 mètres	Accessoires	Cablin	Noir	200,00

- **<!--#4DBKSelectionDistinct/categorie-->**
  Cette instruction balaye toute la sélection que nous venons de créer pour n'en garder que les articles dont la valeur du champ nommé "catégorie" (dans notre exemple) est différente de celle des autres.

Nous obtenons dans la sélection numéro 1 :

Produit	Catégorie	Marque	Couleur	Prix HT
Imac DV 400	Ordinateurs de bureau	Apple	Bleu	9 000,00
Souris USB 2 boutons	Accessoires	MacAlly	Orange	200,00
HP 1240C	Imprimante encre	HP	Blanc	1 400,00
HP 2100TN	Imprimante laser	HP	Blanc	4 500,00

- **<!--#4DBKSortSet/categorie-->**
  Nous fixons le tri de la sélection numéro 1 par catégorie.
- **<!--#4DBKSelectionSort-->**
  Nous effectuons ce tri.

Nous obtenons dans la sélection numéro 1 :

Produit	Catégorie	Marque	Couleur	Prix HT
Souris USB 2 boutons	Accessoires	MacAlly	Orange	200,00
HP 1240C	Imprimante encre	HP	Blanc	1 400,00
HP 2100TN	Imprimante laser	HP	Blanc	4 500,00
Imac DV 400	Ordinateurs de bureau	Apple	Bleu	9 000,00

- **<!--#4DBKRecordSet/1-->**
  Nous nous positionnons sur la première fiche de la sélection, c'est-à-dire la Souris USB 2 boutons.

Effectuer une boucle sur la liste afin de générer les choix possibles du menu

Générer les différents choix possibles du menu signifie que nous allons générer les lignes correspondant aux balises "<option value>".

Pour cela, nous devons effectuer une boucle sur cette sélection.

Note : Pour rappel, une boucle de programmation se définit par une variable numérique qui va démarrer à la valeur 1 et s'incrémenter jusqu'à la valeur finale (n) correspondant au nombre d'itérations à effectuer.

Dans notre exemple, la valeur finale étant 4, nous allons boucler 4 fois sur cette sélection selon le code suivant :

```
<!--#4DBKLoop/VL01,1,4DBKSelectionSize-->
<option value="<!--#4DBKField/categorie,U&L-->"> <!--#4DBKField/catego-
        rie--> option>
<!--#4DBKRecordNext-->
<!--#4DBKEndLoop/VL01-->
```

Examinons ces instructions :

- **<!--#4DBKLoop/VL01-->**
  Nous démarrons la boucle. Tout ce qui est compris entre 4DBKLoop et 4DBKEndLoop sera répété 4 fois (la fonction 4DBKSelectionSize retourne la valeur 4).
- **<option value="<!--#4DBKField/categorie,U&L-->" > <!--#4DBKField/categorie--> option>**
  Nous générons la partie <option value= en utilisant l'instruction <!--#4DBKField/categorie,U&L--> qui retourne le champ "catégorie" de la fiche courante.
  Dans certains navigateurs, la valeur rendue lors d'un clic sur un menu ne doit pas contenir de caractère d'espacement. De même les accents ne sont pas transmis de la même manière sur tous les navigateurs. Nous convertirons donc la partie "donnée n" en majuscules et dans un format "quoted" universel à l'aide de l'instruction <!--#4DBKField/categorie,U&L-->
  Nous générons la valeur de la catégorie de la fiche qui nous sera retournée (choix) lors d'un clic sur ce menu ainsi que la catégorie qui sera affichée grâce à <!--#4DBKField/categorie-->.
  Pour rappel nous devons obtenir notre code sous la forme :

  ```
  <option value="donnee">choix</option>
  ```
- **<!--#4DBKRecordNext-->**
  Cette instruction permet de passer à la fiche suivante (dans notre exemple, le numéro s'incrémente de 1 en 1 jusqu'à 4).
- **<!--#4DBKEndLoop/VL01-->**
  Nous indiquons la fin de la boucle.

La boucle et le code qu'elle contient génèrent donc les instructions suivantes :

```
<option value="ACCESSOIRES">Accessoires</option>
<option value="IMPRIMANTE+ENCRE">Imprimante encre</option>
<option value="IMPRIMANTE+LASER">Imprimante laser</option>
<option value="ORDINATEURS+DE+BUREAU">Ordinateurs de bureau</op-
        tion>
```

3 Aide au débogage du code 4D Business Kit

Lorsque vous développez et testez votre code 4D Business Kit inséré dans les pages HTML, il est important que vous puissiez repérer et corriger les éventuelles erreurs qui s'y sont glissées.

Par ailleurs, même si tout fonctionne comme vous le souhaitez, vous pouvez avoir besoin de visualiser ce qui se passe en coulisses, c'est-à-dire les valeurs et les calculs manipulés par le serveur 4D Business Kit.

Pour cela, 4D Business Kit propose plusieurs types d'outils permettant de contrôler et de déboguer votre code :

- un analyseur syntaxique (automatique) détectant instantanément les erreurs de syntaxe,
- une fenêtre de trace, affichant les valeurs courantes de toutes les variables,
- une fenêtre d'historique, permettant de visualiser le code exécuté.

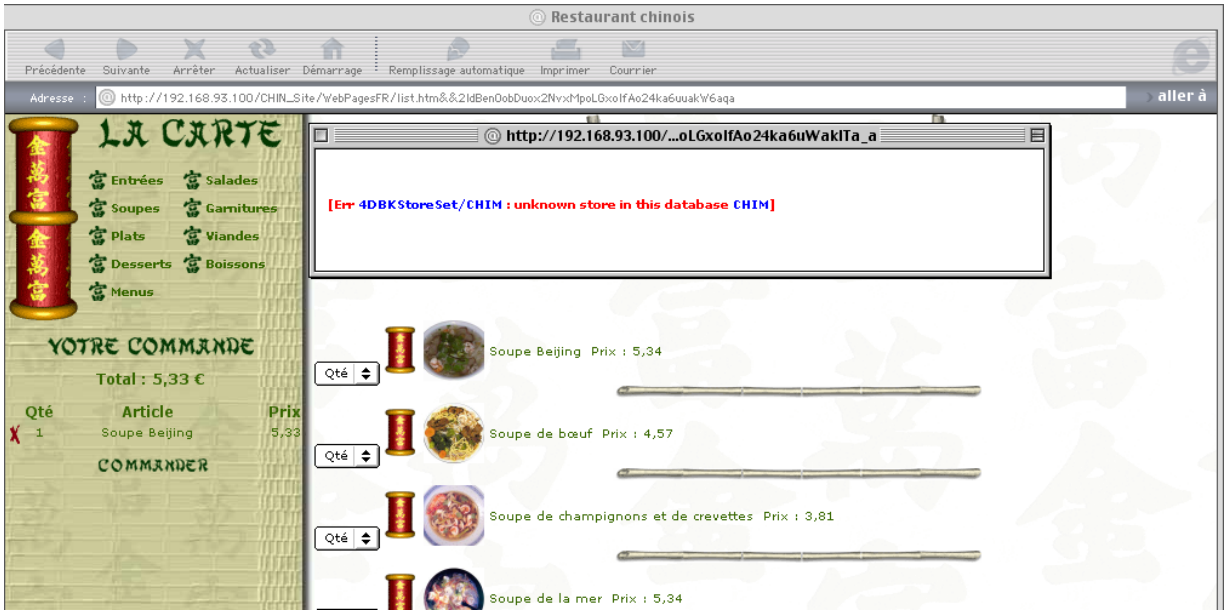
Note : Pour plus d'informations sur les commandes utilitaires proposées par 4D Business Kit, reportez-vous au chapitre 18, "Utilitaires" page 147.

Analyseur syntaxique

L'analyseur syntaxique de 4D Business Kit est activé en permanence. Il ne nécessite l'insertion d'aucune commande particulière.

L'analyseur syntaxique permet de détecter automatiquement toute erreur de programmation au moment de l'envoi des pages Web.

Lorsqu'une erreur est détectée, une fenêtre d'alerte est affichée au-dessus de la page Web dans le navigateur :



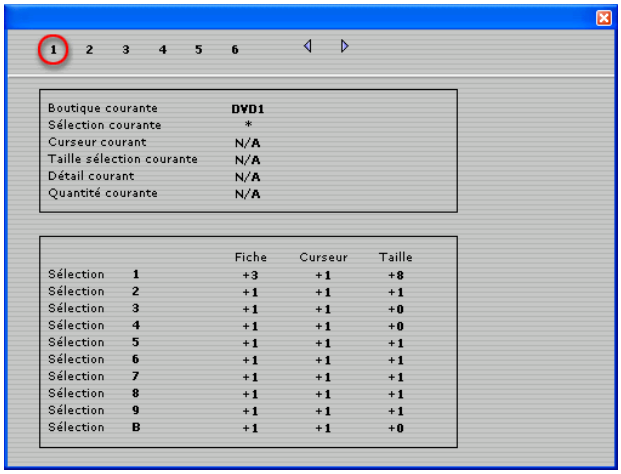
Dans l'exemple ci-dessus, le code de la boutique a été mal orthographié dans une instruction. Au moment de l'exécution de l'instruction, 4D Business Kit détecte l'erreur et affiche automatiquement une fenêtre d'alerte avec un message expliquant l'origine du bogue.

Pour pouvoir repérer ce type d'erreur, il vous suffit donc de tester le fonctionnement de votre boutique à l'aide d'un navigateur.

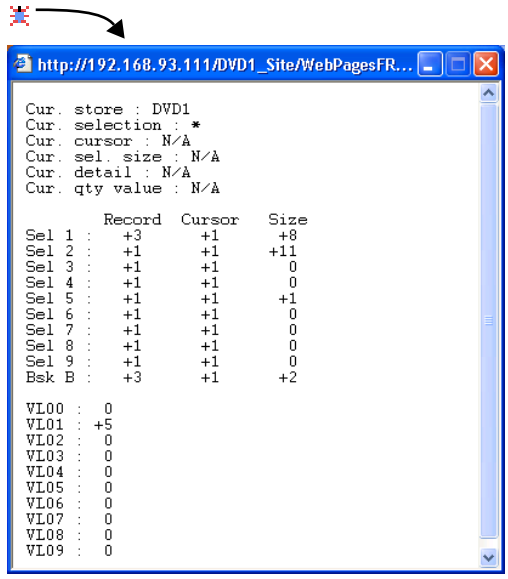
De nombreux types d'erreurs sont détectés par l'analyseur syntaxique : fautes de frappe, erreurs de syntaxe, etc. La liste complète des messages d'erreur pouvant être retournés par 4D Business Kit est fournie dans l'Annexe A, page 233.

Fenêtre de trace

4D Business Kit vous permet également de programmer l'affichage sur le serveur ou dans le navigateur d'une **fenêtre de trace** détaillée, affichant la valeur courante de toutes les variables et des principaux paramètres :



Fenêtre de trace affichée **sur le serveur**



Fenêtre de trace affichée **sur le navigateur**

Les paramètres fournis sont les suivants :

- **Boutique courante** : code de la boutique courante
- **Sélection courante** : numéro de la sélection courante
- **Curseur courant** : position du curseur courant
- **Taille sélection courante** : nombre d'articles dans la sélection courante
- **Détail courant** : numéro de la fiche détail courante
- **Quantité courante** : quantité d'articles ajoutée dans le panier
- Position du **curseur** et du numéro de **fiche courante** pour chacune des 6 sélections.

La fenêtre représente une "photographie" statique des valeurs courantes des paramètres à l'instant précis où la commande d'appel est exécutée. Par conséquent, il est conseillé d'insérer la commande à l'emplacement précis où les valeurs doivent être visualisées.

Il apparaît une nouvelle fenêtre de trace sur le serveur ou sur le navigateur pour chaque commande d'appel de la fenêtre exécutée.

Affichage sur le serveur

Pour afficher une fenêtre de trace sur le serveur 4D Business Kit, insérez la commande 4DBKTraceOnServer à l'emplacement auquel vous souhaitez visualiser les informations.

Affichage sur le navigateur

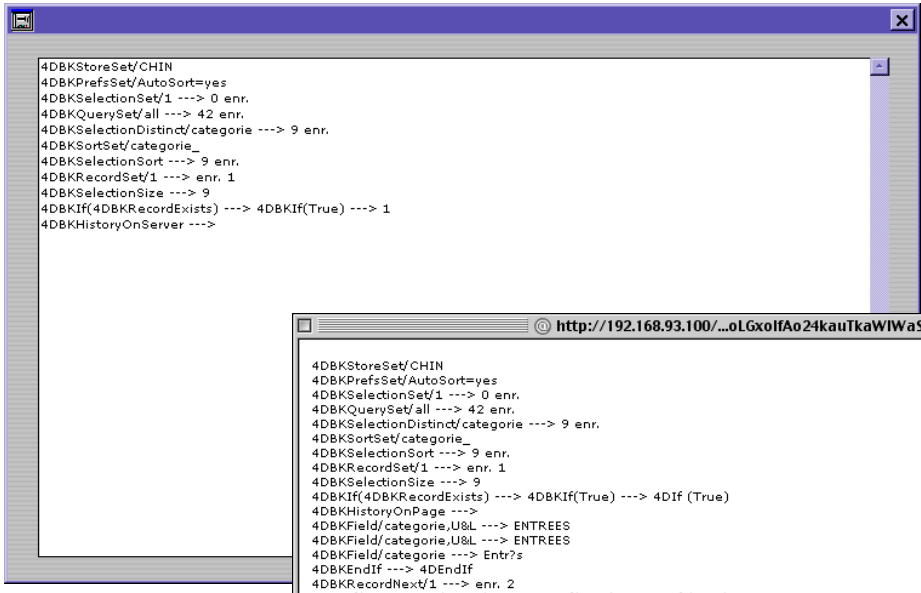
Pour afficher une fenêtre de trace dans le navigateur, insérez la commande 4DBKTraceOnPage à l'emplacement auquel vous souhaitez visualiser les informations. Cliquez sur l'icône  qui apparaît dans la page HTML pour afficher la fenêtre de trace.

Note : La fenêtre de trace est affichée à l'aide d'un JavaScript.

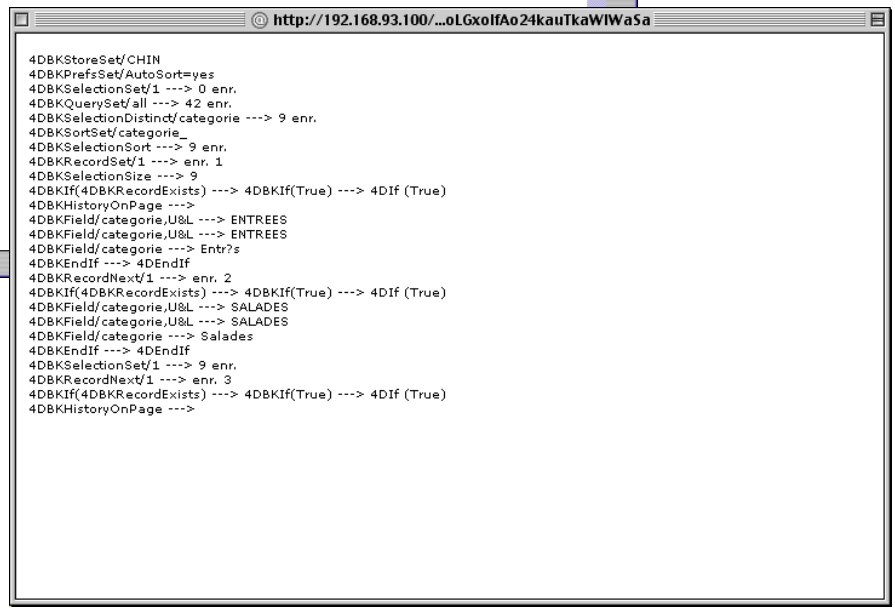
Fenêtre d'historique

4D Business Kit vous permet de visualiser les commandes 4DBK exécutées par le serveur Web ainsi que les valeurs manipulées par chaque commande. Cette possibilité est utile par exemple dans le cadre d'une programmation HTML utilisant des conditions et des boucles, pour visualiser le code réellement exécuté.

Ces informations sont affichées dans la **fenêtre d'historique**. Vous pouvez choisir d'afficher cette fenêtre sur le serveur ou sur le navigateur :



Fenêtre d'historique affichée **sur le serveur**



Fenêtre d'historique affichée **sur le navigateur**

La fenêtre liste toutes les commandes 4DBK exécutées par 4D Business Kit depuis le début de la page et jusqu'à l'emplacement de la commande d'appel. Par conséquent, il est conseillé d'insérer la commande juste après le code que vous souhaitez visualiser.

Il apparaît une nouvelle fenêtre d'historique sur le serveur ou sur le navigateur pour chaque commande d'appel de la fenêtre exécutée.

Affichage sur le serveur

Pour afficher une fenêtre d'historique sur le serveur 4D Business Kit, insérez la commande 4DBKHistoryOnServer à l'emplacement auquel vous souhaitez visualiser les informations.

Affichage sur le navigateur

Pour afficher une fenêtre d'environnement dans le navigateur, insérez la commande 4DBKHistoryOnPage à l'emplacement auquel vous souhaitez visualiser les informations.

Note : La fenêtre d'historique est affichée à l'aide d'un JavaScript.

4 Boutiques

4DBKStoreSet

<!--#4DBKStoreSet/boutique-->

Paramètre	Description	Valeur(s)
boutique	Code de la boutique défini dans le fichier de données	Chaîne de 4 caractères

Objectif : fixer la boutique courante.

Description

La commande 4DBKStoreSet désigne *boutique* comme boutique courante. Il est nécessaire d'appeler cette commande au début de chaque page contenant des balises 4DBK concernant les boutiques ou chaque fois que vous souhaitez changer de boutique courante.

Elle doit également être spécifiée au début de chaque lien contenant des balises 4DBK. En effet, ces liens peuvent être rangés par l'utilisateur dans les "favoris" des navigateurs et rappelés par la suite. Le serveur ne pourra rattacher les instructions contenues dans le lien qu'après exécution d'une première commande 4DBKStoreSet.

Cette commande exécutant un certain nombre d'opérations complexes, il est judicieux de ne l'appeler que lorsque cela s'avère nécessaire pour éviter toute perte de temps au serveur.

Conditions

La boutique doit exister dans le fichier de données.

Exemple

L'exemple suivant permet d'afficher dans une même page des données issues de boutiques différentes :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1 de la boutique MCOM
<!--#4DBKField/categorie--> //afficher la valeur courante du champ "categorie"
<!--#4DBKStoreSet/MDVD--> //passer sur la boutique MDVD
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1 de la boutique MDVD
<!--#4DBKField/C03--> //afficher la valeur courante du champ "C03"
```

5 Sélections

4DBKSelectionSet

<!--#4DBKSelectionSet/sélection-->

Paramètre	Description	Valeur(s)
sélection	Nouvelle sélection courante	• 1 à 9 : utiliser la sélection 1 à 9 de la boutique courante • B : utiliser le panier de la boutique courante

Objectif : Fixer la sélection courante.

Description

La commande 4DBKSelectionSet désigne *sélection* comme nouvelle sélection courante. Il est nécessaire d'appeler cette instruction chaque fois que vous souhaitez changer de sélection courante.

Le paramètre *sélection* accepte deux types de valeurs :

- de 1 à 9 : utiliser la sélection numéro 1 à numéro 9 de la boutique courante.
- B : utiliser le panier de la boutique courante.

La première fiche de la nouvelle sélection courante devient la fiche courante.

Conditions

Il faut avoir fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemple

L'exemple suivant affiche dans la même page des valeurs issues de deux sélections différentes : la sélection numéro 2 et le panier.

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/2--> //passer sur la sélection numéro 2
<!--#4DBKField/Type--> //afficher la valeur courante du champ "Type" de la
        sélection numéro 2
<!--#4DBKSelectionSet/B--> //passer sur la sélection du panier
<!--#4DBKField/Produit--> //afficher la valeur courante du champ "produit"
        du panier
```

Référence : 4DBKSelectionSort

4DBKQuerySet

<!--#4DBKQuerySet/requête-->

Paramètre	Description	Valeur(s)
requête	Chaîne de recherche	• ALL • Champ Opérateur Valeur [[and/or] Champ Opérateur Valeur]

Objectif : Effectuer une recherche et affecter l'ensemble de fiches trouvées à une sélection courante.

Description

La commande 4DBKQuerySet effectue une recherche sur la base de la formule définie dans le paramètre *requête* et place l'ensemble de fiches trouvées dans la sélection courante.

Le paramètre *requête* peut contenir différents types de valeurs :

- ALL : sélection de tous les articles de la boutique.
- Champ Opérateur Valeur :
  • Champ doit être un champ valide en recherche (voir chapitre Champs).
  • Opérateur est l'un des opérateurs de recherche suivants :
  = != < <= > >=
  • Valeur est la chaîne de caractères recherchée. Elle doit généralement être encadrée par des guillemets (simples ou doubles). Valeur peut être le résultat d'une instruction 4D Business Kit.

Il ne doit pas y avoir d'espaces de part et d'autre de la condition (par exemple type="imprimante" est valide, type = "imprimante" n'est pas valide).

Il est préférable d'encadrer les valeurs par des guillemets doubles (par exemple type="imprimante") ou des guillemets simples (par exemple type='imprimante'). Dans certains cas, il n'est pas possible d'utiliser des guillemets, par exemple avec des fonctions Javascript exécutées dans un lien (du type "javascript: instruction"), car il n'est alors possible d'employer que 2 niveaux de guillemets (" et '). Dans ce cas, il n'est pas nécessaire d'encadrer les valeurs, vous pourrez alors écrire uniquement type=imprimante.

Les opérateurs doivent impérativement être encadrés par un seul espace (par exemple type="ordinateur" and fabricant="apple" est valide, en revanche type="ordinateur" and fabricant="apple" n'est pas valide).

> *Note :* Vous pouvez également utiliser les symboles "&" et "|" à la place de "and" et "or".

Ces champs peuvent être complétés par l'opérateur @ (arobase) qui placé devant, au milieu ou derrière va signifier :
ordin@ : tout ce qui commence par ordin...
@teur : tout ce qui finit par ...teur
@nat@ : tout ce qui contient ...nat...
ordi@teur : tout ce qui est de la forme ordi...teur

> *Note :* A l'aide de cette commande, il est possible de générer des sélections sur la base de recherches complexes. Néanmoins, pour des recherches encore plus complexes, vous pourrez utiliser les instructions de croisement de sélections.

4DBKQuerySet positionne la sélection sur la première fiche (comme si un 4DBKRecordSet/1 avait été exécuté).

Une fois qu'une instruction 4DBKQuerySet a été exécutée, la sélection des fiches trouvées est stockée dans la session courante de l'utilisateur. Il sera donc possible de changer de sélection et même de boutique puis de revenir à la sélection initiale pour la retrouver immédiatement dans l'état où elle avait été laissée.

Par défaut, 4DBKQuerySet ne retourne pas de valeur. Il est toutefois possible d'obtenir le nombre d'articles placés dans la sélection en préfixant la requête du caractère * (voir exemple 2).

La liste des champs sur lesquels il est possible d'effectuer une recherche est fournie dans le chapitre 19, "Champs", page 177.

Conditions

Il faut avoir fixé la boutique courante à l'aide de l'instruction 4DBKStoreSet et la sélection courante à l'aide de l'instruction 4DBKSelectionSet.

Exemples

(1) L'exemple suivant effectue une recherche en fonction de deux critères, puis affiche la valeur d'un champ de la première fiche de la sélection issue de la recherche :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1
<!--#4DBKQuerySet/Type=ordinateur port@ and Fabricant=apple--> //effectuer la recherche
<!--#4DBKRecordSet/1--> //aller à la 1re fiche
<!--#4DBKField/Produit--> //afficher IBOOK MANDARINE en place de la balise
```

(2) Cet exemple illustre la possiblité de récupérer le nombre de fiches placées dans la sélection :

```
<!--#4DBKQuerySet/L01=Mobile--> remplit la sélection courante avec les 12 articles dont la famille 1 est "Mobile"
<!--#4DBKQuerySet/*L01=Mobile--> remplit la sélection courante avec les 12 articles dont la famille 1 est "Mobile" et retourne 12
```

4DBKQueryInSelectionSet

<!--#4DBKQueryInSelectionSet/requête-->

Paramètre	Description	Valeur(s)
requête	Chaîne de recherche	• ALL • Champ Opérateur Valeur [[and/or] Champ Opérateur Valeur]

Objectif : Effectuer une recherche parmi la sélection courante et affecter l'ensemble de fiches trouvées à la sélection courante.

Description

La commande 4DBKQueryInSelectionSet effectue une recherche parmi les fiches de la sélection courante sur la base de la formule définie dans le paramètre *requête* et remplace la sélection courante par l'ensemble de fiches trouvées. La sélection courante est définie par la commande 4DBKSelectionSet.

Cette commande est utile dans le cadre de recherches progressives au cours desquelles différents critères sont appliqués successivement à la sélection courante.

Reportez-vous à la définition de la commande 4DBKQuerySet pour une description détaillée du paramètre *requête* (son fonctionnement est identique pour les deux commandes).

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de l'instruction 4DBKStoreSet et la sélection courante à l'aide de l'instruction 4DBKSelectionSet.

Exemple

L'exemple suivant effectue deux recherches successives dans la même sélection :

```
<!---#4DBKSelectionSet/1-->
<!---#4DBKQuerySet/color=noir-->
<!---#4DBKQueryInSelectionSet/label=organis@ & weight<=3000-->
```

4DBKSortSet

<!--#4DBKSortSet/critère[> ou <][,critère2[> ou <],…,critèreN[> ou <]]-->

Paramètre	Description	Valeur(s)
critère	Critère de tri	• Nom ou code de champ valide pour le tri • TS : ordre d'ajout dans la sélection
> ou <	Sens du tri	• > : tri croissant • < : tri décroissant Si omis, tri croissant (par défaut)

Objectif : Assigner un ou plusieurs critère(s) de tri à la sélection courante.

Description

Cette commande permet de définir le ou les *critère*(s) de tri de la sélection courante, qui seront appliqués par la commande 4DBKSelectionSort. Pour que le tri soit effectif, il est nécessaire d'appeler la commande 4DBKSelectionSort.

Vous pouvez trier la sélection en fonction des valeurs d'un ou plusieurs champ(s) ou en fonction de l'ordre chronologique d'ajout des éléments dans cette sélection.

- Pour utiliser les valeurs d'un champ, passez son nom ou son code dans le paramètre *critère*. La liste des champs sur lesquels il est possible d'effectuer un tri est fournie dans le chapitre 19, "Champs", page 177.
- Pour utiliser l'ordre d'ajout dans la sélection, passez le code TS (pour *TimeStamp*) dans le paramètre *critère*. Ce mécanisme permet notamment de trier le panier par ordre d'ajout.

Si aucun sens de tri n'est précisé, le tri sera croissant par défaut.

Le critère de tri défini par cette commande est conservé en mémoire durant toute la session Web. L'appel à l'instruction 4DBKSelectionSort peut donc être effectué dans une autre page.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de l'instruction 4DBKStoreSet et la sélection courante à l'aide de l'instruction 4DBKSelectionSet.

Exemples

(1) L'exemple suivant définit le critère de tri d'une sélection issue d'une recherche (la sélection n'est pas triée à ce stade) :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1
<!--#4DBKQuerySet/Type=ordinateur port@ & Fabricant=apple--> //effectuer une recherche
<!--#4DBKSortSet/categorie>,produit>--> //fixer le critère de tri de la sélection numéro 1 sur les champs catégorie puis produit de façon croissante
```

(2) Cet exemple définit comme critère de tri du panier l'ordre d'ajout des éléments. Le dernier article ajouté au panier sera le premier élément de la sélection triée :

```
<!--#4DBKStoreSet/MCOM--> //Boutique MCOM
<!--#4DBKSelection/B--> // Sélection du panier
<!--#4DBKSortSet/TS< --> // fixer le critère de tri du panier sur l'ordre
        décroissant d'ajout des éléments dans le panier
<!--#4DBKSort--> // Tri effectif du panier
```

Référence : 4DBKSort, 4DBKSelectionSort

4DBKSelectionSort

<!--#4DBKSelectionSort-->
Cette commande ne requiert pas de paramètre.

Objectif : Trier la sélection courante avec le(s) critère(s) défini(s) par la commande 4DBKSortSet.

Description

La commande 4DBKSelectionSort trie la sélection courante en utilisant le ou les critère(s) de tri défini(s) préalablement par la commande 4DBKSortSet.

Après le tri d'une sélection, le passage sur une autre sélection puis le retour à la première, deux cas peuvent se présenter, en fonction de la valeur courante du paramètre *Autosort* de la commande 4DBKPrefsSet :

- si le paramètre *Autosort* est fixé à "no" (par défaut), il sera nécessaire de rappeler la commande 4DBKSelectionSort pour retrier la sélection.
- si le paramètre *Autosort* est fixé à "yes", la sélection sera retriée automatiquement à chaque nouveau 4DBKSelectionSet, selon les critères définis par 4DBKSortSet.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet et la sélection courante à l'aide de la commande 4DBKSelectionSet. Le(s) critère(s) de tri doivent avoir été défini(s) à l'aide de la commande 4DBKSortSet.

Exemple

L'exemple suivant effectue un tri sur la sélection numéro 1. Le critère de tri courant doit avoir été défini auparavant a l'aide de la commande 4DBKSortSet :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1
<!--#4DBKSelectionSort--> //trier la sélection numéro 1
```

Référence : 4DBKSort, 4DBKSortSet

4DBKSort

<!--#4DBKSort-->
Cette commande ne requiert pas de paramètre.

Objectif : Récupérer le(s) critère(s) de tri courant(s) de la sélection courante.

Description

La commande 4DBKSort retourne le ou les critère(s) de tri courant(s) associé(s) à la sélection courante.

A l'aide de cette commande, il est possible de construire par exemple un menu HTML des critères de tri qui permettra à l'utilisateur de choisir son propre tri pour visualiser une liste.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet et la sélection courante à l'aide de la commande 4DBKSelectionSet.

Exemple

L'exemple suivant définit le critère de tri d'une sélection et affiche le champ utilisé comme critère de tri :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1
<!--#4DBKSortSet/categorie>--> //fixer le critère de tri de la sélection numé-
            ro 1 sur le champ catégorie (code L02)
<!--#4DBKSort--> //retourne L02>
```

Référence : 4DBKSelectionSort, 4DBKSortSet

4DBKSelectionIntersect

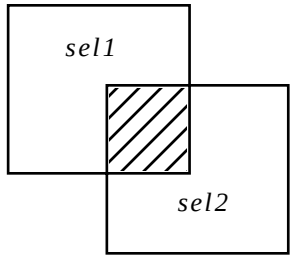
<!--#4DBKSelectionIntersect/sel1,sel2,seldst-->

Paramètre	Description	Valeur(s)
sel1	1re sélection à croiser	• 1 à 9 : utiliser la sélection 1 à 9 de la boutique courante, OU • B : utiliser le panier de la boutique courante
sel2	2e sélection à croiser	
seldst	Sélection réceptrice du résultat de l'intersection	

Objectif : Croiser deux sélections et placer le résultat dans *seldst*.

Description

Cette commande retourne le résultat de l'intersection de deux sélections (*sel1* et *sel2*) dans la sélection *seldst*.



Chaque paramètre "sélection" peut recevoir l'une des valeurs suivantes :

- 1 à 9 : utiliser la sélection numéro 1 à numéro 9 de la boutique courante.
- B : utiliser le panier de la boutique courante.

Cette commande est utilisée pour effectuer des recherches élaborées.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemple

L'exemple suivant effectue deux recherches parmi deux sélections différentes et place les éléments communs dans la première sélection :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1
<!--#4DBKQuerySet/Type=ordinateur port@--> //effectuer la recherche
<!--#4DBKSelectionSet/2--> //passer sur la sélection numéro 2
<!--#4DBKQuerySet/Fabricant=ibm or Fabricant=apple--> //effectuer la re-
        cherche
<!--#4DBKSelectionIntersect/1,2,1--> //intersection, la sélection 1 contient
        les ordinateurs portables Apple et Ibm
```

Référence : 4DBKSelectionUnion, 4DBKSelectionDifference, 4DBKSelectionCopy.

4DBKSelectionUnion

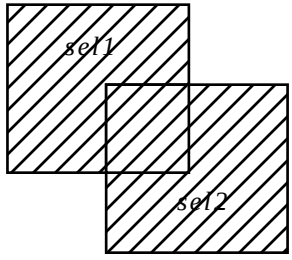
<!--#4DBKSelectionUnion/sel1,sel2,seldst-->

Paramètre	Description	Valeur(s)
sel1	1re sélection à réunir	• 1 à 9 : utiliser la sélection 1 à 9 de la boutique courante, OU • B : utiliser le panier de la boutique courante
sel2	2e sélection à réunir	
seldst	Sélection réceptrice du résultat de l'union	

Objectif : Unir deux sélections et placer le résultat dans *seldst*.

Description

Cette commande retourne le résultat de l'union de deux sélections (*sel1* et *sel2*) dans la sélection *seldst*.



Chaque paramètre “sélection” peut recevoir l’une des valeurs suivantes :

- 1 à 9 : utiliser la sélection numéro 1 à numéro 9 de la boutique courante.
- B : utiliser le panier de la boutique courante.

Cette commande est utilisée pour effectuer des recherches élaborées.

Conditions

Il faut avoir préalablement fixé la boutique courante à l’aide de la commande 4DBKStoreSet.

Exemple

Cet exemple effectue trois recherches successives et traite les résultats à l’aide d’une intersection puis d’une union :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1
<!--#4DBKQuerySet/Type=ordinateur port@--> //effectuer une recherche
<!--#4DBKSelectionSet/2--> //passer sur la sélection numéro 2
<!--#4DBKQuerySet/Fabricant=ibm or Fabricant=apple--> //effectuer une
        2e recherche
<!--#4DBKSelectionIntersect/1,2,1--> //intersection, la sélection 1 contient
        les ordinateurs portables Apple ou Ibm
<!--#4DBKQuerySet/Type=Accessoires and Usb=1--> //effectuer une 3e re-
        cherche dans la sélection numéro 2
<!--#4DBKSelectionUnion/1,2,1--> //réunion, la sélection 1 contient les or-
        dinateurs portables Apple et Ibm et les accessoires USB.
```

Référence : 4DBKSelectionIntersect, 4DBKSelectionDifference, 4DBKSelectionCopy.

4DBKSelectionDifference

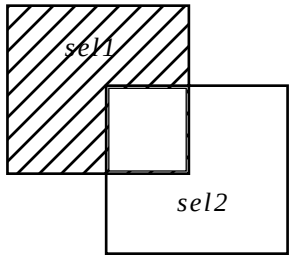
<!--#4DBKSelectionDifference/sel1,sel2,seldst-->

Paramètre	Description	Valeur(s)
sel1	Sélection dont les articles doivent être enlevés	• 1 à 9 : utiliser la sélection 1 à 9 de la boutique courante, OU • B : utiliser le panier de la boutique courante
sel2	Sélection contenant les articles à enlever de *sel1*	
seldst	Sélection réceptrice du résultat de la différence	

Objectif : Soustraire de *sel1* les articles présents dans *sel2*.

Description

Cette commande soustrait de la sélection *sel1* les articles présents dans la sélection *sel2* et place le résultat dans *seldst*.



Chaque paramètre “sélection” peut recevoir l’une des valeurs suivantes :

- 1 à 9 : utiliser la sélection numéro 1 à numéro 9 de la boutique courante.
- B : utiliser le panier de la boutique courante.

Cette commande est utilisée pour effectuer des recherches élaborées.

Conditions

Il faut avoir préalablement fixé la boutique courante à l’aide de la commande 4DBKStoreSet.

Exemple

L'exemple suivant effectue deux recherches parmi deux sélections différentes et exclut de la première sélection les éléments présents dans le seconde.

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1
<!--#4DBKQuerySet/Type=ordinateur port@--> //effectuer une recherche
<!--#4DBKSelectionSet/2--> //passer sur la sélection numéro 2
<!--#4DBKQuerySet/Fabricant=apple--> //effectuer une recherche
<!--#4DBKSelectionDifference/1,2,1--> //différence, la sélection 1 contient
        les ordinateurs portables sauf les ordinateurs Apple
```

Référence : 4DBKSelectionIntersect, 4DBKSelectionUnion, 4DBKSelectionCopy.

4DBKSelectionCopy

<!--#4DBKSelectionCopy/selsrc,seldst-->

Paramètre	Description	Valeur(s)
selsrc	Sélection source à copier	• 1 à 9 : utiliser la sélection 1 à 9 de la boutique courante, OU • B : utiliser le panier de la boutique courante
seldst	Sélection de destination	

Objectif : Copier le contenu de la sélection *selsrc* dans la sélection *seldst*.

Description

La commande 4DBKSelectionCopy copie le contenu d'une sélection source (*selsrc*) dans une autre sélection (*seldst*). Le contenu précédent de la sélection *seldst* est entièrement remplacé.

Cette commande permet de conserver intacte une première sélection et de la dupliquer dans d'autres sélections afin d'effectuer des recherches élaborées.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemple

Cet exemple copie la sélection numéro 1 dans la sélection numéro 2 :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1
<!--#4DBKQuerySet/Type=ordinateur port@--> //effectuer une recherche
<!--#4DBKQSelectionCopy/1,2--> //la sélection numéro 2 est maintenant
        identique à la numéro 1
```

Référence : 4DBKSelectionIntersect, 4DBKSelectionUnion, 4DBKSelectionDifference.

4DBKSelectionSize

<!--#4DBKSelectionSize-->
Cette commande ne requiert pas de paramètre.

Objectif : Retourner le nombre de fiches de la sélection courante.

Description

La commande 4DBKSelectionSize retourne le nombre de fiches présentes dans la sélection courante. Elle peut être utilisée dans les boucles ou pour afficher le nombre d'articles trouvés par une recherche.

4DBKSelectionSize peut être utilisée conjointement à la commande 4DBKVarSet pour assigner des valeurs à des variables.

Il est également possible de l'employer au sein de liens URLs. Dans ce cas, si vous souhaitez que la valeur soit retournée au moment du chargement de la page et non au moment où l'utilisateur clique sur le lien, utilisez l'instruction !4DBKSelectionSize.

> *Note :* Vous disposez également de 10 commandes-raccourcis permettant d'obtenir le nombre de fiches pour chaque sélection (1 à 9 et B) sans avoir besoin de coder les instructions (à l'aide de 4DBKSelectionSet et 4DBKSelectionSize). Ces commandes ne modifient pas la sélection courante.
> Ces commandes sont 4DBKSelectionSize1 à 4DBKSelectionSize9 et 4DBKSelectionSizeB.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet et la sélection courante à l'aide de la commande 4DBKSelectionSet.

Exemple

L'exemple suivant effectue une recherche dans la sélection numéro 1 d'une boutique et retourne le nombre d'éléments trouvés :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/3--> //passer sur la sélection numéro 3
<!--#4DBKQuerySet/Type=ordinateur de bureau--> //effectuer la recherche
<!--#4DBKSelectionSize--> //retourne par exemple 12
```

Référence : 4DBKSelectionSizeSet, 4DBKVarSet.

4DBKSelectionSizeSet

<!--#4DBKSelectionSizeSet/nombre-->

Paramètre	Description	Valeur(s)
nombre	Nombre de fiches de la sélection courante à conserver	Numérique

Objectif : Réduire le nombre de fiches de la sélection courante.

Description

La commande 4DBKSelectionSize conserve *nombre* fiches dans la sélection courante. Elle est généralement utilisée pour limiter dans les listes le nombre de fiches trouvées par une recherche.

Il est également possible d'appeler cette commande pour remettre le panier à zéro, en utilisant la sélection B et en passant 0 dans le paramètre *nombre*.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet et la sélection courante à l'aide de la commande 4DBKSelectionSet.

Exemple

L'exemple suivant réduit à quatre fiches la sélection issue d'une recherche :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/3--> //passer sur la sélection numéro 2
<!--#4DBKQuerySet/Type=ordinateur de bureau--> //effectuer une recherche
<!--#4DBKSelectionSize--> //retourne 12 par exemple
<!--#4DBKSelectionSizeSet/4--> //réduire la sélection à 4 fiches
<!--#4DBKSelectionSize--> //retourne 4
```

Référence : 4DBKSelectionSize.

4DBKSelectionDistinct

<!--#4DBKSelectionDistinct/champ-->

Paramètre	Description	Valeur(s)
champ	Champ valide en sélection distincte (voir chapitre Champs)	Nom ou code de champ

Objectif : Filtrer une sélection sur les valeurs distinctes.

Description

La commande 4DBKSelectionDistinct filtre la sélection courante de manière à ne conserver que les fiches pour lesquelles le *champ* contient une valeur différente des autres fiches (valeurs distinctes).

Cette commande permet de générer des listes de valeurs connexes aux articles de la sélection courante (par exemple les familles).

La liste des champs parmi lesquels il est possible de filtrer les valeurs distinctes est fournie dans le chapitre 19, "Champs", page 177.
Il est possible d'utiliser le nom d'un champ tel que défini dans le masque principal. S'il y a plusieurs masques pour la boutique, 4D Business Kit utilisera le masque nommé Standard ou Std. Si aucun de ces deux noms de masques n'est défini, le premier masque créé sera utilisé.

Cette commande est très utile pour filtrer une sélection en fonction d'une famille. Si vous voulez obtenir la liste des catégories (la catégorie est dans ce cas définie comme famille numéro 1), vous rechercherez tous les articles qui ont une valeur de catégorie différente. Ensuite il suffira de "boucler" sur ces articles pour afficher la liste dans un menu par exemple.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet et la sélection courante à l'aide de la commande 4DBKSelectionSet.

Exemple

L'exemple suivant réduit la sélection numéro 1 (contenant tous les articles de la boutique) aux fiches comportant des valeurs différentes dans le champ "categorie'' :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1
<!--#4DBKQuerySet/all--> //effectuer une recherche, ici tous les articles
<!--#4DBKSelectionDistinct/categorie--> //réduit la sélection numéro 1 aux
        articles ayant chacun une catégorie différente
```

6 Fiches

4DBKCursorSet

<!--#4DBKCursorSet/valeur-->

Paramètre	Description	Valeur(s)
valeur	Numéro de fiche vers laquelle le curseur doit pointer	Numérique

Objectif : Fixer la valeur du curseur dans la sélection courante.

Description

La commande 4DBKCursorSet définit la position du curseur dans la sélection courante. Elle est utile lors de l'affichage de listes ou de boutons de début/fin de sélection.

Chaque sélection possède un **curseur**. Le curseur est une valeur qui indique la première fiche affichée sur une page. Il vous permet de savoir où vous vous situez dans la sélection en cours d'affichage (par exemple, lors de l'affichage d'une page de catalogue). A l'aide du curseur, vous pouvez créer des boutons de page suivante ou précédente, des objets d'accès direct aux pages, etc.

Par défaut, tous les curseurs prennent la valeur 1 à l'initialisation d'une session Web. Le curseur est ensuite mémorisé dans la session de l'internaute et de ce fait est "connu" dans toutes les pages Web.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet et la sélection courante à l'aide de la commande 4DBKSelectionSet.

Exemple

L'exemple suivant modifie la valeur courante du curseur de la sélection numéro 1 :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1
<!--#4DBKCursor--> //retourne 3 (par exemple) en place de la balise
<!--#4DBKCursorSet/5--> //fixer le curseur de la sélection numéro 1 à la 5e
        fiche
<!--#4DBKCursor--> //retourne 5 en place de la balise
```

Référence : 4DBKCursor, 4DBKCursorNext.

4DBKCursor

<!--#4DBKCursor-->
Cette commande ne requiert pas de paramètre.

Objectif : Retourne la valeur courante du curseur.

Note : Pour plus d'informations sur les curseurs, reportez-vous à la description de la commande 4DBKCursorSet.

Description

La commande 4DBKCursor retourne la valeur courante du curseur de la sélection courante.

Cette commande est utile lors de l'affichage de listes ou de boutons de début/fin de sélection. La valeur du curseur permet d'effectuer des tests afin de savoir si l'internaute se trouve sur la première ou la dernière page de son catalogue, ou sur la première ou la dernière fiche.

Dans un lien, si vous souhaitez que la valeur soit retournée au moment du chargement de la page et non au moment où l'internaute clique sur le lien, utilisez l'instruction !4DBKCursor.

Note : Vous disposez de 10 commandes-raccourcis permettant d'obtenir la position courante du curseur dans chaque sélection (1 à 9 et B) sans avoir besoin de coder les instructions (à l'aide de 4DBKSelectionSet et 4DBKCursor). Il est à noter que ces commandes ne modifient pas la sélection courante.
Ces commandes sont 4DBKCursor1 à 4DBKCursor9 et 4DBKCursorB.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet et la sélection courante à l'aide de la commande 4DBKSelectionSet.

Exemple

Reportez-vous à l'exemple de la commande 4DBKCursorSet.

Référence : 4DBKCursorSet, 4DBKCursorNext.

4DBKCursorNext

<!--#4DBKCursorNext/valeur-->

Paramètre	Description	Valeur(s)
valeur	Nombre de fiches	Numérique (positif ou négatif)

Objectif : Déplacer le curseur de la sélection courante.

Note : Pour plus d'informations sur les curseurs, reportez-vous à la description de la commande 4DBKCursorSet.

Description

La commande 4DBKCursorNext fait avancer ou reculer le curseur de la sélection courante de *valeur* fiches(s). Passez une *valeur* positive pour faire avancer le curseur et une *valeur* négative pour le faire reculer.

Cette commande est utile pour gérer des boutons de navigation et de déplacement dans les pages d'un catalogue ou même au sein des fiches pour passer à la suivante ou à la précédente.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet et la sélection courante à l'aide de la commande 4DBKSelectionSet.

Exemple

L'exemple suivant illustre le fonctionnement de gestion du curseur :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1
<!--#4DBKCursor--> //retourne 3 (par exemple) en place de la balise
<!--#4DBKCursorNext/2--> //avancer le curseur de 2 fiches
<!--#4DBKCursor--> //retourne 5 en place de la balise
<!--#4DBKCursorNext/-4--> //reculer le curseur de 4 fiches
<!--#4DBKCursor--> //retourne 1 en place de la balise
```

Référence : 4DBKCursorSet, 4DBKCursor.

4DBKRecord

<!--#4DBKRecord-->
Cette commande ne requiert pas de paramètre.

Objectif : Retourner le numéro de la fiche courante.

Description

La commande 4DBKRecord retourne le numéro de la fiche courante de la boutique et de la sélection courantes.

Cette commande est utile dans le cadre de l'affichage de listes. Le numéro de la fiche courante devra être inscrit dans le lien permettant d'afficher la page détail d'un article.
La commande permet également de gérer des boutons de navigation pour passer à l'article précédent ou suivant à partir de la page détail.

Dans un lien, si vous souhaitez que la valeur soit retournée au moment du chargement de la page et non au moment où l'utilisateur clique sur le lien, utilisez l'instruction !4DBKRecord.

> *Note :* Vous disposez de 10 commandes-raccourcis permettant d'obtenir le numéro de la fiche courante pour chaque sélection (1 à 9 et B) sans avoir besoin de coder les instructions (à l'aide de 4DBKSelectionSet et 4DBKRecord). Il est à noter que ces commandes ne modifient pas la sélection courante.
> Ces commandes sont 4DBKRecord1 à 4DBKRecord9 et 4DBKRecordB.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet et la sélection courante à l'aide de la commande 4DBKSelectionSet.

Exemple

L'exemple suivant retourne le numéro de la fiche correspondant à la position du curseur :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1
<!--#4DBKCursorSet/5--> //fixer le curseur de la sélection numéro 1 à la 5e
        fiche
<!--#4DBKRecordSet/C--> //aller à la 5e fiche
<!--#4DBKRecord--> //retourne 5 en place de la balise
```

Référence : 4DBKRecordRandom, 4DBKRecordSet, 4DBKRecordNext.

4DBKRecordRandom

<!--#4DBKRecordRandom-->
Cette commande ne requiert pas de paramètre.

Objectif : Retourner un numéro de fiche aléatoire dans la sélection.

Description

La commande 4DBKRecordRandom retourne un numéro de fiche aléatoire parmi la sélection courante.

Cette commande permet d'afficher des produits au hasard choisis parmi une sélection. Si vous appelez cette commande à plusieurs reprises, 4D Business Kit garantit que le numéro de fiche retourné n'est pas déjà "sorti" lors des appels précédents.

Dans un lien, si vous souhaitez que la valeur soit retournée au moment du chargement de la page et non au moment où l'utilisateur clique sur le lien, utilisez l'instruction !4DBKRecordRandom.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet et la sélection courante à l'aide de la commande 4DBKSelectionSet.

Exemple

L'exemple suivant utilise la commande 4DBKRandom pour définir de façon aléatoire la fiche courante :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1
<!--#4DBKQuerySet/type="imac"--> //la sélection contient maintenant 6 fiches (par exemple)
<!--#4DBKRecordRandom--> //retourne 4 (par exemple)
<!--#4DBKRecordRandom--> //retourne 1 (par exemple)
<!--#4DBKRecordRandom--> //retourne 6 (par exemple)
<!--#4DBKRecordSet/4DBKRecordRandom--> //aller à la 3e fiche (par exemple)
```

Référence : 4DBKRecord, 4DBKRecordSet, 4DBKRecordNext.

4DBKRecordSet

<!--#4DBKRecordSet/numFiche-->
<!--#4DBKRecordSet/C-->

Paramètre	Description	Valeur(s)
numFiche	Numéro de la fiche courante	De 1 à la taille de la sélection
C	Numéro de la fiche pointée par le curseur de la sélection courante	C

Objectif : Définir la fiche courante dans la sélection courante.

Description

La commande 4DBKRecordSet désigne une fiche de la sélection courante comme nouvelle fiche courante.

Lors de l'affichage de listes, cette commande est utile pour se positionner dans la sélection au sein d'une page.

Si vous employez la valeur *C*, vous positionnez la sélection sur la fiche pointée par le curseur de la sélection. Ce fonctionnement vous permet d'obtenir toujours la même première fiche dans une page de catalogue.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet et la sélection courante à l'aide de la commande 4DBKSelectionSet.

Exemple

L'exemple suivant définit la fiche courante à l'aide d'une valeur donnée puis à l'aide du curseur :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1
<!--#4DBKRecordSet/3--> //aller à la 3e fiche
<!--#4DBKCursorSet/5--> //fixer le curseur de la sélection numéro 1 sur la 5e
        fiche
<!--#4DBKRecordSet/C--> //aller à la 5e fiche
```

Référence : 4DBKRecord, 4DBKRecordRandom, 4DBKRecordNext.

4DBKRecordNext

<!--#4DBKRecordNext[/valeur]-->

Paramètre	Description	Valeur(s)
valeur	Nombre de fiches	Numérique (positif ou négatif) Si omis : 1

Objectif : Déplacer le pointeur de fiche courante dans la sélection courante.

Description

La commande 4DBKRecordNext déplace le pointeur de fiche courante de *valeur* fiche(s) dans la sélection courante.

Si le paramètre *valeur* est positif, le pointeur avance dans la sélection, s'il est négatif, le pointeur recule.
Si le paramètre est omis, 4D Business Kit utilise la valeur 1 et avance le pointeur d'une fiche.

Lors de l'affichage de listes, cette commande est utile pour afficher les différentes fiches au sein d'une page de catalogue.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet et la sélection courante à l'aide de la commande 4DBKSelectionSet.

Exemple

L'exemple suivant permet d'afficher la valeur d'un champ de plusieurs fiches désignées à l'aide de la commande 4DBKRecordNext :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1
<!--#4DBKRecordSet/C--> //aller à la 1re fiche (le curseur est ici à 1)
<!--#4DBKField/T01--> //afficher "Article 1" en place de la balise
<!--#4DBKRecordNext--> //avancer le curseur d'une fiche
<!--#4DBKRecord--> //retourne 2 en place de la balise
<!--#4DBKField/T01--> //afficher "Article 2" en place de la balise
<!--#4DBKRecordNext/3--> //avancer le curseur de 3 fiches
<!--#4DBKField/T01--> //affiche "Article 5" en place de la balise
<!--#4DBKRecordNext/-2--> //reculer le curseur de 2 fiches
<!--#4DBKRecord--> //retourne 3 en place de la balise
```

Référence : 4DBKRecord, 4DBKRecordRandom, 4DBKRecordSet.

4DBKRecordExists

<!--#4DBKRecordExists-->
Cette commande ne requiert pas de paramètre.

Objectif : Tester l'existence d'une fiche courante.

Description

La commande 4DBKRecordExist permet de vérifier si une fiche courante existe dans la sélection courante. La commande retourne True si une fiche courante existe, sinon elle retourne False.

Il est possible, à l'aide de cette commande, de décider d'afficher ou non, au sein d'un catalogue, le code HTML d'une fiche selon qu'elle existe ou non.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet et la sélection courante à l'aide de la commande 4DBKSelectionSet.

Exemple

L'exemple suivant affiche des valeurs des champs de la 1re fiche de la sélection numéro 1 si elle existe, c'est-à-dire si la sélection numéro 1 n'est pas vide.

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1
<!--#4DBKRecordSet/1--> //aller à la 1re fiche
<!--#4DBKIf(4DBKRecordExists)--> //si la fiche existe
<!--#4DBKField/Nom--> * <!--#4DBKField/Prénom--> //retourner les champs
        Nom & Prénom en place des balises
<!--#4DBKEndIf-->
```

4DBKDetailSet

<!--#4DBKDetailSet/data-->

Paramètre	Description	Valeur(s)
data	Données correspondant à la fiche souhaitée	Données

Objectif : Définir la fiche pour laquelle on souhaite la page détail.

Description

La commande 4DBKDetailSet désigne la fiche de laquelle on souhaite afficher la page détail. Cette commande s'emploie dans le cadre de l'affichage de listes et de pages détail, en conjonction avec la commande 4DBKDetail.

4DBKDetailSet stocke l'information que vous passez dans le paramètre *data* et que vous pourrez rappeler ultérieurement au sein de la fiche détail à l'aide de l'instruction 4DBKDetail. En arrivant dans la fiche détail, il est conseillé d'effectuer une recherche en utilisant 4DBKQuerySet et 4DBKDetail afin de s'assurer d'être positionné sur la bonne fiche.
Dans un catalogue, vous utilisez 4DBKDetailSet dans un lien qui vous permettra de passer à la fiche détail.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemple

L'exemple suivant stocke dans 4DBKDetail la valeur du champ "reference" de la 3e fiche :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1
<!--#4DBKRecordSet/3--> //aller à la 3e fiche
<!--#4DBKDetailSet/!4DBKField/reference--> //fixer le détail à la valeur du
        champ C01
```

Référence : 4DBKDetail.

4DBKDetail

<!--#4DBKDetail-->
Cette commande ne requiert pas de paramètre.

Objectif : Retourner la fiche pour laquelle on souhaite le détail.

Description

La commande 4DBKDetail récupère l'information que vous lui avez passée via la commande 4DBKDetailSet. Cette commande s'emploie dans le cadre de l'affichage de listes et de pages détail, en conjonction avec la commande 4DBKDetailSet.

Dans une page détail, vous pouvez rechercher la fiche correspondant à cette page puis afficher son contenu.

Dans un lien, si vous souhaitez que la valeur soit retournée au moment du chargement de la page et non au moment où l'utilisateur clique sur le lien, utilisez l'instruction !4DBKDetail.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemple

Cet exemple recherche la fiche dont le champ “référence” est égal à la valeur préalablement stockée dans 4DBKDetail à l’aide de la commande 4DBKDetailSet. La fiche trouvée devient la fiche courante ; ses champs “Nom” et “Prénom” sont alors affichés.

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1
<!--#4DBKQuerySet/reference=4DBKDetail--> //effectuer la recherche
<!--#4DBKRecordSet/1--> //aller à la 1re fiche
<!--#4DBKField/Produit--> * <!--#4DBKField/Marque--> //retourne les
        champs Nom & Prénom en place des balises
```

Référence : 4DBKDetailSet.

4DBKField

<!--#4DBKField/champ[.devise][,format]-->

Paramètre	Description	Valeur(s)
champ	Champ valide en affichage (voir chapitre Champs)	Nom ou code de champ
devise	Code devise	3 caractères
format	Format d'affichage valide	Voir chapitre Formats d’affichage et fonctions

Objectif : Retourner la valeur courante d’un champ donné.

Description

La commande 4DBKField retourne la valeur courante du champ désigné par le paramètre *champ*. Cette commande peut être utilisée pour afficher ou tester la valeur d’un champ de données.

La liste des champs qu’il est possible d’afficher est fournie dans le chapitre 19, “Champs” page 177.

Si la *devise* n’est pas spécifiée, les données de type prix seront affichées telles qu’elles ont été saisies dans la boutique.

Dans un lien, si vous souhaitez que la valeur soit retournée au moment du chargement de la page et non au moment où l’utilisateur clique sur le lien, utilisez l’instruction !4DBKField.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet et la sélection courante à l'aide de la commande 4DBKSelectionSet.
En outre, un masque courant doit être défini.

Exemples

(1) L'exemple suivant permet d'afficher différents types de champs. Des formats d'affichage sont appliqués aux deux derniers champs :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1
<!--#4DBKField/T01--> //retourner le champ T01 en place de la balise
<!--#4DBKField/Categorie/U--> //retourner le champ correspondant au
        champ Categorie en majuscules
<!--#4DBKField/PrSaleEXV.FRF,### ##0,00--> //retourner le prix en FRF
```

(2) Cet exemple affiche un prix dans la devise définie à l'aide d'une variable. La valeur de la variable est fixée à l'aide d'un lien situé dans une autre page.
La première page contient les liens suivants :

```
<a href="4DBKExecute:4DBKStoreSet/DEM1;4DBKVarSet/VT01=EUR;4DBK
        Go/mapage.htm">Euros</A>
<a href="4DBKExecute:4DBKStoreSet/DEM1;4DBKVarSet/VT01=USD;4DBK
        Go/mapage.htm">Dollars</A>
```

La page "mapage.htm" contient le code suivant :

```
<!--#4DBKField/PrItemEXV.4DBKVar/VT01,### ##0,00> <!--4DBKVar/VT01-
        -> // affiche par exemple "459,99 EUR"
```

Référence : 4DBKFieldSet.

4DBKFieldSet

<!--#4DBKFieldSet/champ=valeur OU =autreFonction-->

Paramètre	Description	Valeur(s)
champ	Champ valide en affichage et en modification (voir chapitre Champs)	Nom ou code de champ
=valeur	Nouvelle valeur du champ	Valeur dont le type correspond au type de champ
=autreFonction	Fonction 4D Business Kit	Toute fonction 4D Business Kit retournant une valeur

Objectif : Modifier et enregistrer la valeur d'un champ.

Description

La commande 4DBKFieldSet permet de modifier et de stocker la valeur du *champ*. La valeur peut avoir été modifiée par exemple à l'aide d'un formulaire HTML.

Cette commande n'est utilisable actuellement que pour créer ou modifier les données d'un compte client.

La liste des champs des comptes clients qu'il est possible d'afficher et de modifier est fournie dans le chapitre 19, "Champs" page 177.

A - Création d'un nouveau compte client

- Il n'est pas possible de créer un nouveau compte client si une session identifiée existe sur le poste local. Si c'est le cas, il est nécessaire de la refermer en utilisant la commande 4DBKCustomerLogout. Il est alors possible d'exécuter des instructions 4DBKFieldSet pour la création d'un nouveau compte client.
- La toute première commande 4DBKFieldSet doit concerner le code client : 4DBKFieldSet/CusCode=code_client_saisi
- Si *code_client_saisi* n'existe pas dans la table des clients, tous les champs spécifiés par 4DBKFieldSet sont sauvegardés et le client est identifié sous ce code.

- Si *code_client_saisi* existe déjà dans la table des clients, l'utilisateur se verra alors redirigé vers une page d'erreur par défaut, ou la page fixée par les instructions suivantes :

  <!--#4DBKPrefsSet/CusErrorMode=PAGE-->
  <!--#4DBKPrefsSet/CusErrorDuplicateAdd=page_erreur.htm-->

  L'utilisateur se verra alors notifier l'erreur ainsi que le code client temporaire qui lui été attribué par défaut. Le client est identifié avec ce code temporaire et lorsqu'il revient sur le formulaire de données client, il n'a plus qu'à fournir un nouveau code client. Les autres champs spécifiés par 4DBKFieldSet ont été bien entendu stockés.

B - Modification d'un compte client existant

- Pour que la commande puisse modifier un compte client, un client doit avoir été identifié préalablement via 4DBKCustomerLogin.
- Si le développeur souhaite laisser au client la possibilité de changer son code, la toute première commande 4DBKFieldSet doit concerner le code client : 4DBKFieldSet/CusCode=code_client_saisi
- Dans ce cas, si *code_client_saisi* n'est pas déjà attribué à un autre client, tous les champs spécifiés par 4DBKFieldSet sont mis à jour.
- Si *code_client_saisi* est déjà attribué à un autre client, l'utilisateur se verra redirigé vers une page d'erreur par défaut, ou la page fixée par les instructions suivantes :

  <!--#4DBKPrefsSet/CusErrorMode=PAGE-->
  <!--#4DBKPrefsSet/CusErrorDuplicateUpd=page_erreur.htm-->

  L'utilisateur se voit alors notifier l'erreur. Lorsqu'il revient sur le formulaire de données client, son code n'a pas changé mais les autres champs spécifiés par 4DBKFieldSet ont été mis à jour.

Scénarios pour la gestion des formulaires du compte client

Lorsque le développeur veut définir des pages Web qui serviront d'interface pour le stockage des comptes clients, trois possibilités s'offrent à lui :

1. *Un formulaire validé par un Http Post sur 4DBKCustomerForm*

- *Avantages* :
  - L'appel aux commandes 4DBKFieldSet est automatique.
  - Très peu de développement.
  - Une seule page Web.
- *Contraintes* :
  - Aucun contrôle n'est possible au niveau du stockage.
  - Les champs du formulaire doivent porter le même nom que ceux du compte client (par exemple *CusEmail* pour l'adresse E-mail).
- *Exemple* :
  - Voir la page *account_post.htm* dans le site TEST.

2. *Un formulaire validé par un Http Post sur 4DBKHttpPostProcess*

- *Avantages* :
  - Pas de javascript pour l'exécution des commandes 4DBKFieldSet.
  - Contrôle complet au niveau du stockage.
- *Contraintes* :
  - 2 pages html.
- *Exemple* :
  - Voir les pages *account_free_post.htm* et *account_free_post2.htm* dans le site TEST.

3. *4DBKExecute*

- *Avantages* :
  - Tout le code réside dans la même page Web.
  - Contrôle complet au niveau du stockage.
  - Grande liberté de filtrage des données.
- *Contraintes* :
  - Ne fonctionne qu'avec Javascript.
- *Exemple* :
  - Voir la page *account.htm* dans le site TEST.

Dans les trois méthodes, il est souhaitable que le développeur teste ou filtre les valeurs saisies dans les champs du formulaire avant la validation de celui-ci au moyen d'un événement onSubmit dans l'en-tête <form>. Des librairies Javascript fournies avec 4D Business Kit permettent de vérifier par exemple si une adresse E-mail est valide ou si une donnée obligatoire a bien été saisie. Ces contrôles sont exécutés sur le navigateur, aucune requête vers le serveur n'est donc effectuée.

Note : 4D Business Kit considère les caractères **, : ; espace** et **&** comme caractères de contrôle. Par exemple, la virgule est utilisée comme séparateur de paramètres.
Si vous souhaitez placer un ou plusieurs de ces caractère(s) dans un champ, vous devez utiliser les caractères spéciaux suivants :

,	,
:	:
;	&semicolon;
espace	&space;
&	&

Par exemple, pour placer la valeur "3, rue du pont" dans le champ CusStreetMain, vous devez écrire :

```
4DBKFieldSet/CusStreetMain=3&comma; rue du pont
```

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemple

Ce lien permet de changer le nom du compte client courant :

```
<a href="4DBKExecute:4DBKStoreSet/DEM1;4DBKFieldSet/CusLastName=
        Martinez">
```

Référence : 4DBKField, 4DBKCustomerLogin, 4DBKCustomerLogout, 4DBKCustomerExists, 4DBKCustomerForm, 4DBKPrefsSet.

4DBKFieldMultipleSize

<!--#4DBKFieldMultipleSize/champ-->

Paramètre	Description	Valeur(s)
champ	Nom ou code de champ de données multiples	• M01 à M05 : données multiples • Nom de champ de type Mxx présent dans le calque nommé STD, Standard ou le cas échéant, dans le premier calque rencontré

Objectif : Retourner le nombre de données multiples liées à l'article courant.

Description

La commande 4DBKFieldMultipleSize retourne le nombre de données multiples associées au *champ* de l'article courant.

Cette commande est utilisée dans le cadre de l'affichage des données multiples d'un article.

Dans un lien, si vous souhaitez que la valeur soit retournée au moment du chargement de la page et non au moment où l'utilisateur clique sur le lien, utilisez l'instruction !4DBKFieldMultipleSize.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet, la sélection courante à l'aide de la commande 4DBKSelectionSet et la fiche courante à l'aide de la commande 4DBKRecordSet.

Exemple

L'exemple suivant permet d'indiquer le nombre de données multiples d'un champ donné :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1
<!--#4DBKRecordSet/1--> //passer sur la 1re fiche
<!--#4DBKFieldMultipleSize/M01--> //si le champ M01 contient rouge, vert,
        bleu, alors cette ligne d'instruction retourne 3
```

Référence : 4DBKFieldMultiple.

4DBKFieldMultiple

<!--#4DBKFieldMultiple/champ,index[,format]-->

Paramètre	Description	Valeur(s)
champ	Nom ou code de champ de données multiples	• M01 à M05 : données multiples • Nom de champ de type Mxx présent dans le calque nommé STD, Standard ou le cas échéant, dans le premier calque rencontré
index	Numéro du Nième élément de données multiples	Numérique
format	Format d'affichage valide	Voir chapitre Formats d'affichage et fonctions

Objectif : Retourner la valeur de la Nième donnée multiple liée à l'article courant.

Description

La commande 4DBKFieldMultiple retourne le Nième élément du *champ* de données multiples pour l'article courant. Cette commande est utilisée dans le cadre de l'affichage des données multiples d'un article.

Dans un lien, si vous souhaitez que la valeur soit retournée au moment du chargement de la page et non au moment où l'utilisateur clique sur le lien, utilisez l'instruction !4DBKFieldMultiple.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet, la sélection courante à l'aide de la commande 4DBKSelectionSet et la fiche courante à l'aide de la commande 4DBKRecordSet.

Exemple

L'exemple suivant retourne chaque donnée multiple d'un champ à l'aide d'une boucle :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1
<!--#4DBKRecordSet/1--> //passer sur la 1re fiche
<!--#4DBKLoop/VL01,1,4DBKFieldMultipleSize/M01--> //bou-
        cle(VL01;1;nombre de données multiples de M01)
```

```
<!--#4DBKFieldMultiple/M01,4DBKVar/VL01--><br> //si le champ M01 contient rouge, vert, bleu alors cette instruction retourne rouge puis vert puis bleu
<!--#4DBKEndLoop/VL01--> //fin de boucle
```

Référence : 4DBKFieldMultipleSize.

4DBKFieldLinkedSize

<!--#4DBKFieldLinkedSize-->
Cette commande ne requiert pas de paramètre.

Objectif : Retourner le nombre d'articles liés à l'article courant.

Description

La commande 4DBKFieldLinkedSize retourne le nombre d'articles liés à l'article courant.

Cette commande est utilisée pour afficher une liste d'articles liés.

Dans un lien, si vous souhaitez que la valeur soit retournée au moment du chargement de la page et non au moment où l'utilisateur clique sur le lien, utilisez l'instruction !4DBKFieldLinkedSize.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet, la sélection courante à l'aide de la commande 4DBKSelectionSet et la fiche courante à l'aide de la commande 4DBKRecordSet.

Exemple

L'exemple suivant permet de retourner le nombre d'articles liés au premier article de la sélection :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1
<!--#4DBKRecordSet/1--> //passer sur la 1re fiche
<!--#4DBKFieldLinkedSize--> //retourner le nombre d'articles liés à l'article correspondant à la 1re fiche
```

Référence : 4DBKFieldLinked.

4DBKFieldLinked

<!--#4DBKFieldLinked/index-->

Paramètre	Description	Valeur(s)
index	Numéro d'article lié	Numérique

Objectif : Retourner le code du N^{ième} article lié à l'article courant.

Description

La commande 4DBKFieldLinked retourne le code du Nième article (N étant défini par le paramètre *index)* lié à l'article courant. Cette commande est utilisée pour afficher une liste d'articles liés.

Dans un lien, si vous souhaitez que la valeur soit retournée au moment du chargement de la page et non au moment où l'utilisateur clique sur le lien, utilisez l'instruction !4DBKFieldLinked.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet, la sélection courante à l'aide de la commande 4DBKSelectionSet et la fiche courante à l'aide de la commande 4DBKRecordSet.

Exemple

L'exemple suivant permet d'afficher le nom de chaque article lié à un article donné :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1
<!--#4DBKRecordSet/1--> //passer sur la 1re fiche
<!--#4DBKLoop/VL01,1,4DBKFieldLinkedSize--> //boucle(VL01;1;nombre
        d'articles liés)
<!--#4DBKQuerySet/C01=4DBKDetail--> //chercher la fiche détail courante
<!--#4DBKQuerySet/C01=4DBKFieldLinked/4DBKVar/VL01--> //chercher le
        VL01ième article lié
<!--#4DBKRecordSet/1--> //début sélection
<!--#4DBKField/produit--><br> //afficher le nom de l'article lié
<!--#4DBKEndLoop/VL01--> //fin de boucle
```

Référence : 4DBKFieldLinkedSize.

4DBKPictureExists

<!--#4DBKPictureExists/index-->

Paramètre	Description	Valeur(s)
index	Numéro de vue de l'article courant	Numérique

Objectif : Tester l'existence d'une N^{ième} image pour l'article courant.

Description

La commande 4DBKPictureExists retourne True ou False selon que la vue numéro *index* de l'article courant contient ou non une image.

Cette commande est utilisée dans le cadre de l'affichage des vues d'un article.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet, la sélection courante à l'aide de la commande 4DBKSelectionSet et la fiche courante à l'aide de la commande 4DBKRecordSet.

Exemple

L'exemple suivant vérifie s'il existe une image pour une fiche donnée dans la sélection et le cas échéant l'affiche :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1
<!--#4DBKRecordSet/1--> //passer sur la 1re fiche
<!--#4DBKIf (4DBKPictureExists/4)--> //s'il y a une 4e image associée à l'article courant
<!--#4DBKPicture/4--> //alors afficher cette image
<!--#4DBKEndif--> //fin de si
```

Référence : 4DBKThumbnailExists, 4DBKIconExists, 4DBKTextExists, 4DBKPicture.

4DBKThumbnailExists

<!--#4DBKThumbnailExists/index-->

Paramètre	Description	Valeur(s)
index	Numéro de vue de l'article courant	Numérique

Objectif : Tester l'existence d'une N^{ième} imagette pour l'article courant.

Description

La commande 4DBKThumbnailExists retourne True ou False selon que la vue numéro *index* de l'article courant contient ou non une imagette. Cette commande est utilisée dans le cadre de l'affichage des vues d'un article.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet, la sélection courante à l'aide de 4DBKSelectionSet et la fiche courante à l'aide de 4DBKRecordSet.

Exemple

Reportez-vous à l'exemple de la commande 4DBKPictureExists.

Référence : 4DBKPictureExists, 4DBKIconExists, 4DBKTextExists, 4DBKThumbnail.

4DBKIconExists

<!--#4DBKIconExists/index-->

Paramètre	Description	Valeur(s)
index	Numéro de vue de l'article courant	Numérique

Objectif : Tester l'existence d'une N^{ième} icône pour l'article courant.

Description

La commande 4DBKIconExists retourne True ou False selon que la vue numéro *index* de l'article courant contient ou non une icône.
Cette commande est utilisée dans le cadre de l'affichage des vues d'un article.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet, la sélection courante à l'aide de la commande 4DBKSelectionSet et la fiche courante à l'aide de la commande 4DBKRecordSet.

Exemple

Reportez-vous à l'exemple de la commande 4DBKPictureExists.

Référence : 4DBKPictureExists, 4DBKThumbnailExists, 4DBKTextExists, 4DBKIcon.

4DBKTextExists

<!--#4DBKTextExists/index-->

Paramètre	Description	Valeur(s)
index	Numéro de vue de l'article courant	Numérique

Objectif : Tester l'existence d'une N^{ième} description d'image pour l'article courant.

Description

La commande 4DBKTextExists retourne True ou False selon que la vue numéro *index* de l'article courant contient ou non une description.

Cette commande est utilisée dans le cadre de l'affichage des vues d'un article.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet, la sélection courante à l'aide de la commande 4DBKSelectionSet et la fiche courante à l'aide de la commande 4DBKRecordSet.

Exemple

Reportez-vous à l'exemple de la commande 4DBKPictureExists.

Référence : 4DBKPictureExists, 4DBKThumbnailExists, 4DBKIconExists, 4DBKText.

4DBKPicture

<!--#4DBKPicture/index[,info]-->

Paramètre	Description	Valeur(s)
index	Numéro de vue de l'article courant	Numérique
info	Information à obtenir	W=largeur en pixels H=hauteur en pixels S=taille en octets

Objectif : Retourner le chemin d'accès ou une caractéristique spécifique de la N^{ième} image de l'article courant.

Description

Lorsque le paramètre optionnel *info* n'est pas utilisé, la commande 4DBKPicture retourne le chemin d'accès disque de l'image de la vue numéro *index* de l'article courant. Cette commande permet d'afficher l'image associée à un article.

Dans un lien, si vous souhaitez que la valeur soit retournée au moment du chargement de la page et non au moment où l'utilisateur clique sur le lien, utilisez l'instruction !4DBKPicture.

Lorsque le paramètre *info* est utilisé, la commande retourne une caractéristique de taille de l'image. Trois caractéristiques peuvent être retournées, en fonction de la valeur (W, H ou S) passée dans *info* :

- 4DBKPicture/index,W retourne la largeur de l'image en pixels.
- 4DBKPicture/index,H retourne la hauteur de l'image en pixels.
- 4DBKPicture/index,S retourne la taille de l'image en octets.

Note : Ces trois fonctions doivent charger l'image en mémoire pour pouvoir retourner les informations demandées, ce qui peut, selon la taille des images, augmenter sensiblement la charge processeur.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet, la sélection courante à l'aide de la commande 4DBKSelectionSet et la fiche courante à l'aide de la commande 4DBKRecordSet.

Exemple

Reportez-vous à l'exemple de la commande 4DBKThumbnail.

Référence : 4DBKThumbnail, 4DBKIcon, 4DBKText, 4DBKPictureExists.

4DBKThumbnail

<!--#4DBKThumbnail/index[,info]-->

Paramètre	Description	Valeur(s)
index	Numéro de vue de l'article courant	Numérique
info	Information à obtenir	W=largeur en pixels H=hauteur en pixels S=taille en octets

Objectif : Retourne le chemin d'accès ou une caractéristique spécifique de la N^{ième} image réduite de l'article courant.

Description

Lorsque le paramètre optionnel *info* n'est pas utilisé, la commande 4DBKThumbnail retourne le chemin d'accès disque de l'image réduite de la vue numéro *index* de l'article courant. Cette commande permet d'afficher l'image réduite associée à un article.

Dans un lien, si vous souhaitez que la valeur soit retournée au moment du chargement de la page et non au moment où l'utilisateur clique sur le lien, utilisez l'instruction !4DBKThumbnail.

Lorsque le paramètre *info* est utilisé, la commande retourne une caractéristique de taille de l'image réduite. Trois caractéristiques peuvent être retournées, en fonction de la valeur (W, H ou S) passée dans *info* :

- 4DBKPicture/index,W retourne la largeur en pixels de l'image réduite.
- 4DBKPicture/index,H retourne la hauteur en pixels de l'image réduite.
- 4DBKPicture/index,S retourne la taille en octets de l'image réduite.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet, la sélection courante à l'aide de la commande 4DBKSelectionSet et la fiche courante à l'aide de la commande 4DBKRecordSet.

Exemple

Pour les trois fiches de la sélection numéro 1, cet exemple affiche l'image réduite si elle existe :

```
<!--#4DBKStoreSet/MCOM--> //passe sur la boutique MCOM
<!--#4DBKSelectionSet/1--> //passe sur la sélection numéro 1
<!--#4DBKRecordSet/1--> //passe sur la 1re fiche
<!--#4DBKLoop/VL01,1,3--> //boucle(VL01;1;3)
<!--#4DBKIf (4DBKThumbnailExists/4DBKVar/VL01)--> //s'il y a une VL01
        ième image réduite associée à l'article courant
<!--#4DBKThumbnail/4DBKVar/VL01-> //alors affiche cette image réduite
<!--#4DBKEndif--> //fin de si
<!--#4DBKRecordNext--> //passe à la fiche suivante
<!--#4DBKEndLoop/VL01--> //fin de boucle
```

Référence : 4DBKPicture, 4DBKIcon, 4DBKText, 4DBKThumbnailExists.

4DBKIcon

<!--#4DBKIcon/index[,S]-->

Paramètre	Description	Valeur(s)
index	Numéro de vue de l'article courant	Numérique
S	Information à obtenir	S : Taille en octets

Objectif : Retourner le chemin d'accès ou la taille de la N^{ième} icône de l'article courant.

Description

Par défaut, la commande 4DBKIcon retourne le chemin d'accès disque de l'icône de la vue numéro *index* de l'article courant. Cette commande permet d'afficher l'icône associée à un article.

Dans un lien, si vous souhaitez que la valeur soit retournée au moment du chargement de la page et non au moment où l'utilisateur clique sur le lien, utilisez l'instruction !4DBKIcon.

Lorsque la syntaxe 4DBKIcon/index,S est utilisée, la fonction retourne la taille en octets de l'icône.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet, la sélection courante à l'aide de la commande 4DBKSelectionSet et la fiche courante à l'aide de la commande 4DBKRecordSet.

Exemple

Reportez-vous à l'exemple de la commande 4DBKThumbnail.

Référence : 4DBKThumbnail, 4DBKPicture, 4DBKText, 4DBKIconExists.

4DBKText

<!--#4DBKText/index[,S]-->

Paramètre	Description	Valeur(s)
index	Numéro de vue de l'article courant	Numérique
S	Information à obtenir	S : Taille en octets

Objectif : Retourner le chemin d'accès ou la taille de la N^{ième} description d'image de l'article courant.

Description

Par défaut, la commande 4DBKText retourne le chemin d'accès disque de la description d'image de la vue numéro *index* de l'article courant. Cette commande permet d'afficher la description d'image associée à un article.

Dans un lien, si vous souhaitez que la valeur soit retournée au moment du chargement de la page et non au moment où l'utilisateur clique sur le lien, utilisez l'instruction !4DBKText.

Lorsque la syntaxe 4DBKText/index,S est utilisée, la fonction retourne la taille en octets du texte de description.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet, la sélection courante à l'aide de la commande 4DBKSelectionSet et la fiche courante à l'aide de la commande 4DBKRecordSet.

Exemple

Reportez-vous à l'exemple de la commande 4DBKThumbnail.

Référence : 4DBKThumbnail, 4DBKPicture, 4DBKIcon, 4DBKTextExists.

4DBKCounterSet

<!--#4DBKCounterSet/valeur,compteur,+N-->
<!--#4DBKCounterSet/valeur,compteur,-N-->
<!--#4DBKCounterSet/valeur,compteur,=N-->

Paramètre	Description	Valeur(s)
valeur	Valeur du champ code C01 de l'article	Alpha 30
compteur	N° du compteur à afficher	1, 2 ou 3
+N	Valeur à ajouter au compteur	Numérique
-N	Valeur à supprimer du compteur	
=N	Nouvelle valeur du compteur	

Objectif : Définir la valeur d'un des compteurs d'une fiche article.

Description

Chaque fiche article possède jusqu'à 3 compteurs de type réel qui peuvent contenir tout type de donnée numérique incrémentale (par exemple, le nombre de fois où l'article a été consulté ou téléchargé, le chiffre d'affaires réalisé sur cet article, etc.).

La valeur de ces compteurs peut être définie à l'aide de la commande 4DBKCounterSet.

Passez dans le paramètre *valeur* la valeur du champ code C01 de l'article dont vous voulez modifier le compteur. Passez dans le paramètre *compteur* le numéro (1, 2 ou 3) du compteur à modifier.

Vous pouvez passer une valeur *N* positive (/+N) ou négative (/-N), respectivement pour incrémenter ou décrémenter le compteur. Vous pouvez également passer une valeur fixe (/=N), dans ce cas la valeur définie remplacera la valeur courante du compteur.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemples

(1) Reportez-vous, dans le site TEST, à la page *counters.htm*.

(2) Exemple d'ajout de valeur :

<!--#4DBKCounterSet/WIN-98SE-CD,1,+1--> ajouter 1 au premier compteur de l'article dont le code est WIN-98SE-CD

Référence : 4DBKCounter.

4DBKCounter

<!--#4DBKCounter/valeur,compteur[,format]-->

Paramètre	Description	Valeur(s)
valeur	Valeur du champ code C01 de l'article	Alpha 30
compteur	N° du compteur à afficher	1, 2 ou 3
format	Format d'affichage réel valide	Voir chapitre Formats d'affichage et fonctions

Objectif : Afficher un des compteurs d'une fiche article.

Description

Chaque fiche article possède jusqu'à 3 compteurs de type réel qui peuvent contenir tout type de donnée numérique incrémentale (par exemple, le nombre de fois où l'article a été consulté ou téléchargé, le chiffre d'affaires réalisé sur cet article, etc.).

Ces compteurs sont positionnés par la commande 4DBKCounterSet et il est possible de les relire par la commande 4DBKCounter.

Passez dans le paramètre *valeur* la valeur du champ code C01 de l'article dont vous voulez afficher le compteur.

Passez dans le paramètre *compteur* le numéro (1, 2 ou 3) du compteur à afficher.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemples

(1) Reportez-vous, dans le site TEST, à la page *counters.htm*.

(2) Cet exemple affiche la valeur du compteur 1 :

```
<!--#4DBKCounter/WIN-98SE-CD,1,###'##0--> retourne le premier
        compteur de l'article dont le code est WIN-98SE-CD sous la forme
        ###'##0.
```

Référence : 4DBKCounterSet.

7 Panier

4DBKQuantitySet

<!--#4DBKQuantitySet/+N-->
<!--#4DBKQuantitySet/-N-->
<!--#4DBKQuantitySet/=N-->

Paramètre	Description	Valeur(s)
+N	Quantité à ajouter au panier	Numérique
-N	Quantité à supprimer du panier	
=N	Nouvelle quantité du panier	

Objectif : Définir la quantité à utiliser pour les opérations panier.

Description

La commande 4DBKQuantitySet spécifie la quantité qui sera utilisée pour les opérations panier, effectuées à l'aide de la commande 4DBKBasketSet. Cette commande doit être appelée avant toute commande 4DBKBasketSet.

Vous pouvez passer une quantité *N* positive (/+N) ou négative (/-N), respectivement pour ajouter ou supprimer des articles dans le panier. Vous pouvez également passer une valeur fixe (/=N), dans ce cas la quantité définie remplacera la quantité éventuellement définie dans le panier.

La quantité définie conservée pendant toute la session Web jusqu'à ce qu'une autre instruction 4DBKQuantitySet la remplace.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemple

A l'aide d'un lien, cet exemple ajoute l'article référencé 0012 au panier et recharge la page courante :

```
<a href="4DBKExecute:4DBKStoreSet/DEM1;4DBKQuantitySet/+1;
4DBKBasketSet/0012;4DBKNoCache">Lien</a>
```

4DBKInfoSet

<!--#4DBKInfoSet/commentaire-->

Paramètre	Description	Valeur(s)
commentaire	Commentaire à associer à un article du panier	Chaîne de caractères

Objectif : Associer un commentaire à un article du panier.

Description

La commande 4DBKInfoSet permet de définir un *commentaire* qui sera utilisé lors des opérations panier (par exemple pour une commande personnalisée). Cette commande doit être appelée avant la commande 4DBKBasketSet.

Le commentaire est conservé pendant toute le session Web jusqu'à ce qu'une autre instruction 4DBKInfoSet la remplace.

Cette information sera stockée dans le champ "Informations sur la commande" de la ligne de commande lors de l'exécution de 4DBKOrderValidate.

Pour effacer le contenu de ce commentaire, utilisez l'instruction <!--#4DBKInfoSet /-->.

Cette instruction est particulièrement utile lorsqu'il s'agit de stocker dans le panier des articles qui ont la même référence mais qui décrivent des combinaisons différentes. C'est le cas, par exemple, d'une commande dans un site "fast-food" de deux menus identiques sur la carte, mais dont les ingrédients varient (l'un contient un burger de type A et l'autre un burger de type B).

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemple

A l'aide d'un lien, cet exemple ajoute un article référencé 0012 au panier (avec comme commentaire 0004,0006,0018, pouvant être des numéros d'articles liés) et recharge la page courante :

```
<a href="4DBKExecute:4DBKStoreSet/DEM1;4DBKQuantitySet/+1;
        4DBKInfoSet/0004,0006,0018;4DBKBasketSet/0012;
        4DBKNoCache">Lien</a>
```

Référence : 4DBKOrderShippingSet, 4DBKOrderCommentSet, 4DBKOrderComment2Set, 4DBKOrderComment3Set, 4DBKInfoUpdate.

4DBKBasketSet

<!--#4DBKBasketSet/valeur-->

Paramètre	Description	Valeur(s)
valeur	Valeur du champ code C01 de l'article	Chaîne (30 caractères maxi)

Objectif : Ajouter ou retirer un article du panier.

Description

La commande 4DBKBasketSet ajoute ou retire un article du panier. Une seule référence d'article peut être traitée à chaque opération, toutefois la quantité d'articles peut varier.

Cette quantité ainsi que le type d'opération (ajout, retrait ou nouvelle quantité) doivent avoir été définis préalablement à l'aide de la commande 4DBKQuantitySet.

Note : Il est possible d'appeler cette commande sous le nom 4DBKCartSet en lieu et place de 4DBKBasketSet.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemple

L'exemple suivant effectue une recherche sur un type d'article puis un tri. Le champ "produit" de chacun des cinq premiers articles trouvés est affiché sous forme d'URL et sera ajouté au panier si l'internaute clique dessus.

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1
<!--#4DBKQuerySet/Type="ordinateur@"--> //effectuer une recherche
<!--#4DBKSortSet/produit--> //fixer le tri sur le champ produit
<!--#4DBKSelectionSort--> //trier la sélection numéro 1
<!--#4DBKRecordSet/C--> //passer à la 1re fiche de la liste
<!--#4DBKLoop/VL01,1,5--> //boucle(VL01;1;5)
<a href="4DBKExecute:4DBKStoreSet/MCOM;4DBKQuantitySet/+1;
        4DBKBasketSet/!4DBKField/C01;4DBKNoCache">
        4DBKField/produit</a>
<!--#4DBKRecordNext--> //passer à la fiche suivante
<!--#4DBKEndLoop/VL01--> //fin de boucle
```

4DBKInfoUpdate

<!--#4DBKInfoUpdate/index=valeur-->

Paramètre	Description	Valeur(s)
index	Numéro de la ligne du panier	Numérique
=valeur	Nouvelle valeur	Chaîne de caractères

Objectif : Modifier le texte discriminant associé à une ligne de commande.

Description

Le commentaire discriminant fixé par les commandes 4DBKInfoSet et 4DBKBasketSet permet de dissocier deux lignes de commande du panier qui concernent le même article.

Dans certains cas, il peut être utile de modifier ce texte. Par exemple, si la vente concerne une mise à jour de logiciel et que le client souhaite mettre à jour le numéro de série qu'il avait saisi un peu plus tôt. La commande 4DBKInfoUpdate permet de modifier ce commentaire.

Passez dans le paramètre *index* le numéro de la ligne de commande à modifier. Ce numéro peut être obtenu par le champ BskId (4DBKField/BskId).

Exemples

Voici deux exemples d'utilisation typique de cette commande :

<!--#4DBKInfoUpdate/1=test--> // le texte discriminant de la ligne de commande n°1 devient test

<!--#4DBKInfoUpdate/4DBKVar/VL30=4DBKVar/VT30--> // le nouveau numéro de série a été saisi et placé dans la variable VT30, le numéro de la ligne du panier a été placé dans la variable VL30

Référence : 4DBKInfoSet.

8 Identification

4DBKCustomerLogin

<!--#4DBKCustomerLogin/code,motDePasse[,pgRéussite[,pgEchec]]-->

Paramètre	Description	Valeur(s)
code	Code d'accès du client	Chaîne de caractères
motDePasse	Mot de passe du client	Chaîne de caractères
pgRéussite	Page à afficher si le client est correctement identifié	Chemin HTML
pgEchec	Page à afficher si le client n'est pas correctement identifié	Chemin HTML

Objectif : Identifier le client.

Description

La commande 4DBKCustomerLogin permet d'identifier la connexion d'un client. Cette commande doit être appelée lors de la validation d'un formulaire d'identification.

L'étape d'identification est indispensable pour valider une commande. Une fois le client identifié, sa commande est recalculée pour tenir compte des remises auxquelles il a droit, du ou des taux de taxe applicables, etc.

- Si le client est correctement identifié, le champ *CustomerLogged* (4DBKField/CustomerLogged) prend la valeur True et le serveur 4D Business Kit charge la page définie par le paramètre *pgRéussite*.

- Si l'identification du client a échoué, le champ *CustomerLogged* (4DBKField/CustomerLogged) prend la valeur False et le serveur 4D Business Kit charge la page définie par le paramètre *pgEchec*. Dans ce cas, il est possible de tester la cause de l'échec à l'aide des champs *CustomerLoginStatus* (4DBKField/CustomerLoginStatus) qui est permanent ou *ErrorCode* (4DBKField/ErrorCode) qui prennent les valeurs suivantes :
  - EMP_COD : le code saisi est vide
  - EMP_PWD : le mot de passe saisi est vide
  - ACC_LCK : le code correspond à un client radié
  - ABD_PWD : le mot de passe est incorrect
  - CUS_UNK : le code saisi ne correspond à aucun client

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemple

L'identité d'un client est envoyée via un formulaire pour être vérifiée :

```
<form name="form" method="post" action="javascript:window.top.
        location.href='4DBKExecute:4DBKStoreSet/TEST;
        4DBKCustomerLogin/'+document.form.CustomerCode.value+','+
        document.form.CustomerPassword.value+',home.htm,
        error.htm';">
```

Dans le cas où l'identification n'est pas correcte, la page "error.htm" teste le motif d'invalidité :

```
<!--#4DBKStoreSet/TEST-->
<!--#4DBKCaseOf-->
<!--#4DBKCase/"4DBKField/ErrorCode"="EMP_COD"-->Vous n'avez pas en-
        tré de code d'accès...
<!--#4DBKCase/"4DBKField/ErrorCode"="EMP_PWD"-->Vous n'avez pas en-
        tré de mot de passe...
<!--#4DBKCase/"4DBKField/ErrorCode"="ACC_LCK"-->Votre compte client a
        été radié...
<!--#4DBKCase/"4DBKField/ErrorCode"="BAD_PWD"-->Ce mot de passe est
        invalide...
<!--#4DBKCase/"4DBKField/ErrorCode"="CUS_UNK"-->Ce code d'accès est
        invalide...
<!--#4DBKEndCaseOf-->
```

Référence : 4DBKCustomerLogout.

4DBKCustomerLogout

<!--#4DBKCustomerLogout-->
Cette commande ne requiert pas de paramètre.

Objectif : Annuler l'identification du client.

Description

Lorsqu'un client est identifié, il peut être utile d'annuler son identification, soit pour permettre à un autre client de s'identifier dans la même session Web (par exemple pour une borne Internet de réservations dans une galerie commerciale), soit parce que le client souhaite créer un nouveau compte Web.

Une fois qu'un client n'est plus identifié, le champ CustomerLogged (4DBKField/CustomerLogged) prend la valeur False et le prix des articles est éventuellement recalculé en fonction du nouveau contexte.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemple

Dans le lien suivant, l'identification du client est annulée s'il était identifié. Le client est alors transféré vers une page de saisie de compte afin de créer un nouveau compte :

```
<a href="4DBKExecute:4DBKStoreSet/DEM1;4DBKCustomerLogout;
        4DBKGo/account.htm">
```

Référence : 4DBKCustomerLogin.

4DBKCustomerExists

<!--#4DBKCustomerExists/code[,motDePasse][,format]-->

Paramètre	Description	Valeur(s)
code	Code d'accès du client	Chaîne de caractères
motDePasse	Mot de passe du client	Chaîne de caractères
format	Format d'affichage valide	Voir chapitre Formats d'affichage et fonctions

Objectif : Tester l'existence d'un compte client.

Description

La commande 4DBKCustomerExists permet de tester l'existence d'un compte client dans la boutique avant son identification.

Cette commande retourne les valeurs suivantes :

- pas de valeur : le client existe
- [Err=CUS_UNK] : le client n'existe pas dans la boutique
- [Err=CUS_LCK] : le client existe mais a été radié.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemple

Cette structure conditionnelle permet d'exécuter une partie de code différente en fonction de l'existence d'un client :

```
<!--#4DBKIf (4DBKCustomerExists/kurt,E)-->
    kurt existe
<!--#4DBKElse-->
    kurt n'existe pas
<!--#4DBKEndif-->
```

Référence : 4DBKCustomerLogin, 4DBKCustomerLogout.

4DBKCustomerForm

<form method="POST" action="/4daction/4DBKCustomerForm/boutique/page">

Paramètre	Description	Valeur(s)
boutique	Code boutique	Chaîne de 4 caractères
page	Page à afficher après le stockage des champs	Chaîne de caractères

Objectif : Traiter automatiquement la validation du formulaire de compte client à l'aide d'une requête HTTP POST.

Description

La commande 4DBKCustomerForm permet de valider très simplement un formulaire de compte client.

Après validation du formulaire, les champs présents sont stockés automatiquement puis l'utilisateur est redirigé vers la page désignée par le paramètre *page*.

Les noms des champs présents dans le formulaire doivent être identiques à ceux des champs clients (par exemple CusCode pour le code client). La liste des champs des comptes clients qu'il est possible d'afficher et de modifier est fournie dans le section 19 on page 177.

Cette méthode de saisie des fiches client permet de s'affranchir du code Javascript ; il est néanmois souhaitable pour le développeur de tester ou de filtrer les valeurs saisies dans les champs avant leur validation au moyen d'un événement onSubmit dans l'entête *<form>*.

Si vous souhaitez avoir un meilleur contrôle sur le stockage des champs dans la fiche client, utilisez plutôt la commande 4DBKFieldSet (des exemples de ce type sont fournis dans le site TEST avec les pages *account.htm* ou les pages *account_free_post.htm* et *account_free_post2.htm*).

> *Note :* Certains réglages de la commande 4DBKPrefsSet (*CusErrorDuplicateAdd*, *CusErrorDuplicateUpd*, *CusErrorLocked*, *CusErrorMode*) peuvent modifier le fonctionnement de cette commande.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemples

(1) Voir le site TEST, page *account_post.htm*.

(2) Voici une ligne complète de validation de formulaire avec vérification de la saisie via Javascript :

```
<form name="myForm" enctype="application/x-www-form-urlencoded"
        method="POST" action="/4daction/4DBKCustomerForm/TEST/
        myPage.htm" onsubmit="return CheckmyForm(document.my-
        Form);">
```

Référence : 4DBKHttpPostProcess, 4DBKFieldSet, 4DBKPrefsSet.

4DBKCustomerCodeNew

<!--#4DBKCustomerCodeNew-->
Cette commande ne requiert pas de paramètre.

Objectif : Réserver le numéro de client utilisé lors de la création d'un nouveau compte client.

Description

Lors de la demande de création d'un nouveau compte client par l'utilisateur, un appel à cette fonction permet de réserver le premier numéro de client libre suivant (cf. page **Données** de la fenêtre des propriétés de la boutique).

Ce numéro est retourné par la fonction mais il est également possible de le connaître en affichant le champ CusCode — même si à cet instant aucun client n'est encore identifié.

Exemples

(1) Reportez-vous au site TEST, page *account.htm*.

(2) Cet exemple permet de proposer à l'utilisateur un champ code (son code d'identification) pré-rempli avec un numéro unique :

```
   // Si création d'un nouveau compte :
<!--#4DBKCustomerCodeNew-->
   // Si affichage du formulaire de saisie client :
   <input name="CusCode" type="text" value="<!--#4DBKField/
        CusCode-->">
```

Référence : 4DBKField.

4DBKCustomerMemo

<!--#4DBKCustomerMemo/code-->

Paramètre	Description	Valeur(s)
code	Code client	Chaîne de caractères

Objectif : Retourner le mémo de l'utilisateur pour lui permettre de se souvenir de son mot de passe.

Description

Cette commande affiche le contenu du champ "Mémo" associé à l'utilisateur désigné par *code*.

Lorsqu'un utilisateur a perdu son mot de passe, la première chose à faire est de lui proposer le "mémo" qu'il avait défini, lors de son inscription, pour se souvenir de son mot de passe.

> *Note :* Dans une 2e phase, il sera possible de lui renvoyer son mot de passe à son adresse E-mail à l'aide de la commande 4DBKCustomerMailPwd.

Exemples

(1) Reportez-vous, dans le site TEST, à la page *lostpassword2.htm*.

(2) Exemple simple d'utilisation :

<!--#4DBKCustomerMemo/email@host.com--> affiche le mémo du mot de passe de l'utilisateur dont le code est email@host.com

Référence : 4DBKCustomerMailPwd.

4DBKCustomerMailPwd

<!--#4DBKCustomerMailPwd/code[,sujet[,fichier]]-->

Paramètre	Description	Valeur(s)
code	Code client	Chaîne de caractères
sujet	Sujet du message à envoyer	Chaîne de caractères
fichier	Nom du fichier texte ou de la page html à envoyer	Chaîne de caractères

Objectif : Envoyer par E-mail son mot de passe à un client.

Description

Cette commande envoie un E-mail contenant son mot de passe à l'utilisateur désigné par *code*. Cette commande est utile lorsqu'un utilisateur a perdu son mot de passe.

Passez dans le paramètre *sujet* le titre de l'E-mail à envoyer (par exemple "Votre mot de passe").

L'E-mail envoyé peut être un fichier texte ou une page html, dont vous passez le chemin d'accès dans *fichier*. Le fichier texte ou la page html doivent contenir la balise [password] qui sera remplacée au moment de l'envoi par le mot de passe du client.

Exemples

(1) Reportez-vous, dans le site TEST, aux pages *lostpassword2.htm* et *lostpasswordmsg.txt* ou *lostpasswordmsg.htm*.

(2) Dans l'exemple suivant, l'utilisateur qui a perdu son mot de passe a pour code email@host.com. Les différentes possibilités d'utilisation de la commande sont présentées :

```
<!--4DBKCustomerMailPwd/email@host.com--> // Envoi d'un E-mail
        simplifié, généré automatiquement, contenant le mot de passe

<!--4DBKCustomerMailPwd/email@host.com,Votre mot de
        passe,lostpasswordmsg.txt--> // Envoi par E-mail du texte
        contenu dans le fichier lostpasswordmsg.txt

<!--#4DBKCustomerMailPwd/email@host.com,Votre mot de
        passe,lostpasswordmsg.htm--> // Envoi par E-mail de la page
        HTML lostpasswordmsg.htm
```

Référence : 4DBKCustomerMemo.

4DBKVATNumberSet

<!--#4DBKVATNumberSet/numéro-->

Paramètre	Description	Valeur(s)
numéro	Numéro de TVA intracommunautaire	Chaîne

Objectif : Définir le numéro de TVA intracommunautaire du client courant.

Description

La commande 4DBKVATNumberSet permet de fixer le *numéro* de TVA intracommunautaire du client courant.

Ce numéro ne concerne que les boutiques situées à l'intérieur de la CEE. Dans ce cas, lorsqu'un client est une société disposant d'un numéro de TVA intracommunautaire, les articles peuvent lui être facturés sans TVA si l'option **Ne pas appliquer cette taxe** est sélectionnée pour la taxe TVA et si les pays du client et de l'expéditeur sont différents.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet et le client courant à l'aide de la commande 4DBKCustomerLogin.

Référence : 4DBKVATNumber.

4DBKVATNumber

<!--#4DBKVATNumber-->
Cette commande ne requiert pas de paramètre.

Objectif : Retourner le numéro de TVA intracommunautaire du client courant.

Description

La commande 4DBKVATNumber retourne le numéro de TVA intracommunautaire du client courant. Pour plus d'informations sur ce numéro, reportez-vous à la description de la commande 4DBKVATNumberSet.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet et le client courant à l'aide de la commande 4DBKCustomerLogin.

Référence : 4DBKVATNumberSet.

9 Données multiples

4DBKStoreMultipleSize

<!--#4DBKStoreMultipleSize/champ-->

Paramètre	Description	Valeur(s)
champ	Champ de données multiples	• M01 à M05 : données multiples • Nom de champ de type Mxx présent dans le calque nommé STD, Standard ou le cas échéant, dans le premier calque rencontré

Objectif : Retourner le nombre d'éléments distincts d'un champ de données multiples d'une boutique.

Description

La commande 4DBKStoreMultipleSize retourne le nombre de données multiples distinctes associées à un champ de la boutique courante. Cette commande est utile dans le cadre de l'affichage des données multiples. Elle doit être associée à la commande 4DBKStoreMultiple.

Dans un lien, si vous souhaitez que la valeur soit retournée au moment du chargement de la page et non au moment où l'utilisateur clique sur le lien, utilisez l'instruction !4DBKMultipleSize.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemple

L'exemple suivant permet d'indiquer le nombre de données multiples différentes dans le champ M01 :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKStoreMultipleSize/M01--> //retourner le nombre d'éléments différents pour le champ M01 dans la boutique
```

Référence : 4DBKStoreMultiple

4DBKStoreMultiple

<!--#4DBKStoreMultiple/champ,index[,format]-->

Paramètre	Description	Valeur(s)
champ	Champ de données multiples	• M01 à M05 : données multiples • Nom de champ de type Mxx présent dans le calque nommé STD, Standard ou le cas échéant, dans le premier calque rencontré
index	Numéro du Nième élément des données multiples triées par nom	Numérique
format	Format d'affichage valide	Voir chapitre Formats d'affichage et fonctions

Objectif : Retourner la Nième élément distinct d'un champ de données multiples d'une boutique.

Description

La commande 4DBKStoreMultiple retourne la valeur du Nième élément du *champ* de données multiples de la boutique courante. Cette commande, associée à la commande 4DBKStoreMultipleSize, permet d'afficher les données multiples.

Dans un lien, si vous souhaitez que la valeur soit retournée au moment du chargement de la page et non au moment où l'utilisateur clique sur le lien, utilisez l'instruction !4DBKFieldMultiple.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemple

L'exemple suivant permet d'afficher toutes les données multiples du champ M01 :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKLoop/VL01,1,4DBKStoreMultipleSize/M01--> //bou-
        cle(VL01;1;nombre d'éléments de données multiples de M01)
<!--#4DBKStoreMultiple/M01,4DBKVar/VL01--><br> //Afficher le Nième
        élément de M01
<!--#4DBKEndLoop/VL01--> //fin de boucle
```

Référence : 4DBKStoreMultipleSize

10 Modes d’expédition

La liste des modes d’expédition possibles pour une commande est calculée automatiquement une fois que le client est identifié et si le panier n’est pas vide. Dans le cas contraire, elle est vide.

La liste est établie en fonction du pays de livraison du client et de la liste des frais de port définis pour le mode de calcul utilisé par la boutique (à la quantité d’articles, au poids ou au prix).

Vous pouvez construire un menu ou une liste des modes d’expédition possibles à l’aide d’une boucle de 1 à 4DBKShippingSize. A chaque itération, vous afficherez le N^{ième} mode d’expédition avec la commande 4DBKShippingLabel par exemple.

Une fois que le client a choisi le mode d’expédition qui lui convient, vous pourrez appeler 4DBKShippingSet et le prix de la commande en cours sera recalculé pour tenir compte de ces nouveaux paramètres.

4DBKShippingSize

<!--#4DBKShippingSize-->
Cette commande ne requiert pas de paramètre.

Objectif : Retourner le nombre de modes d’expédition possibles pour une commande et un client donnés.

Description

La commande 4DBKShippingSize retourne le nombre de modes d’expédition possibles pour la commande en cours et le client identifié.

Cette commande est utile pour l’affichage de la liste des modes d’expédition possibles. Elle doit être appelée lors de la procédure de clôture d’une commande.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemple

L'exemple suivant affiche le nombre de modes d'expédition associés à la boutique courante :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKShippingSize--> //retourne 3 (par exemple)
```

4DBKShipping

<!--#4DBKShipping-->
Cette commande ne requiert pas de paramètre.

Objectif : Retourner le numéro d'indice du mode d'expédition choisi par le client dans la liste des frais de port.

Description

La commande 4DBKShipping retourne le numéro du mode d'expédition courant dans la liste des frais de port pour la boutique courante. Ce numéro peut avoir été choisi par le client ou défini à l'aide de la commande 4DBKShippingSet.

Cette commande doit être appelée lors de la procédure de clôture d'une commande.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemple

L'exemple suivant retourne le mode d'expédition courant associé à une boutique donnée :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKShipping-> //retourne 2 (par exemple)
```

4DBKShippingSet

<!--#4DBKShippingSet/index-->

Paramètre	Description	Valeur(s)
index	Numéro d'indice du mode d'expédition dans la liste des frais de port	De 0 à 4DBKShippingSize

Objectif : Définir le mode d'expédition.

Description

La commande 4DBKShippingSet définit le mode d'expédition courant pour la boutique courante. Vous devez passer dans le paramètre *index* le numéro d'indice, dans la liste des frais de port possibles, du mode d'expédition à utiliser.

La commande en cours est automatiquement recalculée en tenant compte des charges liées au mode d'expédition choisi.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemple

L'exemple suivant définit le mode d'expédition associé à la boutique :

```
<!--#4DBKStoreSet/TEST--> //passer sur la boutique MCOM
<!--#4DBKShippingSet/2--> //le mode d'expédition est maintenant le 2e
        dans la liste
```

4DBKShippingLabel

<!--#4DBKShippingLabel/index-->

Paramètre	Description	Valeur(s)
index	Numéro d'indice du mode d'expédition dans la liste des frais de port	De 0 à 4DBKShippingSize

Objectif : Afficher le libellé du Nième mode d'expédition.

Description

La commande 4DBKShippingLabel affiche le libellé du Nième élément de la liste des modes d'expédition possibles.
Elle est utile pour l'affichage de la liste des modes d'expédition possibles.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemple

Cet exemple construit la liste des modes d'expédition possibles :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKLoop/VL01,1,4DBKShippingSize--> //boucle sur les modes d'expédition
<!--#4DBKShippingLabel/4DBKVar/VL01--> : <!--#4DBKShippingINV.EUR,(+### ##0,00 EUR)/4DBKVar/VL01--> //afficher le mode et son prix
<!--#4DBKEndLoop/VL01-->
```

4DBKShippingComment

<!--#4DBKShippingComment/index-->

Paramètre	Description	Valeur(s)
index	Numéro d'indice du mode d'expédition dans la liste des frais de port	De 0 à 4DBKShippingSize

Objectif : Afficher le commentaire du Nième mode d'expédition.

Description

La commande 4DBKShippingComment affiche le commentaire associé au Nième élément de la liste des modes d'expédition possibles.

Cette commande peut être utile lors l'affichage de la liste des modes d'expédition possibles.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

4DBKShippingCode

<!--#4DBKShippingCode/index-->

Paramètre	Description	Valeur(s)
index	Numéro d'indice du mode d'expédition dans la liste des frais de port	De 0 à 4DBKShippingSize

Objectif : Afficher le code du Nième mode d'expédition.

Description

La commande 4DBKShippingCode affiche le code du Nième élément de la liste des modes d'expédition possibles.

Cette commande est utile pour l'affichage de la liste des modes d'expédition possibles.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemple

Reportez-vous à l'exemple de la commande 4DBKShippingLabel.

4DBKShippingINV

<!--#4DBKShippingINV.devise[,format]/index-->

Paramètre	Description	Valeur(s)
devise	Code de devise	Chaîne de 3 caractères
format	Format d'affichage valide	Voir chapitre Formats d'affichage et fonctions
index	Numéro d'indice du mode d'expédition dans la liste des frais de port	De 0 à 4DBKShippingSize

Objectif : Afficher le montant TTC du Nième mode d'expédition.

Description

La commande 4DBKShippingINV affiche le montant TTC correspondant au Nième élément de la liste des modes d'expédition possibles dans la devise définie par le paramètre *devise*. Ce montant tient compte des caractéristiques de la commande en cours.

Cette commande est utile pour l'affichage de la liste des modes d'expédition possibles.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemple

Reportez-vous à l'exemple de la commande 4DBKShippingLabel.

4DBKShippingEXV

<!--#4DBKShippingEXV.devise[,format]/index-->

Paramètre	Description	Valeur(s)
devise	Code de devise	Chaîne de 3 caractères
format	Format d'affichage valide	Voir chapitre Formats d'affichage et fonctions
index	Numéro d'indice du mode d'expédition dans la liste des frais de port	De 0 à 4DBKShippingSize

Objectif : Afficher le montant HT du Nième mode d'expédition.

Description

La commande 4DBKShippingINV affiche le montant HT correspondant au Nième élément de la liste des modes d'expédition possibles dans la devise définie par le paramètre *devise*. Ce montant tient compte des caractéristiques de la commande en cours.

Cette commande est utile pour l'affichage de la liste des modes d'expédition possibles.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemple

Reportez-vous à l'exemple de la commande 4DBKShippingLabel.

11 Commandes

4DBKOrderPaymentSet

<!--#4DBKOrderPaymentSet/commentaire-->

Paramètre	Description	Valeur(s)
commentaire	Commentaire à associer au paiement de la commande	Chaîne (240 caractères maximum)

Objectif : Définir le commentaire associé au paiement de la commande en cours.

Description

La commande 4DBKOrderPaymentSet définit le *commentaire* à associer au paiement de la commande d'articles en cours.

Lorsque l'utilisateur choisit un mode de paiement, il est possible d'associer une chaîne de caractères à la commande en cours (par exemple les coordonnées de la carte bancaire utilisée).

Cette information sera cryptée et stockée en même temps que la commande lors de l'exécution de la commande 4DBKOrderValidate. Elle sera affichée dans le champ "Informations de paiement" de toutes les lignes de la commande — hors frais d'expédition.

Le niveau de cryptage est défini dans les paramètres de la boutique.

Si vous souhaitez que le commentaire soit vide, utilisez l'instruction <!--#4DBKOrderPaymentSet/-->.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemple

L'exemple suivant définit un commentaire qui sera enregistré dans la commande :

```
<!--#4DBKOderPaymentSet/Cheque--> // Le client a choisi de payer par chèque
```

4DBKOrderShippingSet

<!--#4DBKOrderShippingSet/commentaire-->

Paramètre	Description	Valeur(s)
commentaire	Commentaire à associer au mode d'expédition	Chaîne (240 caractères maximum)

Objectif : Définir le commentaire associé au mode d'expédition de la commande en cours.

Description

La commande 4DBKOrderShippingSet définit le *commentaire* sur le mode d'expédition associé à la commande d'articles en cours.

Lorsque l'utilisateur choisit ou valide le mode d'expédition de la commande, il est possible d'associer une chaîne de caractères à la commande en cours (par exemple, le commentaire peut être de la forme "Traiter cette commande en priorité" ou "Envoyer en recommandé").

Cette information sera stockée dans la ligne de commande destinée aux frais de port lors de l'exécution de la commande 4DBKOrderValidate, dans le champ "Informations sur la commande".

Si vous souhaitez que le commentaire soit vide, utilisez l'instruction <!--#4DBKOrderShippingSet/-->.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemple

L'exemple suivant permet de stocker dans la commande (ligne destinée aux frais de livraison) la mention "Urgent" :

```
<!--#4DBKOrderShippingSet/Urgent--> // Le client veut une livraison en urgent
```

Référence : 4DBKInfoSet, 4DBKOrderCommentSet, 4DBKOrderComment2Set, 4DBKOrderComment3Set.

4DBKOrderCommentSet, 4DBKOrderComment2Set, 4DBKOrderComment3Set

<!--#4DBKOrderCommentSet/commentaire-->

Paramètre	Description	Valeur(s)
commentaire	Commentaire à associer à la commande	Chaîne (80 caractères maximum)

Objectif : Définir le(s) commentaire(s) associé(s) à la commande en cours.

Description

Les commandes 4DBKOrderCommentSet, 4DBKOrderComment2Set et 4DBKOrderComment3Set définissent les zones de *commentaire* correspondantes à associer à la commande courante.

Il est possible d'associer jusqu'à trois commentaires libres (toute chaîne jusqu'à 80 caractères) à une commande en cours. Ces informations seront stockées en même temps que la commande lors de l'exécution de 4DBKOrderValidate. Elles apparaîtront dans les champs "Commentaire" de toutes les lignes de la commande.

Si vous souhaitez qu'un commentaire soit vide, utilisez l'instruction <!--#4DBKOrderCommentSet/-->.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemple

L'exemple suivant définit le commentaire qui sera enregistré dans la commande :

```
<!--#4DBKOrderCommentSet/Client interne --> // Le client appartient à
        l'entreprise
```

Référence : 4DBKOrderShippingSet, 4DBKInfoSet.

4DBKOrderValidate

<!--#4DBKOrderValidate[/pageRéussite[,pageEchec]]-->

Paramètre	Description	Valeur(s)
pageRéussite	Page à afficher si la commande a été correctement validée	Chemin HTML
pageEchec	Page à afficher si la validation de la commande a échoué	Chemin HTML

Objectif : Valider la commande d'article en cours.

Description

La commande 4DBKOrderValidate permet d'effectuer l'opération finale d'une commande d'articles : elle confirme la commande.

Avant d'appeler 4DBKOrderValidate, il est nécessaire de s'assurer que le panier contient bien des articles, qu'un mode d'expédition a bien été défini, qu'un client est bien identifié et qu'un mode de paiement a bien été choisi.

- Si la commande est valide, 4D Business Kit affiche la page désignée par le paramètre *pageRéussite* (s'il a été spécifié).
- Si la commande n'est pas valide, 4D Business Kit affiche la page désignée par le paramètre *pageEchec* (s'il a été spécifié).

Juste avant l'appel à la commande 4DBKOrderValidate, il peut être judicieux d'envoyer au client par courrier électronique son bon de commande et un message personnalisé à l'aide de la commande 4DBKMail.

Par défaut, les commandes sont stockées dans les données de la boutique et le marchand reçoit un courrier électronique formaté. Ces options peuvent être modifiées dans la fenêtre de paramétrage de la boutique, à la page "Commandes".

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemple

L'exemple suivant permet de confirmer une commande après avoir vérifié que le panier n'est pas vide, que l'identification est faite et qu'un mode d'expédition a été sélectionné :

```
<!--#4DBKIf(4DBKSelectionSizeB>0)--> // La commande n'est pas vide
<!--#4DBKIf(4DBKField/CustomerLogged)--> //Le client est identifié
<!--#4DBKIf(4DBKShipping>0)--> // Le mode d'expédition est choisi
// Dans ce cas, le formulaire de saisie du mode de paiement est affiché et à
        sa validation la commande est confirmée
<!--#4DBKOrderValidate,home.htm,error.htm-->
<!--#4DBKEndIf-->
<!--#4DBKEndIf-->
<!--#4DBKEndIf-->
```

Référence : 4DBKOrderClear.

4DBKOrderClear

<!--#4DBKOrderClear-->
Cette commande ne requiert pas de paramètre.

Objectif : Vider le panier.

Description

La commande 4DBKOrderClear permet de vider le panier du client courant par programmation.

Cette commande doit être utilisée conjointement à l'option **Ne pas vider le panier automatiquement** située dans la page "Commandes" de la fenêtre de paramétrage de la boutique. En effet, lorsque cette option est cochée, la commande 4DBKOrderClear permet au développeur du site de gérer le vidage du panier quand il le souhaite.
Lorsque l'option n'est pas cochée, le vidage du panier est effectué automatiquement au moment de l'exécution de la commande 4DBKOrderValidate (cf. page 114).

Le vidage non automatique des données du panier permet d'effectuer diverses opérations juste après la validation de la commande, notamment de mettre en place des envois de mails personnalisés au marchand et/ou au client.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet et validé le panier courant à l'aide de la commande 4DBKOrderValidate.

Exemple

L'exemple suivant suppose que l'option **Ne pas vider le panier automatiquement** de la page "Commandes" de la fenêtre de paramétrage de boutique est cochée. Le contenu du panier est donc conservé après que la commande ait été stockée sur le serveur 4D Business Kit. Une page HTML personnalisée contenant le récapitulatif de la commande est ensuite envoyée au client par email, puis le panier est vidé :

```
<!--#4DBKStoreSet/MCOM-->// on passe sur la boutique MCOM
<!--#4DBKOrderValidate-->// on stocke la commande sur le serveur
<!--#4DBKMail/order@shop.com,customer@thenet.net,Votre commande,
        commande.htm--> // envoi au client de l'email de confirmation
<!--#4DBKOrderClear-->//on vide le panier
```

Référence : 4DBKOrderValidate.

4DBKOrderCode

<!--#4DBKOrderCode-->
Cette commande ne requiert pas de paramètre.

Objectif : Retourner le numéro de la commande en cours.

Description

La commande 4DBKOrderCode retourne le numéro de la commande généré automatiquement par 4D Business Kit après un appel à 4DBKOrderValidate.

Cette commande permet d'afficher un message de confirmation au client après qu'il ait validé le formulaire de commande.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemple

L'exemple suivant retourne le numéro de commande (un appel à 4DBKOrderValidate a été effectué auparavant) :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKOrderCode--> //retourne 00154 (par exemple)
```

4DBKOrderCodeNew

<!--#4DBKOrderCodeNew-->
Cette commande ne requiert pas de paramètre.

Objectif : Réserver le numéro de commande utilisé à la validation de commande.

Description

Lorsque l'instruction 4DBKOrderValidate est appelée, une commande avec un numéro séquentiel est générée à partir des éléments contenus dans le panier et des caractéristiques du client identifié.

Il est ensuite possible de connaître ce numéro en utilisant la fonction 4DBKOrderCode.

Dans certains cas, par exemple lors de l'utilisation de passerelles de paiement, il peut être utile de connaître ce numéro avant que la commande ne soit créée. C'est le rôle de l'instruction 4DBKOrderCodeNew, qui permet de "réserver" un numéro de commande.

4DBKOrderCodeNew réserve le premier numéro libre suivant du module Commandes (cf. page **Données** de la fenêtre des propriétés de la boutique). Ce numéro peut être récupéré ensuite par la commande 4DBKOrderCode.

A chaque appel de 4DBKOrderCodeNew, un nouveau numéro est généré. Par conséquent, vous devez, à l'aide d'une variable par exemple, contrôler l'appel à cette commande pour éviter que plusieurs numéros ne soient perdus (cf. exemple).

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemple

Le positionnement de la variable VL40 détermine la réservation ou non d'un nouveau numéro de commande :

```
<!--#4DBKIf(4DBKVar/VL40=0)-->
  <!--#4DBKOrderCodeNew-->
  <!--#4DBKVarSet/VL40=1-->
<!--#4DBKEndIf-->
```

Référence : 4DBKOrderCode.

4DBKCreditCardProcess

<!--#4DBKCreditCardProcess/fichier,numéro,date,montant[,devise[,arg1[,arg2[,arg3]]]]-->

Paramètre	Description	Valeur(s)
fichier	Nom du fichier de configuration	Chaîne
numéro	Numéro de la carte bancaire	Numérique
date	Date d'expiration de la carte	Date (MM/AA)
montant	Montant à débiter sur la carte	Numérique
devise	Code de devise du montant	Chaîne de 3 caractères
arg1	Informations supplémentaires	Chaîne libre
arg2	Informations supplémentaires	Chaîne libre
arg3	Informations supplémentaires	Chaîne libre

Objectif : Débiter une carte bancaire d'un montant donné.

Description

Pour utiliser cette fonction, le marchand devra être abonné à un des services de paiement fournis avec 4D Business Kit (par exemple VeriSign ou Authorize.Net) ou téléchargé depuis le site Web de 4D.

> *Note :* Pour plus d'informations sur l'utilisation des services de paiement, reportez-vous au *Manuel technique avancé* de 4D Business Kit.

Chaque service de paiement est matérialisé par un fichier texte descriptif localisé dans un sous-dossier du serveur 4D Business Kit. C'est le nom de ce fichier qui doit être passé dans le paramètre *fichier*. Les fichiers de configuration sont situés à l'emplacement suivant :

- Sous Windows : Dossier 4DBK\Services\Payments\CreditCard\.
- Sous MacOS : Dossier 4DBK:Services:Payments:CreditCard:

Le paramètre *date* indique la date d'expiration de la carte sous la forme MM/AA (par exemple 06/03 pour juin 2003).

Le paramètre *montant* indique le montant à débiter. Il doit obligatoirement être sous la forme 00000.00 (par exemple 100.00), avec le point comme séparateur décimal.

Le paramètre *devise* indique la devise utilisée pour le montant. Une devise est en général définie par défaut (USD ou EUR), rendant ce paramètre facultatif. Certains services de paiement n'acceptent d'ailleurs qu'une seule ou qu'un petit nombre de devises.

Les paramètres facultatifs *arg1* à *arg3* permettent d'envoyer des informations supplémentaires à l'organisme de paiement si celui-ci les demande. Bien entendu, ces paramètres sont dans ce cas à préciser dans le script de paiement correspondant, défini dans le paramètre *fichier*.

La commande retourne de quatre à huit valeurs — en fonction des spécificités du service de paiement. Ces valeurs doivent être lues au sein de la session Web à l'aide des commandes-raccourcis suivantes :

```
<!--#4DBKCreditCardCode1-->
<!--#4DBKCreditCardCode2-->
<!--#4DBKCreditCardCode3-->
<!--#4DBKCreditCardCode4-->
<!--#4DBKCreditCardCode5-->
<!--#4DBKCreditCardCode6-->
<!--#4DBKCreditCardTID-->
<!--#4DBKCreditCardText-->
```

La signification de chaque information est généralement la suivante :

- 4DBKCreditCardCode1 retourne le code résultat de l'opération,
- 4DBKCreditCardCode2 retourne un code résultat plus précis,
- 4DBKCreditCardCode3 à 4DBKCreditCardCode6 retournent divers codes utilisés par certains services de paiement,
- 4DBKCreditCardTID retourne un numéro unique de transaction,
- 4DBKCreditCardText retourne une chaîne de caractères indiquant le résultat de l'opération.

Toutes ces instructions ne vous seront peut-être pas nécessaires : le nombre et la nature des informations retournées dépendent de la solution de paiement retenue.

Le sens précis de ces informations est décrit dans la documentation développeur fournie avec le kit de connexion de chaque service de paiement mais également sous la forme d'un résumé dans l'en-tête de chaque fichier de configuration livré avec 4D Business Kit.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemple

Dans cet exemple, nous souhaitons débiter 100 dollars sur la carte VISA 4444333322221111 en passant par Authorize.Net. Dans un premier temps, l'instruction suivante est exécutée :

```
<!--#4DBKCreditCardProcess/AuthorizeNet_AIM.txt,4444333322221111,
        01/02,100.00,USD-->
```

Les commandes suivantes retourneront alors, par exemple :

```
<!--#4DBKCreditCardCode1--> // 1
<!--#4DBKCreditCardCode2--> // 1
<!--#4DBKCreditCardTID--> // T012DJA9837D222
<!--#4DBKCreditCardText--> // This transaction has been approved
```

12 Suivi des commandes

Une fois le client identifié, il est possible d'afficher la liste de ses commandes passées et leur état courant.

Vous pouvez donc construire une liste des commandes en cours à l'aide d'une boucle de 1 à 4DBKOrdersSize. A chaque itération, vous afficherez la N^{ième} commande en cours avec l'instruction 4DBKOrdersNum par exemple.

4DBKOrdersSize

<!--#4DBKOrdersSize-->
Cette commande ne requiert pas de paramètre.

Objectif : Retourner le nombre de commandes passées par un client.

Description

La commande 4DBKOrdersSize retourne le nombre de commandes passées par un client. Le client doit impérativement être déjà identifié.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet et un client doit être identifié à l'aide de la commande 4DBKCustomerLogin.

Exemple

L'exemple suivant retourne le nombre de commandes passées par un client préalablement identifié :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKOrdersSize-> //retourne 3 (par exemple)
```

4DBKOrdersNum

<!--#4DBKOrdersNum/index-->

Paramètre	Description	Valeur(s)
index	Numéro d'indice de la commande dans la liste des commandes	De 0 à 4DBKOrdersSize

Objectif : Afficher le numéro de la Nième commande.

Description

La commande 4DBKOrdersNum affiche le numéro (code) de la Nième commande dans la liste des commandes passées par le client courant.

Cette commande est utile pour l'affichage de la liste des commandes passées d'un client.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet et un client doit être identifié à l'aide de la commande 4DBKCustomerLogin.

Exemple

L'exemple suivant affiche la liste des commandes du client courant ; pour chaque commande, le numéro et le libellé sont affichés :

```
<!--#4DBKStoreSet/TEST--> //passer sur la boutique MCOM
<!--#4DBKLoop/VL01,1,4DBKOrdersSize--> //boucle sur les commandes
<!--#4DBKOrdersNum/4DBKVar/VL01--> -
        <!--#4DBKOrdersLabel/4DBKVar/VL01--> //afficher le numéro et
        le libellé de la commande
<!--#4DBKEndLoop/VL01-->
```

4DBKOrdersDate

<!--#4DBKOrdersDate/index-->

Paramètre	Description	Valeur(s)
index	Numéro d'indice de la commande dans la liste des commandes	De 0 à 4DBKOrdersSize

Objectif : Afficher la date de la Nième commande.

Description

La commande 4DBKOrdersDate affiche la date de la Nième commande dans la liste des commandes passées par le client courant.

Cette commande est utile pour l'affichage de la liste des commandes passées d'un client.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet et un client doit être identifié à l'aide de la commande 4DBKCustomerLogin.

Exemple

Reportez-vous à l'exemple de la commande 4DBKOrdersNum.

4DBKOrdersLabel

<!--#4DBKOrdersLabel/index-->

Paramètre	Description	Valeur(s)
index	Numéro d'indice de la commande dans la liste des commandes	De 0 à 4DBKOrdersSize

Objectif : Afficher le libellé de la Nième commande.

Description

La commande 4DBKOrdersLabel affiche le libellé de la Nième commande dans la liste des commandes passées par le client courant.

Cette commande est utile pour l'affichage de la liste des commandes passées d'un client.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet et un client doit être identifié à l'aide de la commande 4DBKCustomerLogin.

Exemple

Reportez-vous à l'exemple de la commande 4DBKOrdersNum.

4DBKOrdersQty

!--#4DBKOrdersQty/index-->

Paramètre	Description	Valeur(s)
index	Numéro d'indice de la commande dans la liste des commandes	De 0 à 4DBKOrdersSize

Objectif : Afficher la quantité commandée de la Nième commande.

Description

La commande 4DBKOrdersQty affiche la quantité d'articles commandés de la Nième commande dans la liste des commandes passées par le client courant.

Cette commande est utile pour l'affichage de la liste des commandes passées d'un client.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet et un client doit être identifié à l'aide de la commande 4DBKCustomerLogin.

Exemple

Reportez-vous à l'exemple de la commande 4DBKOrdersNum.

4DBKOrdersRemainQty

!--#4DBKOrdersRemainQty/index-->

Paramètre	Description	Valeur(s)
index	Numéro d'indice de la commande dans la liste des commandes	De 0 à 4DBKOrdersSize

Objectif : Afficher la quantité restant à livrer de la Nième commande.

Description

La commande 4DBKOrdersRemainQty affiche la quantité d'articles restant à livrer de la Nième commande dans la liste des commandes passées par le client courant.

Cette commande est utile pour l'affichage de la liste des commandes passées d'un client.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet et un client doit être identifié à l'aide de la commande 4DBKCustomerLogin.

Exemple

Reportez-vous à l'exemple de la commande 4DBKOrdersNum.

4DBKOrdersEXV

!--#4DBKOrdersEXV.devise[,format]/index-->

Paramètre	Description	Valeur(s)
devise	Code de devise	Chaîne de 3 caractères
format	Format d'affichage valide	Voir chapitre Formats d'affichage et fonctions
index	Numéro d'indice de la commande dans la liste des commandes	De 0 à 4DBKOrdersSize

Objectif : Afficher le montant HT de la Nième commande.

Description

La commande 4DBKOrdersEXV affiche le montant HT de la Nième commande dans la liste des commandes passées par le client courant dans la *devise* et le *format* d'affichage (optionnel) définis.

Cette commande est utile pour l'affichage de la liste des commandes passées d'un client.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet et un client doit être identifié à l'aide de la commande 4DBKCustomerLogin.

Exemple

Reportez-vous à l'exemple de la commande 4DBKOrdersNum.

4DBKOrdersINV

!--#4DBKOrdersINV.devise[,format]/index-->

Paramètre	Description	Valeur(s)
devise	Code de devise	Chaîne de 3 caractères
format	Format d'affichage valide	Voir chapitre Formats d'affichage et fonctions
index	Numéro d'indice de la commande dans la liste des commandes	De 0 à 4DBKOrdersSize

Objectif : Afficher le montant TTC de la Nième commande.

Description

La commande 4DBKOrdersINV affiche le montant TTC de la Nième commande dans la liste des commandes passées par le client courant dans la *devise* et le *format* d'affichage (optionnel) définis.

Cette commande est utile pour l'affichage de la liste des commandes passées d'un client.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet et un client doit être identifié à l'aide de la commande 4DBKCustomerLogin.

Exemple

Reportez-vous à l'exemple de la commande 4DBKOrdersNum.

4DBKOrdersState

!--#4DBKOrdersState/index-->

Paramètre	Description	Valeur(s)
index	Numéro d'indice de la commande dans la liste des commandes	De 0 à 4DBKOrdersSize

Objectif : Afficher le statut de la Nième commande.

Description

La commande 4DBKOrdersState affiche le statut courant de la Nième commande dans la liste des commandes passées par le client courant.

Le statut indique si la commande a été traitée ou non.

Cette commande est utile pour l'affichage de la liste des commandes passées d'un client.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet et un client doit être identifié à l'aide de la commande 4DBKCustomerLogin.

Exemple

Reportez-vous à l'exemple de la commande 4DBKOrdersNum.

13 Variables

Vous disposez de 100 variables de type entier long (32 bits), 100 variables réelles (80 bits) et 100 variables texte (30 Ko). Il est possible d'affecter des valeurs à ces variables, d'effectuer des opérations mathématiques (sauf pour les variables texte) puis de les tester et de les afficher.

4DBKVarSet

<!--#4DBKVarSet/variable opérateur valeur OU autreFonction-->

Paramètre	Type	Description
variable	Nom de variable	• VL01 à VL100 : variables entier long (32 bits) • VR01 à VR100 : variables réelles (64 bits) • VT01 à VT100 : variables alphanumériques (30 Ko)
opérateur	Opérateur	=,+=,-=,*=,/=
valeur	Valeur à affecter à la variable	Valeur entière ou réelle ou alpha
autreFonction	Fonction 4D Business Kit	Toute fonction 4D Business Kit retournant une valeur

Objectif : Affecter une valeur à une variable.

Description

La commande 4DBKVarSet affecte une valeur à une variable. Cette commande permet par exemple de dimensionner des boucles ou de stocker des informations globales au site.

Vous pourrez également utiliser les variables pour construire des pages Web en utilisant des boucles, indiquer par un drapeau interne (*flag*) qu'une action a été effectuée dans une page puis tester ce drapeau dans une autre page, etc.

Une fois qu'une valeur est affectée à une variable, celle-ci est conservée pendant toute la durée de la session.

Le paramètre *opérateur* accepte les valeurs suivantes :

Opérateur	Action	Description
V=x	Affectation	V prend la valeur de x
V+=x	Addition	V prend la valeur V+x
V-=x	Soustraction	V prend la valeur V-x
V*=x	Multiplication	V prend la valeur V*x
V/=x	Division	V prend la valeur V/x

Tous les opérateurs de calcul sont applicables aux variables numériques.

En outre, deux opérateurs peuvent être utilisés avec des variables alphanumériques :

- += permet d'effectuer des concaténations de chaînes
- *= permet d'extraire des sous-chaînes.

Note : 4D Business Kit considère la virgule (,) comme séparateur de paramètres. Si vous souhaitez placer une ou plusieurs virgule(s) dans une variable, vous devez utiliser la chaîne **&COMMA;** pour chacune d'entre elles. Par exemple, pour placer la valeur "a,b,c" dans la variable T01, vous devez écrire :
4DBKVarSet/VT01=a&COMMA;b&COMMA;c

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemples

(1) L'exemple suivant présente les modes de définition des valeurs des variables ainsi que les opérateurs :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKVarSet/VL01=10--> //VL01:=10
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1
<!--#4DBKVarSet/VL01=4DBKSelectionSize--> //VL01:=3, nombre de fiches
      de la sélection numéro 1
<!--#4DBKVarSet/VT01=Bonjour--> //VT01:=Bonjour
<!--#4DBKVarSet/VL01+=2--> //VL01:=VL01+2
<!--#4DBKVarSet/VL01/=3--> //VL01:=VL01/3
<!--#4DBKVarSet/VL01-=4--> //VL01:=VL01-4
<!--#4DBKVarSet/VL01*=10--> //VL01:=VL01*10
```

(2) L'exemple suivant illustre l'emploi des opérateurs de calcul avec des variables alphanumériques :

```
      // Concaténation
<!--#4DBKVarSet/VT01=Voici--> // VT01 prend la valeur "Voici"
<!--#4DBKVar/VT01--> // affiche "Voici"
<!--#4DBKVarSet/VT02=un exemple--> // VT02 prend la valeur "un exem-
      ple"
<!--#4DBKVar/VT02--> // affiche "un exemple"
<!--#4DBKVarSet/VT01+=4DBKVar/VT02--> // VT01 prend la valeur
      VT01+VT02
<!--#4DBKVar/VT01--> // affiche "Voici un exemple"


      // Sous chaîne
<!--#4DBKVarSet/VT02=4DBKVar/VT01--> // VT02 prend la valeur de la
      VT01, "Voici un exemple"
<!--#4DBKVar/VT02--> // affiche "Voici un exemple"
<!--#4DBKVarSet/VT02*=7--> // VT02:=sous chaine(VT02;7) = "un exemple"
<!--#4DBKVar/VT02--> // affiche "un exemple"
<!--#4DBKVarSet/VT02=4DBKVar/VT01--> // VT02 prend la valeur de la
      VT01, "Voici un exemple"
<!--#4DBKVar/VT02--> // affiche "Voici un exemple"
<!--#4DBKVarSet/VT02*=7,2--> // VT02:=sous chaine(VT02;7;2) = "un"
<!--#4DBKVar/VT02--> // affiche "un"
```

4DBKVar

<!--#4DBKVar/variable[,format]-->

Paramètre	Description	Valeur(s)
variable	Nom de variable	• VL01 à VL100 : variables entier long (32 bits) • VR01 à VR100 : variables réelles (64 bits) • VT01 à VT100 : variables alphanumériques (30 Ko)
format	Format d'affichage valide	Voir chapitre Formats d'affichage et fonctions

Objectif : Retourner la valeur d'une variable.

Description

La commande 4DBKVar retourne la valeur courante d'une variable donnée. Cette commande permet par exemple d'utiliser la valeur courante de la variable d'une boucle dans un programme javascript.

Vous pouvez définir l'affichage ou la valeur de la variable à l'aide du paramètre *format*. Pour plus d'informations sur ce paramètre, reportez-vous au chapitre "Formats d'affichage et fonctions".

A l'aide de cette commande, vous pourrez construire des pages Web réellement dynamiques et ayant de la mémoire... en rappelant par exemple dans des formulaires les valeurs précédemment saisies.

Dans un lien, si vous souhaitez que la valeur soit retournée au moment du chargement de la page et non au moment où l'utilisateur clique sur le lien, utilisez l'instruction !4DBKVar.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemple

L'exemple suivant présente diverses utilisations des variables :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKVarSet/VL02=4--> //VL02:=4
<!--#4DBKVar/VL02--> //retourne 4 en place de la balise
<!--#4DBKVarSet/VT04=texte--> //VT04:="texte"
<!--#4DBKVar/VT04--> //retourne texte en place de la balise
```

14 Boucles

4DBKLoop, 4DBKEndLoop

<!--#4DBKLoop/varCompteur,début,fin-->
<!--#4DBKEndLoop/varCompteur-->

Paramètre	Description	Valeur(s)
varCompteur	Variable Entier long	VL01 à VL100
début	Valeur d'indice de début	Numérique
fin	Valeur d'indice de fin	Numérique

Objectif : Effectuer une boucle contrôlée par un compteur sur le code.

Description

La commande 4DBKLoop répète le code HTML situé entre les commandes 4DBKLoop et 4DBKEndLoop. La valeur de la variable *varCompteur* est initialisée par le paramètre *début*. Elle est incrémentée à chaque passage dans la boucle. La boucle s'arrête dès que la valeur de *varCompteur* égale *fin*.

Cette commande est utile pour générer des listes, des tableaux de données ou des objets de formulaire tels que des pop-up menus.

Les boucles sont imbricables sur 100 niveaux (un par variable VL).

Les valeurs d'indice *début* et *fin* peuvent être inversées pour effectuer une boucle décroissante (*début* > *fin*).
Ces valeurs peuvent être le résultat d'une fonction (4DBKVar ou 4DBKSelectionSize par exemple). Dans le cas de 4DBKVar, vous pourrez indiquer directement la variable sans l'instruction 4DBKVar (par exemple 4DBKLoop/VL01,1,VL02).

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemples

Cet exemple permet d'afficher une liste simple de 5 variables numériques :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKLoop/VL01,1,5--> //boucle(VL01;1;5)
* <!--#4DBKVar/VL01--> * <br> //afficher "*"+VL01+"*"+cr
<!--#4DBKEndLoop/VL01--> //fin de boucle
```

Cet exemple complet permet de générer et d'afficher un pop up menu des différentes catégories :

```
<select name="menu">
<option value="">Choisissez un élément...</option>
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1
<!--#4DBKQuerySet/all--> //rechercher tous les articles (tout sélectionner)
<!--#4DBKSelectionDistinct/categorie--> //filtre sur les 1ers ayant une caté-
        gorie unique
<!--#4DBKSortSet/produit--> //fixer le tri sur le champ produit
<!--#4DBKSelectionSort--> //trier la sélection numéro 1
<!--#4DBKRecordSet/1--> //1re fiche (début sélection)
<!--#4DBKLoop/VL01,1,4DBKSelectionSize--> //boucle(VL01;1;enregistre-
        ments trouvés)
<option value="<!--#4DBKField/categorie,U-->"> //retourne le champ en
        majuscules
<!--#4DBKField/categorie-->
</option> //retourne le champ
<!--#4DBKRecordNext--> //enregistrement suivant
<!--#4DBKEndLoop/VL01--> //fin de boucle
</select>
```

15 Conditions

4DBKIf
4DBKElse
4DBKEndIf

<!--#4DBKIf(condition)-->
<!--#4DBKElse-->
<!--#4DSEndIf-->

Paramètre	Description	Valeur(s)
condition	Expression booléenne	Toute expression valide retournant un booléen

Objectif : Définir du code conditionnel.

Description

La commande 4DBKIf est un opérateur de type “Si”. Cette commande structure les instructions de manière à permettre à 4D Business Kit, au moment de l’exécution du code, de choisir entre deux alternatives en fonction du résultat, True ou False, du paramètre *condition*.

Si *condition* retourne True, les instructions suivant la commande sont exécutées. Si *condition* retourne False, elles sont ignorées et les instructions suivant la commande 4DBKElse sont exécutées.
La commande 4DBKElse est optionnelle. Lorsqu’elle est omise, l’exécution reprend aux instructions suivant la commande 4DBKEndIf.

Les ensembles d'instructions 4DBKIf / 4DBKEndIf sont imbricables sur plusieurs niveaux.

Note : Lorsque le paramètre *condition* compare des chaînes de caractères, il est nécessaire d'encadrer chaque condition par des guillemets. Par exemple, l'instruction **<!--#4DBKIf(4DBKField/T01=montexte)-->** retourne une erreur. La syntaxe correcte est dans ce cas **<!--#4DBKIf("4DBKField/T01"="montexte")**. En revanche, cette syntaxe ne s'applique pas aux paramètres numériques, date, booléens, etc. Par exemple, la syntaxe de l'instruction **<!--#4DBKIf(4DBKField/VR02<3,00)** est correcte.

Conditions

Aucune.

Exemple

L'exemple suivant permet d'afficher des champs uniquement si plusieurs conditions sont réunies :

```
<!--#4DBKIf(4DBKRecordExists)--> //si la fiche courante existe
<!--#4DBKIf(4DBKField/prenom,F)--> //si le prénom existe
<!--#4DBKField/prenom-->*<!--#4DBKField/nom--> //afficher le prénom et
        le nom
<!--#4DBKElse--> //sinon
<!--#4DBKField/nom--> //n'afficher que le nom
<!--#4DBKEndIf--> //fin de si
<!--#4DBKElse--> //sinon
Cette fiche n'existe pas //indiquer que l'on a rien trouvé
<!--#4DBKEndIf--> //fin de si
```

4DBKCaseOf
4DBKCase
4DBKEndCaseOf

<!--#4DBKCaseOf-->
<!--#4DBKCase/condition-->
<!--#4DBKEndCaseOf-->

Paramètre	Description	Valeur(s)
condition	Expression booléenne	Toute expression valide retournant un booléen

Objectif : Définir du code conditionnel.

Description

La commande 4DBKCaseOf est un opérateur de type “Au cas ou”. Cette commande structure les instructions de manière à permettre à 4D Business Kit, au moment de l’exécution du code, de choisir entre plusieurs instructions. A la différence des commandes 4DBKIf 4DBKElse 4DBKEndIf, 4DBKCaseOf peut tester un nombre illimité de conditions booléennes à l’aide de la commande 4DBKCase. Les instructions correspondant à la première *condition* retournant Vrai sont exécutées.

Chaque test booléen est défini par une commande 4DBKCase. Si aucune condition ne retourne True, aucune instruction n’est exécutée, l’exécution reprend aux instructions suivant la commande 4DBKEndCaseOf.

Les ensembles d’instructions 4DBKCaseOf / 4DBKCase / 4DBKEndCaseOf sont imbricables sur plusieurs niveaux.

> *Note :* Lorsque le paramètre *condition* compare des chaînes de caractères, il est nécessaire d’encadrer chaque condition par des guillemets. Par exemple, l’instruction **<!--#4DBKCase/4DBKVar/VT04=FRF)-->** retourne une erreur. La syntaxe correcte est dans ce cas **<!--#4DBKCase/"4DBKVar/VT04"="FRF")**.
> En revanche, cette syntaxe ne s’applique pas aux paramètres numériques, date, booléens, etc. Par exemple, la syntaxe de l’instruction **<!--#4DBKCase/4DBKVar/VL04=12)** est correcte.

Conditions

Aucune.

Exemple

L'exemple suivant permet d'afficher un texte différent en fonction du nombre d'articles présents dans la sélection :

```
<!--#4DBKCaseOf--> //début du Au cas ou
<!--#4DBKCase/4DBKSelectionSize=0--> // (articles dans la sélection = 0)
Aucun article<br>trouvé
<!--#4DBKCase/4DBKSelectionSize=1--> // (articles dans la sélection = 1)
1 seul article<br>trouvé
<!--#4DBKCase/4DBKSelectionSize>1--> // (articles dans la sélection > 1)
<!--#4DBKSelectionSize--> articles<br>trouvés
<!--#4DBKEndCaseOf--> //fin de cas
<!--#4DBKCaseOf--> //Au cas ou
<!--#4DBKCase/4DBKVar/VL01=1-->c'est 1 //:(VL01=1) c'est un
<!--#4DBKCase/4DBKVar/VL01=2-->c'est 2 //:(VL01=2) c'est deux
<!--#4DBKCase/4DBKVar/VL01>2-->c'est <!--#4DBKVar/VL01--> //:(VL01>2)
        c'est 3, 4...5
<!--#4DBKEndCaseOf--> //fin de cas
```

16 Instructions élaborées

4DBKMenu

<!--#4DBKMenu/champ[[[,ligne],format],tri]-->

Paramètre	Description	Valeur(s)
champ	Nom ou Code de champ	• C01 à C03 pour les codes • T01 à T16 pour les textes • B01 à B16 pour les booléens • L01 à L05 pour les familles
ligne	Ligne de menu sélectionnée par défaut	Numérique
format	Format d'affichage valide	Voir chapitre Formats d'affichage et fonctions
tri	Tri du contenu du menu	> tri croissant (A vers Z) < tri décroissant (Z vers A)

Objectif : Générer les lignes d'un pop up menu.

Description

La commande 4DBKMenu permet d'afficher un pop up menu. Elle génère automatiquement les lignes <option> jusqu'à </option> pour une sélection donnée.

Cette commande remplace le tri et la boucle nécessaires pour coder toutes les lignes d'un pop up menu. Elle accélère sensiblement l'affichage des menus.

La sélection courante est triée selon le *champ*.

Il est possible de choisir le format d'affichage des éléments du menu au moyen du paramètre *format* (par exemple pour afficher les lignes en majuscules ou pour limiter l'affichage à un nombre défini de caractères ou de mots).

Le paramètre *tri* permet de trier le contenu du pop up menu affiché. L'un des caractères suivants doit être utilisé :

- **>** : le pop up menu est trié dans un ordre croissant (de A vers Z),
- **<** : le pop up menu est trié dans un ordre décroissant (de Z vers A).

Note : Si vous passez ce paramètre sans utiliser les paramètres optionnels précédents (*ligne* et *format*), vous devez saisir les virgules correspondantes. Par exemple : <!--#4DBKMenu/departement,,,>-->.

Si vous ne passez pas le paramètre *tri*, les lignes du menu seront triées suivant le tri courant de la sélection.

Conditions

Il faut avoir fixé la boutique courante à l'aide de l'instruction 4DBKStoreSet et la sélection courante à l'aide de l'instruction 4DBKSelectionSet.

Exemples

(1) L'exemple suivant...

```
<select name="menucategorie">
<!--#4DBKStoreSet/MCOM-->
<!--#4DBKSelectionSet/1-->
<!--#4DBKQuerySet/all-->
<!--#4DBKSelectionDistinct/categorie-->
<!--#4DBKSortSet/categorie-->
<!--#4DBKSelectionSort-->
<!--#4DBKRecordSet/1-->
<option value="">Choisissez une catégorie...</option>
<!--#4DBKLoop/VL01,1,4DBKSelectionSize-->
<option value="<!--#4DBKField/categorie,U&L-->" > <!--#4DBKField/catego-
        rie--> </option>
<!--#4DBKRecordNext-->
<!--#4DBKEndLoop/VL01-->
</select>
```

... peut être avantageusement remplacé par :

```
<select name="menucategorie">
<!--#4DBKStoreSet/MCOM-->
<!--#4DBKSelectionSet/1-->
<!--#4DBKQuerySet/all-->
<!--#4DBKSelectionDistinct/categorie-->
<option value="">Choisissez une catégorie...</option>
<!--#4DBKMenu/categorie-->
</select>
```

(2) Basé sur le précédent, cet exemple affiche un menu des catégories, la 3e ligne est sélectionnée par défaut et les éléments apparaissent en majuscules coupés à 10 caractères :

```
<select name="menucategorie">
<!--#4DBKStoreSet/MCOM-->
<!--#4DBKSelectionSet/1-->
<!--#4DBKQuerySet/all-->
<!--#4DBKSelectionDistinct/categorie-->
<!--#4DBKMenu/categorie,3,U&&10+-->
</select>
```

(3) L'exemple suivant permet d'afficher un pop up menu trié par ordre croissant :

```
<select name="menucategorie">
<!--#4DBKStoreSet/MCOM-->
<!--#4DBKSelectionSet/1-->
<!--#4DBKQuerySet/all-->
<!--#4DBKSelectionDistinct/categorie-->
<option value="">Choisissez une catégorie</option>
<!--#4DBKMenu/categorie,,,>-->   //Tri du menu dans un ordre croissant ; le
            tri courant de la sélection n'est pas modifié
</select>
```

4DBKMenuCountriesAll

<!--#4DBKMenuCountriesAll[/pays]-->

Paramètre	Description	Valeur(s)
pays	Ligne de menu sélectionnée par défaut	Chaîne du libellé du pays

Objectif : Générer les lignes d'un pop up menu affichant tous les pays du monde au format ISO.

Description

La commande 4DBKMenuCountriesAll permet d'afficher un pop up menu listant les pays du monde dans le format ISO. Elle génère automatiquement les lignes <option> jusqu'à </option> pour la liste des pays.

Cette commande est particulièrement utile pour la création des fiches client.

Conditions

Aucune.

Exemples

L'exemple suivant génère un pop up menu de tous les pays du monde sans positionnement initial :

```
<select name="Country">
<!--#4DBKMenuCountriesAll-->
</select>
```

L'exemple suivant génère un pop up menu de tous les pays du monde positionné par défaut sur la France :

```
<select name="Country">
<!--#4DBKMenuCountriesAll/France-->
</select>
```

L'exemple suivant génère un pop up menu de tous les pays du monde positionné par défaut sur le pays du client identifié :

```
<select name="Country">
<!--#4DBKMenuCountriesAll/4DBKField/CusCountryMain-->
</select>
```

17 Préférences

4DBKPrefsSet

<!--#4DBKPrefsSet/pref=valeur-->

Paramètre	Description	Valeur(s)
pref	Type de préférence à définir	• AutoSort, ou • WebLanguage, ou • DataLanguage, ou • CusErrorMode, ou • CusErrorDuplicateAdd, ou • CusErrorDuplicateUpd, ou • CusErrorLocked
=valeur	Valeur de la préférence	• yes ou no (AutoSort) • code de langue (WebLanguage et DataLanguage) • PAGE ou POP (CusErrorMode) • Nom de page (CusErrorDuplicateAdd, CusErrorDuplicateUpd, et CusErrorLocked)

Objectif : Définir un paramètre global à l'ensemble de la boutique.

Description

La commande 4DBKPrefsSet permet de modifier le fonctionnement global du serveur 4D Business Kit pour la session courante. Vous passez dans le paramètre *pref* une préférence à définir et dans le paramètre *=valeur* la nouvelle valeur de cette préférence.

- **AutoSort**
  Ce paramètre permet d'optimiser l'utilisation de sélections contenant un grand nombre d'enregistrements.

Une fois qu'un critère de tri a été défini au moyen de la commande 4DBKSortSet, toute instruction 4DBKSelectionSet suivante provoquera le tri de la sélection selon le critère établi si AutoSort est à yes. Si AutoSort est à no, la sélection ne sera pas triée ou retriée. Il reste alors possible de la trier à l'aide de la commande 4DBKSelectionSort.

- **WebLanguage**
  Ce paramètre indique au serveur 4D Business Kit l'emplacement des pages Web à servir. Par défaut, le dossier utilisé est le dossier "WebPagesXX", où XX représente la langue principale de la boutique. Si vous exécutez par exemple l'instruction <!--#4DBKPrefsSet/WebLanguage=DE-->, 4D Business Kit servira les pages présentes dans le dossier "WebPagesDE", quelle que soit la langue principale de la boutique.

- **DataLanguage**
  Ce paramètre indique au serveur 4D Business Kit la langue des données à servir. Par défaut, 4D Business Kit utilise la langue principale de la boutique. Si la boutique a été définie en deux langues (FR et US par exemple) et si les données ont été saisies dans les deux langues, après l'exécution de l'instruction <!--#4DBKPrefsSet/DataLanguage=FR--> le serveur utilisera la partie française des données à chaque appel de la commande 4DBKField. Si un champ n'est pas traduit, la langue principale de la boutique est utilisée.

- **CusErrorMode**, **CusErrorDuplicateAdd**, **CusErrorDuplicateUpd** et **CusErrorLocked**
  Lors de l'utilisation de l'instruction 4DBKFieldSet/CusCode, des erreurs peuvent survenir lorsque :
  • l'utilisateur essaye de créer un compte client qui existe déjà,
  • l'utilisateur essaye de changer son code alors qu'un autre client possède déjà ce code,
  • la fiche du client est verrouillée.
  Ces erreurs sont gérées automatiquement par 4D Business Kit mais il est possible de changer la façon dont elle sont présentées à l'utilisateur, à l'aide de la commande 4DBKPrefsSet :

  • <!--#4DBKPrefsSet/CusErrorMode=PAGE--> l'erreur sera affichée dans une nouvelle page.
  • <!--#4DBKPrefsSet/CusErrorMode=POP-->
  l'erreur sera affichée dans une fenêtre popup (par défaut).

• <!--#4DBKPrefsSet/CusErrorDuplicateAdd=page_erreur.htm-->
page_erreur.htm est la page qui sera affichée si l'utilisateur essaye de créer un compte client qui existe déjà.

• <!--#4DBKPrefsSet/CusErrorDuplicateUpd=page_erreur.htm-->
page_erreur.htm est la page qui sera affichée si l'utilisateur essaye de changer son code alors qu'un autre client possède déjà ce code.

• <!--#4DBKPrefsSet/CusErrorLocked=page_erreur.htm-->
page_erreur.htm est la page qui sera affichée si la fiche client est verrouillée (par exemple lorsqu'elle est en cours de modification sur le serveur 4D Business Kit).

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemple

Cet exemple active le tri automatique :

```
<!--#4DBKStoreSet/MCOM--> //passe sur la boutique MCOM
<!--#4DBKPrefsSet/AutoSort=yes--> //active le tri automatique pour chaque
        4DBKSelectionSet
```

18 Utilitaires

4DBKExecute

4DBKExecute:Commande1[;Commande2[;..;CommandeN]]

Paramètre	Description	Valeur(s)
Commande1...N	Instruction 4D Business Kit	Toute instruction 4D Business Kit valide (commande+paramètres)

Objectif : Exécuter une série de commandes.

Description

La commande 4DBKExecute permet d'exécuter une ou plusieurs instructions 4D Business Kit en un seul appel. Cette commande doit être placée dans un lien ou dans l'action d'un formulaire.

> *Note :* Les instructions contenues dans un 4DBKExecute ne sont traitées (calculées) qu'au moment où l'utilisateur clique sur le lien. Si vous souhaitez qu'une de ces instructions soit traitée au moment du chargement de la page contenant le lien, il vous suffit de faire précéder l'instruction du caractère "!"

Une fois la suite d'instructions exécutée, le navigateur réaffichera la page courante si vous n'avez pas utilisé la commande 4DBKGo à la fin du 4DBKExecute.

Cette commande n'étant pas liée à une boutique, il vous sera possible d'effectuer des traitements sur plusieurs boutiques.

Conditions

Aucune.

Exemple

L'exemple suivant permet à l'utilisateur, en cliquant sur un lien, d'avancer le curseur de deux fiches au sein d'une sélection et de recharger la page courante.

```
<a href="4DBKExecute:4DBKStoreSet/DEM1;4DBKSelectionSet/1;
        4DBKCursorNext/2;4DBKNoCache">Lien</a>
```

4DBKMail

<!--#4DBKMail/de,vers,sujet,page[,format]-->

Paramètre	Description	Valeur(s)
de	Adresse e-mail de l'émetteur	Chaîne de caractères
vers	Adresse e-mail du destinataire	Chaîne de caractères
sujet	Sujet du message	Chaîne de caractères
page	Page HTML à envoyer	Chemin HTML de la page
format	Format d'envoi du message	htm ou txt

Objectif : Envoyer une page HTML par courrier électronique.

Description

La commande 4DBKMail envoie une *page* Web par courrier électronique au destinataire dont l'adresse est indiquée dans le paramètre *vers*, sous l'identité de l'émetteur *de*, avec le *sujet* défini.

Les paramètres *de* et *vers* peuvent être le résultat de fonctions 4D Business Kit.

Cette commande permet d'envoyer une page de catalogue, une demande de renseignements ou une confirmation de commande.

> *Note :* La page Web peut contenir du code 4D Business Kit mais pas de code 4D.

Le paramètre *format* permet de préciser le format d'envoi du message :

- Si vous passez la valeur htm, la page sera envoyée au format html (multipart).
- Si vous passez la valeur txt, la page sera envoyée au format texte seul (plain text).

Par défaut (si le paramètre *format* est omis), le message est envoyé au format html.

Si la page est envoyée au format html, les images présentes sont encodées et incluses dans le message lui-même, permettant ainsi une déconnexion totale du serveur lors de la lecture du message et la possibilité de l'archiver.

Pour l'envoi du courrier, la commande utilise le serveur SMTP défini dans la fenêtre de paramétrage de la boutique (page **Courrier**).

Si l'envoi du message a échoué, la commande retourne l'erreur [err=MSG_NTX].

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemple

L'exemple suivant permet d'envoyer par E-mail à un client une page HTML reprenant une commande passée. Le message est envoyé au format html :

```
<!--#4DBKStoreSet/DEM1-->
<!--#4DBKMail/order@shop.com,customer@thenet.net,votre comman-
        de,orderform.htm,htm-->
```

4DBKGo

<!--#4DBKGo/page[,direct[,type]]-->

Paramètre	Description	Valeur(s)
page	Page HTML à afficher ou URL complet	Chemin d'accès HTML
direct	Option d'envoi direct de la page	YES ou NO
type	Type MIME de l'envoi direct	Chaîne de caractères

Objectif : Aller à une page HTML.

Description

La commande 4DBKGo provoque l'envoi de la *page* HTML définie. Le paramètre *page* peut désigner une page de votre site ou une page extérieure au site ; pour ce dernier cas, il suffit de passer dans le paramètre l'URL complet à atteindre.

Cette commande permet d'afficher une page donnée en passant par un lien ou lors de la validation d'un formulaire.

Note : Il n'est pas nécessaire de faire suivre cette instruction d'un 4DBKNoCache.

Le paramètre *direct* vous permet d'indiquer si vous souhaitez que le serveur 4D Business Kit envoie la page en mode direct : passez YES pour l'envoyer directement et NO sinon (mode standard, utilisé par défaut). Dans les deux cas bien entendu, le serveur analyse le contenu des pages avant de les envoyer.
L'envoi direct permet une meilleure compatibilité des liens avec les plug-ins d'animation Web tels que Flash® ou ShockWave® pour l'échange de variables par exemple.

En mode direct, vous avez également la possibilité de définir le *type* MIME de l'envoi. Par défaut, si le paramètre *type* est omis, le type text/html est utilisé.

Conditions

Aucune.

Exemples

(1) L'exemple suivant permet à l'utilisateur, en cliquant sur un lien, d'avancer le curseur de 2 fiches et d'aller à la page d'accueil :

```
<a href="4DBKExecute:4DBKStoreSet/DEM1;4DBKSelectionSet/1;
        4DBKCursorNext/2;4DBKGo/home.htm">Lien</a>
```

(2) L'exemple suivant représente une ligne de commande d'un script Flash® ou ShockWave® :

```
LoadVariables("http://www.mystore.com/4daction/4DBKExecute/
        4DBKStoreSet/DEM1;4DBKGo/result.txt,yes,text/plain")
```

(3) Dans l'exemple suivant, différentes opérations sont effectuées dans la boutique par l'intermédiaire d'un lien vers un site externe :

```
<a href="4DBKExecute:4DBKStoreSet/TEST;4DBKSelectionSet/1;
        4DBKRecordSet/!4DBKRecord;4DBKDetailSet/!4DBKField/C01;
        4DBKGo/http://www.4d.fr">
```

(4) Dans cet exemple, la valeur de la variable VT01 a été préalablement définie dans la page :

```
<a href="4DBKExecute:4DBKStoreSet/TEST;...;4DBKGo/4DBKVar/VT01">
```

(5) Voici un exemple de saut vers une page d'un site externe (notez la présence du tiret - à la fin de l'URL, permettant de conserver le lien intact) :

```
<a href="4DBKExecute:4DBKStoreSet/TEST;4DBKGo/http://www.foxnews
        .com/business/index.html-">
```

(6) Voici un exemple de passage de paramètres à une autre page depuis un lien :

```
<a href="4DBKExecute:4DBKStoreSet/TEST;4DBKGo/urlparms.htm?var1
        =parm1&var2=parm2">
```

Il sera possible de récupérer les paramètres *parm1* et *parm2* dans la page urlparms.htm en utilisant la commande 4DBKURLParms.

(7) Il peut être également intéressant d'utiliser le passage de paramètres dans un 4DBKGo non pas pour exploiter ces paramètres, mais pour laisser une trace plus précise dans le fichier *logweb.txt*. Il sera par exemple possible, à l'aide d'outils comme WebTrend ou FunnelWeb, de savoir combien de fois chaque fiche article a pu être consultée avec le lien suivant en analysant les requêtes qui contiennent ?itm= :

```
<a href="4DBKExecute:4DBKStoreSet/TEST;4DBKSelectionSet/1;
        4DBKRecordSet/!4DBKRecord;4DBKDetailSet/!4DBKField/C01;
        4DBKGo/detail.htm?itm=!4DBKField/C01">
```

4DBKURLParms

<!--#4DBKURLParms[/indice,séparateur[,format]]-->

Paramètres	Description	Valeur(s)
indice	Indice du paramètre à récupérer	Numérique
séparateur	Séparateur de la liste de paramètres	Chaîne de caractères
format	Format d'affichage valide	Voir chapitre Formats d'affichage et fonctions

Objectif : Retourner les paramètres de l'URL envoyé à une page HTML.

Description

La commande 4DBKURLParms retourne les paramètres de l'URL envoyé à la page dans laquelle elle est appelée.

Les paramètres d'un URL correspondent à sa partie variable, située à droite du caractère **?** (point d'interrogation).

Par exemple, dans l'URL suivant :
http://www.4d.com/product.html?=2$5$6&var2=1&var3=x&date=0
... la chaîne "=2$5$6&var2=1&var3=x&date=0" représente les paramètres de l'URL.

- Si vous ne passez aucun paramètre à 4DBKURLParms, tous les paramètres de l'URL sont retournés.
- Si vous passez les paramètres *indice* et *séparateur*, vous indiquez à 4D Business Kit de retourner le Nième paramètre de l'URL parmi la liste des paramètres délimités par séparateur.
- En outre, vous pouvez passer un format d'affichage permettant d'appliquer un traitement supplémentaire à la chaîne retournée.

Conditions

Aucune.

Exemple

Dans le cas où la page HTML dans laquelle se trouve la commande 4DBKURLParms reçoit l'URL suivant :
http://www.4d.com/product.html?=2$5$6&var2=1&var3=x&date=0

```
<!--#4DBKURLParms-->
        //retourne "=2$5$6&var2=1&var3=x&date=0"
<!--#4DBKURLParms/1,&-->
        //retourne "=2$5$6"
<!--#4DBKURLParms/2,&-->
        //retourne "var2=1"
<!--#4DBKURLParms/3,&-->
        //retourne "var3=x"
```

Un filtrage supplémentaire peut être effectué sur les paramètres d'URL. Par exemple :

```
<!--#4DBKURLParms/1,&,LST&2&$-->
        // retourne le 2e élément de la liste "2$5$6" (dont le séparateur est
           $), c'est-à-dire 5
```

4DBKNoCache

<!--#4DBKNoCache-->
Cette commande ne requiert pas de paramètre.

Objectif : Retourner un code aléatoire.

Description

La commande 4DBKNoCache retourne un code aléatoire. Cette commande doit être placée dans les liens pour éviter que les pages correspondantes soient lues dans le cache.

En effet, lorsque l'utilisateur clique sur un lien pour afficher une page, celle-ci est chargée dans le cache du navigateur. Ce fonctionnement permet d'accélérer les accès aux pages statiques car, lors des consultations ultérieures de la même page à l'aide du lien, la page est directement lue dans le cache.
Avec les pages dynamiques, ce fonctionnement peut s'avérer problématique car les pages stockées dans le cache correspondent au contexte dans lequel elles ont été générées. Par exemple, une page result.htm appelée par un lien "Liste des fiches trouvées" n'aura pas le même contenu en fonction du contexte.

Pour éviter que dans cette page puisse être réutilisée hors de son contexte, la commande 4DBKNoCache permet d'insérer un code aléatoire dans l'URL de la page, la rendant ainsi unique. En cas de réactivation du lien dans un contexte différent, le navigateur ira directement charger la page du serveur, car aucune page du même nom n'existera dans le cache.

Conditions

Aucune.

Exemple

L'exemple suivant retourne un code aléatoire. Il doit être placé dans un lien :

```
<!--#4DBKNoCache--> //retourne NC12372 (par exemple) en place de la balise
```

4DBKInclude

<!--#4DBKInclude/page-->

Paramètre	Description	Valeur(s)
page	Nom de la page HTML à inclure	Chemin d'accès HTML

Objectif : Inclure une page HTML à l'intérieur d'une autre page.

Description

La commande 4DBKInclude inclut une *page* Web à l'emplacement de la balise. Cette commande permet notamment d'utiliser des bandeaux de navigation sans répéter le code HTML dans chaque page.

Note : Une page incluse par une commande 4DBKInclude peut elle-même contenir des commandes 4DBKInclude et ainsi de suite.

Conditions

Aucune.

Exemple

L'exemple suivant insère dans une page Web une bannière :

```
<!--#4DBKInclude/bandeau.htm--> //inclut le code de la page bandeau.htm
                                à la place de la balise
```

4DBKTraceOnPage

<!--#4DBKTraceOnPage[type][début]-->

Paramètre	Description	Valeur(s)
type	Type de variable à afficher	T=Variables texte R=Variables réel L=Variables entier long (défaut)
début	Début de l'intervalle de valeurs	Numérique

Objectif : Afficher une fenêtre de trace au-dessus de la page HTML courante.

Description

La commande 4DBKTraceOnPage provoque l'affichage d'une fenêtre de trace (débogage) au-dessus de la fenêtre de la page HTML courante, dans le navigateur de l'utilisateur.

Cette commande est utile pour déboguer à distance le code HTML inséré à l'endroit où a été placée la balise. La fenêtre de trace affiche les valeurs courantes des éléments utilisés dans le code.

La fenêtre peut afficher simultanément 10 valeurs de variables. Par défaut, les 10 premières variables Entier long sont affichées (cf. "Fenêtre de trace" page 29).
Vous pouvez cependant afficher les 10 premières variables de type texte (passez T dans le paramètre *type*) ou les 10 premières variables de type réel (passez L dans le paramètre *type*).
Pour afficher un autre intervalle de 10 variables (par exemple les variables texte 20 à 29), passez le numéro de la première variable dans le paramètre *début*.

Conditions

Aucune.

Exemples

(1) L'exemple suivant affiche dans le navigateur une fenêtre listant les paramètres d'exécution courants ainsi que les 10 premières variables Entier long :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1
<!--#4DBKCursorNext/2--> //avancer le curseur de 2 fiches
<!--#4DBKTraceOnPage--> //afficher une fenêtre avec les paramètres courants
```

(2) L'exemple suivant affiche la fenêtre de trace avec les 10 premières variables Texte :

```
<!--#4DBKTraceOnPageT-->
```

(3) L'exemple suivant affiche la fenêtre de trace avec les variables réelles R12 à R21 :

```
<!--#4DBKTraceOnPageR12-->
```

(4) L'exemple suivant affiche la fenêtre de trace avec les variables Entier long L43 à L54 :

```
<!--#4DBKTraceOnPage43-->
```

Référence : 4DBKTraceOnServer.

4DBKTraceOnServer

<!--#4DBKTraceOnServer-->
Cette commande ne requiert pas de paramètre.

Objectif : Afficher une fenêtre de trace sur le serveur 4D Business Kit.

Description

La commande 4DBKTraceOnServer provoque l'affichage d'une fenêtre de trace (débogage) sur le serveur 4D Business Kit.

Cette commande est utile pour déboguer le code HTML inséré à l'endroit où a été placée la balise. La fenêtre de trace affiche les valeurs courantes des éléments utilisés dans le code.

Conditions

Aucune.

Exemple

L'exemple suivant affiche sur le serveur une fenêtre listant les paramètres d'exécution courants :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1
<!--#4DBKCursorNext/2--> //avancer le curseur de 2 fiches
<!--#4DBKTraceOnServer--> //afficher une fenêtre avec les paramètres courants sur le poste serveur
```

Référence : 4DBKTraceOnPage

4DBKHistoryOnPage

<!--#4DBKHistoryOnPage-->
Cette commande ne requiert pas de paramètre.

Objectif : Afficher une fenêtre des instructions traitées au-dessus de la page HTML courante.

Description

La commande 4DBKHistoryOnPage provoque l'affichage d'une fenêtre des instructions traitées au-dessus de la fenêtre de la page HTML courante, dans le navigateur de l'utilisateur.

Cette commande est utile pour déboguer à distance le code HTML inséré à l'endroit où a été placée la balise.

Conditions

Aucune.

Exemple

L'exemple suivant affiche dans une fenêtre du navigateur les lignes d'instructions en cours d'exécution :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1
<!--#4DBKCursorNext/2--> //avancer le curseur de 2 fiches
<!--#4DBKHistoryOnPage-> //afficher une fenêtre avec les instructions
        traitées
```

4DBKHistoryOnServer

<!--#4DBKHistoryOnServer-->
Cette commande ne requiert pas de paramètre.

Objectif : Afficher sur le serveur une fenêtre des instructions traitées.

Description

La commande 4DBKHistoryOnServer provoque l'affichage d'une fenêtre des instructions traitées sur le poste serveur.

Cette commande est utile pour déboguer le code HTML inséré à l'endroit où a été placée la balise.

Conditions

Aucune.

Exemple

L'exemple suivant affiche sur le serveur les lignes d'instructions en cours d'exécution :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKSelectionSet/1--> //passer sur la sélection numéro 1
<!--#4DBKCursorNext/2--> //avancer le curseur de 2 fiches
<!--#4DBKHistoryOnServer--> //afficher une fenêtre avec les instructions
        traitées
```

4DBK//

<!--#4DBK// commentaire-->

Paramètre	Description	Valeur(s)
commentaire	Commentaire libre	Chaîne de caractères

Objectif : Insérer des commentaires dans le code HTML.

Description

La commande 4DBK// définit un commentaire qui disparaît au moment où la page est servie.

Cette commande permet de commenter des instructions 4D Business Kit qui ne devront pas êtres vues par l'utilisateur.

Ces commentaires peuvent être aussi longs que souhaités et comporter plusieurs lignes de texte.

Conditions

Aucune.

Exemple

L'exemple suivant ajoute des commentaires aux instructions.

```
<!--#4DBKStoreSet/MCOM--> <!--#4DBK// passer sur la boutique MCOM-->
<!--#4DBKSelectionSet/1--> <!--#4DBK// passer sur la sélection numéro 1-->
```

4DBKProcessingTime

<!--#4DBKProcessingTime-->
Cette commande ne requiert pas de paramètre.

Objectif : Retourner le temps de traitement de la page.

Description

La commande 4DBKProcessingTime retourne le temps de traitement de la page. Cette commande vous permet par exemple d'analyser les charges.

Note : Le temps de traitement est retourné en millisecondes.

4DBKBrowserLanguage

<!--#4DBKBrowserLanguage-->
Cette commande ne requiert pas de paramètre.

Objectif : Retourner la langue du navigateur.

Description

La commande 4DBKBrowserLanguage retourne la langue choisie dans les préférences du navigateur. Elle vous permet de déterminer dès la page d'accueil l'affichage d'un site dans une langue ou dans une autre.

Conditions

Aucune.

Exemple

Voir la page *misc.htm* dans le site TEST.

4DBKScoreServer

<!--#4DBKScoreServer[,format]-->

Paramètre	Description	Valeur(s)
format	Format d'affichage valide	Voir chapitre Formats d'affichage et fonctions

Objectif : Retourner le nombre de pages servies.

Description

La commande 4DBKScoreServer retourne le nombre de pages servies par le serveur, toutes boutiques confondues et depuis la 1re mise en service du serveur.

Il s'agit du nombre réel de pages servies, quels que soient leur complexité et le nombre d'appels différents à 4D Business Kit qu'elles contiennent.

La valeur retournée peut être formatée à l'aide du paramètre *format*.

Conditions

Aucune.

Exemple

L'exemple suivant permet de retourner le nombre de pages servies par le serveur :

```
<!--#4DBKcoreServer,### ### ##0--> pages
```

4DBKScoreStore

<!--#4DBKScoreStore/année[,mois[,jour[,format]]]-->

Paramètre	Description	Valeur(s)
année	Année d'analyse du journal des connexions	Numérique (4 chiffres)
mois	Mois d'analyse du journal des connexions (*année* doit avoir été spécifiée)	Numérique (1 ou 2 chiffres)
jour	Jour d'analyse du journal des connexions (*année* et *mois* doivent avoir été spécifiés)	Numérique (1 ou 2 chiffres)
format	Format d'affichage valide	Voir chapitre Formats d'affichage et fonctions

Objectif : Retourner le nombre de pages servies pour une boutique et une période données.

Description

La commande 4DBKScoreStore retourne le nombre de pages servies par le serveur 4D Business Kit pour la boutique courante et ce, pour la période définie par les paramètres *année*, *mois* et *jour*.

Les paramètres *année*, *mois* et *jour* peuvent être le résultat de fonctions 4D Business Kit.

Il s'agit du nombre réel de pages servies, quels que soient leur complexité et le nombre d'appels différents à 4D Business Kit qu'elles contiennent.

> *Note :* Cette commande effectuant diverses recherches dans la base de données, il n'est pas conseillé d'y faire appel dans la page d'accueil du site mais plutôt sur une page dédiée à la consultation des connexions.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemple

L'exemple suivant retourne le nombre de pages servies pour la boutique MCOM en avril 2002 :

```
<!--#4DBKStoreSet/MCOM--> //passer sur la boutique MCOM
<!--#4DBKScoreStore,2002,4,### ### ##0--> pages
```

4DBKToday

<!--#4DBKToday[,format]-->

Paramètre	Description	Valeur(s)
format	Format d'affichage valide	Voir chapitre Formats d'affichage et fonctions

Objectif : Retourner la date du jour.

Description

La commande 4DBKToday retourne la date du jour telle que définie sur le poste serveur, dans le format indiqué par le paramètre *format*.

Exemple

L'exemple suivant permet d'afficher la date du jour :

```
<!--#4DBKToday,yyyy-mm-dd--> //retourne la date au format 2002-12-14
```

4DBKDateOffSet

<!--#4DBKDateOffSet/date,nbAnnées[,nbMois[,nbJours[,format]]]-->

Paramètres	Description	Valeur(s)
date	Date à transformer	Champ ou variable de type date
nbAnnées	Nombre d'années	Valeur numérique
nbMois	Nombre de mois	Valeur numérique
nbJours	Nombre de jours	Valeur numérique
format	Format d'affichage valide	Voir chapitre Formats d'affichage et fonctions

Objectif : Ajouter ou enlever un certain nombre d'années, de mois et/ou de jours à la date passée en paramètre.

Description

La commande 4DBKDateOffSet retourne la *date* passée en paramètre augmentée ou diminuée du nombre d'années, de mois et/ou de jours définis dans les paramètres correspondants.

Passez 0 dans *nbAnnées* et *nbMois* si vous souhaitez ajouter uniquement des jours.

Passez une valeur négative dans les paramètres *nbAnnées*, *nbMois* et *nbJours* si vous souhaitez retrancher des durées.

Le paramètre *format* vous permet de définir le format d'affichage de la date résultante.

Exemples

(1) L'exemple suivant retourne la date du jour (4DBKToday) + 45 jours :

```
<!--#4DBKDateOffSet/4DBKToday,0,0,45,dd mm yyyy-->
```

(2) Cet exemple retourne la date du 18 avril 1966 + 35 ans et 4 mois, soit le 18/08/01 :

```
<!--#4DBKDateOffSet/18/04/66,35,4,0,dd/mm/yy-->
```

(3) Cet exemple retourne le dernier jour de l'année précédente :

```
<!--#4DBKVarSet/VT01=31/12/-->
<!--#4DBKVarSet/VT01+=4DBKToday,yyyy-->
<!--#4DBKDateOffSet/4DBKVar/VT01,-1,0,0,dd/mm/yy-->
```

4DBKNow

<!--#4DBKNow[,format]-->

Paramètre	Description	Valeur(s)
format	Format d'affichage d'heure valide	Voir chapitre Formats d'affichage et fonctions

Objectif : Retourner l'heure courante.

Description

La commande 4DBKNow retourne l'heure courante du serveur 4D Business Kit (sur 24 heures) dans le format indiqué par le paramètre *format*.

Si vous ne passez pas le paramètre *format*, l'heure est retournée sous la forme "HH:MM:SS" (par exemple, 17:35:24).

Vous pouvez également définir le format d'affichage de l'heure en passant dans *format* les caractères hh,mm et ss sous toutes les combinaisons. Il est possible d'insérer du texte ("heures", "minutes", etc.) dans le format.

Exemple

Si l'heure est 18:28:35 :

```
<!--#4DBKNow--> //retourne 18:28:35
<!--#4DBKNow, hh heures mm minutes --> //retourne 18 heures 28 minutes
```

4DBKTimeOffSet

<!--#4DBKTimeOffSet/heure,nbHeures[,nbMin[,nbSec[,format]]]-->

Paramètre	Description	Valeur(s)
heure	Heure à transformer	Valeur de type heure
nbHeures	Nombre d'heures	Valeur numérique
nbMin	Nombre de minutes	Valeur numérique
nbSec	Nombre de secondes	Valeur numérique
format	Format d'affichage d'heure valide	Voir chapitre Formats d'affichage et fonctions

Objectif : Ajouter ou enlever un certain nombre d'heures, de minutes et/ou de secondes à l'heure passée en paramètre.

Description

La commande 4DBKTimeOffSet retourne l'*heure* passée en paramètre augmentée ou diminuée du nombre d'heures, de minutes et/ou de secondes définis dans les paramètres correspondants.

Passez 0 dans *nbHeures* et *nbMin* si vous souhaitez ajouter uniquement des secondes.

Passez une valeur négative dans les paramètres *nbHeures, nbMin* et *nbSec* si vous souhaitez retrancher des durées.

Le paramètre *format* permet de définir le format d'affichage de l'heure résultante. Si vous ne passez pas ce paramètre, l'heure est retournée sous la forme "HH:MM:SS" (par exemple, 17:35:24).

Exemples

(1) L'exemple suivant ajoute trois heures à l'heure courante :

```
<!--#4DBKTimeOffSet/4DBKNow,3-->
```

(2) Cet exemple retranche 2 heures, 20 minutes et 30 secondes à l'heure courante :

```
<!--#4DBKTimeOffSet/4DBKNow,-2,-20,-30,hh heures mm minutes ss
        secondes-->
```

4DBKHttpPostProcess

<form method="POST" action="/4daction/4DBKHttpPostProcess/page">

Paramètre	Description	Valeur(s)
page	Page à afficher après validation	Chaîne de caractères

Objectif : Traiter les champs saisis dans un formulaire après sa validation.

Description

Cette commande permet de “poster” tout formulaire contenant des champs 4D Business Kit. Après validation du formulaire, 4D Business Kit effectue une redirection sur la page html définie par le paramètre *page* puis exécute les balises 4DBK présentes dans cette page.

A l’aide de la commande 4DBKHttpPostParms, il est possible de lire la valeur de tous les champs présents dans le formulaire et ainsi d’effectuer des tests ou des affectations dans la *page*. Typiquement, la page peut être terminée par une instruction 4DBKGo pour rediriger l’utilisateur vers une autre page html, de sorte qu’il aura l’impression de passer directement du formulaire à cette page.

Pour tout formulaire, il est souhaitable d’inclure dans l’en-tête <form> un évenement Javascript onSubmit qui permet de refuser ou de filtrer certaines valeurs saisies dans les champs.

Exemples

(1) Reportez-vous, dans le site TEST, aux pages *account_free_post.htm* et *account_free_post2.htm*.

(2) Cet exemple permet à l’utilisateur de valider des paramètres :

```
<form name="myForm" enctype="application/x-www-form-urlencoded"
      method="POST" action="/4daction/4DBKHttpPostProcess/
      myPage.htm" onsubmit="return CheckmyForm(document.
      myForm);">
```

Dans la page *myPage.htm* :

```
<!--#4DBKStoreSet/TEST-->
<!--#4DBKFieldSet/CusCode=4DBKHttpPostParms/CusCode-->
<!--#4DBKFieldSet/CusPassword=4DBKHttpPostParms/CusPassword-->
<!--#4DBKFieldSet/CusPasswordInfo=4DBKHttpPostParms/
        CusPasswordInfo-->
<!--#4DBKGo/home.htm-->
```

Référence : 4DBKHttpPostParms

4DBKHttpPostParms

<!--#4DBKHttpPostParms/nomChamp-->

Paramètre	Description	Valeur(s)
nomChamp	Nom du champ du formulaire dont il faut récupérer la valeur	Chaîne de caractères

Objectif : Lire la valeur d'un champ passé dans un HTTP POST.

Description

Après validation d'un formulaire par un HTTP POST à l'aide de la commande 4DBKHttpPostProcess, 4D Business Kit stocke les valeurs saisies dans les champs du formulaire. 4DBKHttpPostParms permet de récupérer ces valeurs au sein d'une autre page Web.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemples

(1) Reportez-vous, dans le site TEST, aux pages *account_free_post.htm* et *account_free_post2.htm*.

(2) Cet exemple récupère la valeur d'un champ :

```
<!--#4DBKHttpPostParms/phone--> // récupère la valeur du champ code.
```

(3) Cet exemple récupère et réaffecte la valeur d'un champ :

```
<!--#4DBKFieldSet/CusPhone=4DBKHttpPostParms/phone--> // récupère la
        valeur du champ phone et l'affecte au champ téléphone du client
        identifié.
```

Référence : 4DBKHttpPostProcess.

4DBKHttpPostCallBack

http://serveur/4daction/4DBKHttpPostCallBack/champSession/page/debug

Paramètre	Description	Valeur(s)
champSession	Nom du champ dans lequel l'identifiant de la session a été stocké	Nom de champ
page	Nom de la page appelée	Chaîne de caractères
debug	Créer un fichier de résultat	0 : Ne pas créer de fichier 1 : Créer un fichier

Objectif : Définir le point de rappel d'une passerelle de paiement vers le serveur 4D Business Kit.

Description

Si votre boutique délègue la gestion des paiements à une passerelle de paiement, l'utilisateur sera redirigé du site servi par 4D Business Kit vers une page Web de cette passerelle de paiement[1]. Là, il pourra saisir, par exemple, les coordonnées de sa carte bancaire.

A cet instant, la passerelle de paiement émettra une requête de type Http POST vers le serveur 4D Business Kit. Celui-ci lira la valeur du champ *champSession* et rouvrira la session dans laquelle l'utilisateur naviguait avant le saut vers la passerelle de paiement. Il chargera ensuite la page Web *page* pour en exécuter les instructions 4DBK (par exemple mettre à jour l'adresse de livraison, envoyer des E-mails, valider la commande, etc.).

> *Note :* La page est chargée en mémoire puis toutes les instructions 4DBK sont exécutées séquentiellement comme dans le cas d'une page Web ordinaire. La dernière instruction doit être une réponse à la passerelle de paiement expédiée par 4DBKHttpPostResponse (par exemple <!--#4DBKHttpPostResponse/OK-->). Il n'est pas possible

1. L'appel initial à la passerelle de paiement se fait souvent par la validation d'un formulaire invisible de type Http POST ou par un url paramétré de type Http GET. C'est à ce moment que le serveur 4D Business Kit transmet à la passerelle de paiement l'identifiant de la session que la passerelle de paiement lui retournera lors de la validation du paiement. Dans le cas du Http POST, le champ peut être rempli de la façon suivante :
<input type="hidden" name="champ_de_session" value="<!--#4DBKField/SessionID,X-->">

d'inclure dans cette page du code javascript car elle ne circule pas alors par le navigateur de l'utilisateur. Les champs transmis par la passerelle de paiement peuvent être lus au sein de cette page par la commande 4DBKHttpPostParms.

L'utilisateur, qui se trouve à ce moment-là toujours sur la passerelle de paiement, se verra proposer de revenir à la boutique, en général via un bouton.

Lorsque le paramètre *debug* est à 1, un fichier de résultat est créé dans le dossier [Logs] de la boutique, ce qui permet au développeur de visualiser les champs appelés au sein de la *page*.

Exemple

Reportez-vous, dans le site TEST, aux pages *payment_yellowpay.htm* et *payment_yellowpay_cb.htm*.

Référence : 4DBKHttpPostResponse, 4DBKHttpPostParms.

4DBKHttpPostResponse

<!--#4DBKHttpPostResponse/données-->

Paramètre	Description	Valeur(s)
données	Message de réponse	Alpha

Objectif : Envoyer un message de réponse à la machine initiatrice d'un HTTP POST.

Description

Lorsque le serveur 4D Business Kit reçoit une requête de type HTTP POST à l'aide de 4DBKHttpPostCallBack, celui-ci doit renvoyer une réponse à la machine qui a initié cette requête pour fermer la session TCP/IP.

La commande 4DBKHttpPostResponse permet l'envoi de cette réponse.

Exemples

(1) Reportez-vous, dans le site TEST, à la page *payment_yellowpay_cb.htm*.

(2) Exemple de réponse simple :

<!--#4DBKHttpPostResponse/OK-->

Référence : 4DBKHttpPostCallBack.

4DBKHttpGet

<!--#4DBKHttpGet/serveur,port,requête[,début,fin]-->

Paramètre	Description	Valeur(s)
serveur	Adresse du serveur Web à contacter	Adresse HTTP
port	Port du serveur Web à contacter	Numérique
requête	Requête http de type get	Chaîne de caractères
début	Chaîne de départ du résultat	Chaîne de caractères
fin	Chaîne de fin du résultat	Chaîne de caractères

Objectif : Utiliser des informations contenues dans des pages Web externes ou effectuer des requêtes vers des sites externes.

Description

La commande 4DBKHttpGet permet de se connecter à d'autres serveurs Web via le protocole HTTP GET pour récupérer des résultats statiques (informations stockées dans une page Web) ou dynamiques (informations stockées dans une page Web après avoir effectué une requête paramétrée).

Cette commande offre des multiples possibilités d'extensions et permet de considérer tout site Web comme un fournisseur de service potentiel.

Passez dans le paramètre *requête* l'URL de la requête HTTP complète. Cet URL est visible dans la zone de lien du navigateur après que la requête ait été effectuée (par exemple *http://www.time.gov:8085/timezone.cgi?America/n/+3*)

A partir de cet URL, il est possible de déduire le paramètre *serveur* qui se situe entre http:// et le caractère *:* ou / suivant. Dans l'exemple précédent : *www.time.gov*

Le *port* est le paramètre suivant éventuellement le caractère *:*
Si aucun caractère *:* n'est affiché, le port est 80 par défaut. Dans l'exemple précédent : *8085*

Les paramètres *début* et *fin* décrivent les chaînes de caractères qui encadrent le résultat à récupérer dans la page HTML interrogée. Il vous suffit donc de repérer auparavant dans le code source de la page HTML l'emplacement du résultat que vous souhaitez récupérer.

Imaginons que, dans le cadre de l'exemple précédent, le code source de la page soit :

```
align="center"><font size="7" color="white"><b>23:34:45<br>
</b></font><font size="5" color="white">Wednesday, April 23, 2003<br>
</font>
```

Le résultat est donc compris entre align="center"><font size="7" color="white"><b> et
.
Le paramètre *début* devra donc contenir align="center"><font size="7"@<b> et le paramètre *fin*
.

Il est possible d'utiliser le caractère @ (joker) dans *début* et *fin*. Ce caractère signifie "quelque chose". Par exemple, align@center@size@7@b> désigne une chaîne contenant *align...center...size...7...b>*. Il est possible de cumuler plusieurs de ces caractères dans *début* ou *fin*.

Dans certains cas, il peut être nécessaire de préciser les chaînes <!-- et --> dans les paramètres *début* et *fin*. Toutefois, ces chaînes empêcheront le fonctionnement de la commande car elle est encadrée par ces mêmes chaînes. Il faut alors utiliser les mots-clés COMMENT_START et COMMENT_END.
De même, les caractères , et ; réservés par 4D Business Kit doivent être remplacés par les mots-clés COMMA et SEMICOLON.

Exemples

(1) Reportez-vous, dans le site TEST, aux pages *httpgets.htm* et *network.htm*.

(2) La requête suivante permet de récupérer l'heure atomique en GMT+1. Elle est exécutée sur le serveur www.time.gov (port 80). Le résultat est encadré par les caractères <font size="7" __quelquechose__ <b> et
.

```
<!--#4DBKHTTPGet/www.time.gov,80,http://www.time.gov/timezone.cgi
?Europe/s/+1,<font size="7"@<b>,<br>-->
```

(3) La requête suivante permet de récupérer l'adresse IP du serveur. Elle est exécutée sur le serveur www.timestocome.com (port 80). Le résultat est le seul élément de la page Web, on ne précise donc pas de paramètres *début* et *fin*.

```
<!--#4DBKHTTPGet/www.timestocome.com,80,http://www.timesto-
        come.com/webtools/getip.shtml-->
```

(4) Cette requête se connecte à la base de données du RIPE pour obtenir le nom de domaine correspondant à l'adresse IP 195.22.12.45. Le résultat que l'on souhaite obtenir se trouve entre netname: et la ligne suivante qui commence par descr.

```
<!--#4DBKVarSet/VT10=http://www.ripe.net/perl/whois?form_type=simpl
        e&full_query_string=&searchtext=-->
<!--#4DBKVarSet/VT10+=195.22.12.45-->
<!--#4DBKVarSet/VT10+=&do_search=Search-->
<!--#4DBKHTTPGet/www.ripe.net,80,4DBKVar/VT10,netname:,descr-->
```

4DBKCookieSet

<!--#4DBKCookieSet/nom=valeur-->

Paramètre	Description	Valeur(s)
nom	Nom du cookie	Texte
valeur	Texte à stocker dans le cookie	Texte

Objectif : Ecrire une valeur dans un cookie.

Description

Il est possible de stocker des valeurs sur la machine de l'utilisateur qui navigue sur le site de la boutique. Ces valeurs sont stockées dans des cookies. Chaque cookie est identifié par un *nom*.

Le développeur peut stocker tout type d'information dans les cookies. Généralement, les cookies servent à conserver des préférences utilisateur permettant, par exemple, de présenter des sélections d'articles ciblées aux utilisateurs.

Les cookies sont écrits par la commande 4DBKCookieSet et sont relus par la commande 4DBKCookie.

Passez dans le paramètre *nom* le nom du cookie à écrire et dans *valeur* la valeur à stocker dans le fichier.

Note : Pour des raisons de sécurité, certains internautes interdisent la prise en charge des cookies au niveau de leur navigateur.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemples

(1) Reportez-vous, dans le site TEST, à la page *cookies.htm*.

(2) Cet exemple stocke la valeur "bleu,vert,jaune" dans le cookie couleurs_preferees :

```
<!--#4DBKCookieSet/couleurs_preferees=bleu,vert,jaune-->
```

Référence : 4DBKCookie.

4DBKCookie

<!--#4DBKCookie/nom[,format]-->

Paramètre	Description	Valeur(s)
nom	Nom du cookie	Texte
format	Format d'affichage valide	Voir chapitre Formats d'affichage et fonctions

Objectif : Afficher le contenu d'un cookie.

Description

Il est possible de stocker des valeurs sur la machine de l'utilisateur qui navigue sur le site de la boutique. Ces valeurs sont stockées dans des cookies. Chaque cookie est identifié par un *nom*.

Les cookies sont écrits par la commande 4DBKCookieSet et sont relus par la commande 4DBKCookie.

Note : Pour des raisons de sécurité, certains internautes interdisent la prise en charge des cookies au niveau de leur navigateur.

Conditions

Il faut avoir préalablement fixé la boutique courante à l'aide de la commande 4DBKStoreSet.

Exemples

(1) Reportez-vous, dans le site TEST, à la page *cookies.htm*.

(2) Cet exemple retourne la valeur du cookie couleurs_preferees :

```
<!--#4DBKCookie/couleurs_preferees--> retourne par exemple bleu,vert,
        jaune
```

Référence : 4DBKCookieSet.

4DBKStringRandom

<!--#4DBKStringRandom/longueur-->

Paramètre	Description	Valeur(s)
longueur	Longueur de la chaîne à générer	Numérique

Objectif : Retourner une chaîne composée de caractères choisis au hasard.

Description

4DBKStringRandom retourne une chaîne de caractères de *longueur* souhaitée, construite de manière aléatoire. Cette fonction est utile notamment pour la génération de mots de passe ou de clés.

Exemples

(1) Exemple simple d'utilisation :

```
<!--#4DBKStringRandom/8--> // retourne par exemple wDMDZgFO
```

(2) Cet exemple permet la création d'un compte automatique et l'identification de l'utilisateur :

```
<a href="4DBKExecute:4DBKStoreSet/TEST;4DBKCustomerLogout;
        4DBKFieldSet/CusCode=4DBKCustomerCodeNew;4DBKFieldSet/
        CusPassword=4DBKStringRandom/4;4DBKGo/info.htm">Cliquez
        ici pour créer un nouveau compte</a>
```

Dans la page *info.htm*, on pourra afficher:

```
<!--#4DBKField/CusCode--> // le code
<!--#4DBKField/CusPassword--> // le mot de passe
```

4DBKExtCallParm1Set, 4DBKExtCallParm2Set, 4DBKExtCallParm3Set

<!--#4DBKExtCallParm1Set/données-->

Paramètre	Description	Valeur(s)
données	Données à communiquer aux méthodes 4DBKPrice et 4DBKShipping d'une autre application 4D	Texte

Objectif : Passer des paramètres supplémentaires lors de la délégation de calcul de prix ou de modes d'expédition.

Description

Lorsque le serveur 4D Business Kit a besoin de calculer le prix d'un article ou la liste des modes d'expédition, et si les options correspondantes sont cochées et paramétrées dans les pages **Appels ext.** et **Appels ext. (2)** de la fenêtre de définition de boutique, les méthodes projet 4DBKPrice ou 4DBKShipping du composant 4DBK installé dans votre application 4D sont appelées.

Dans certains cas, il peut être nécessaire de passer des paramètres personnalisés à ces méthodes. Par exemple le numéro d'un bon de réduction si le client connecté est un particulier, ou toute autre caractéristique requise pour fournir des résultats adéquats.

Les paramètres personnalisés seront reçus par 4DBKPrice et 4DBKShipping dans les variables *ExtCallArg1*, *ExtCallArg2* et *ExtCallArg3* (la documentation de ces commandes est fournie dans le *Manuel technique avancé*).

Les commandes 4DBKExtCallParm*N*Set doivent être définies dans les pages HTML avant les commandes d'affichage des prix — ou, plus précisément, dès qu'une fiche article devient courante (via, par exemple, les commandes 4DBKRecordSet, 4DBKRecordNext, 4DBKQuerySet, etc.).

Exemple

L'exemple suivant passe un numéro de bon de réduction :

```
<!--#4DBKExtCallParm1Set/NOEL4380X1389-->
```

4DBKMD5

<!--#4DBKMD5/chaîne-->

Paramètre	Description	Valeur(s)
chaîne	Chaîne dont on veut produire un *digest* de type MD5	Texte

Objectif : Produire un *Message digest 5* (MD5) à partir d'une chaîne de caractères.

Description

Cette commande retourne un *digest* de type MD5 sur la base de la *chaîne* passée en paramètre. MD5 est un algorithme de cryptage basé sur le hachage unidirectionnel d'un message.

Cette commande est utile pour passer des paramètres à certaines passerelles de paiement.

4DBKMD5_HMAC

<!--#4DBKMD5_HMAC/clé,chaîne-->

Paramètre	Description	Valeur(s)
clé	Clé de hachage	Texte
chaîne	Chaîne dont on veut produire un *digest* de type HMAC MD5	Texte

Objectif : Produire un *Message digest 5* (MD5) de type *HMAC* à partir d'une chaîne de caractères.

Description

Cette commande retourne un *digest* de type MD5 HMAC sur la base de la *chaîne* et de la *clé* passées en paramètres. MD5 HMAC est un mécanisme d'authentification des messages combinant le hachage MD5 et une clé secrète.

Cette commande est utile pour passer des paramètres à certaines passerelles de paiement.

19 Champs

Ce chapitre liste les champs utilisables dans les pages Web ainsi que leurs propriétés d'affichage, de lecture, d'écriture, de recherches, de tris, de sélection, etc.

Voici la signification de chaque propriété :

- **Field** : le champ peut être lu (en général par 4DBKField, par exemple 4DBKField/T01).
- **Query** : le champ peut être utilisé pour une recherche dans le cadre d'un 4DBKQuerySet (par exemple 4DBKQuerySet/T01=mobile@) depuis une page Web ou depuis une autre application 4D via le composant 4DBK.
- **Distinct** : le champ peut être utilisé pour un filtrage dans le cadre d'un 4DBKSelectionDistinct (par exemple 4DBKSelectionDistinct/L01).
- **Sort** : le champ peut être trié dans le cadre d'un 4DBKSortSet (par exemple 4DBKSortSet/C01).
- **Basket** : le champ peut être utilisé dans le cadre d'un 4DBKField ou 4DBKSortSet pour une sélection de panier (sélection B).
- **Set** : le champ peut être écrit (en général par un 4DBKFieldSet).
- **Import** : le champ peut être spécifié dans un fichier d'import sous la forme [champ] (par exemple [T01]).
- **Export** : le champ existe dans les fichiers d'export sous la forme [champ] (par exemple [T03]).

Articles

Les champs d'articles peuvent être lus à l'aide de la commande 4DBKField depuis une page Web, ou en utilisant le module d'exportation, ou encore depuis une autre application 4D via le composant 4DBK.

Ils peuvent être écrits à l'aide du module d'importation ou du composant 4DBK.

La colonne *Query* s'applique aux champs depuis une page Web ou depuis le composant 4DBK.

Id

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X	X		X				[Id]	Alpha 14

Identifiant interne de la fiche (par exemple *20412382737486*).

L'utilisation de cet identifiant n'est pas nécessaire dans la construction d'un site.

Note : Cet identifiant interne peut éventuellement être utilisé pour la génération de requêtes de type GET vers des sites extérieurs (partenariats) ou pour recomposer le nom d'une vue (ex: T01_20412382737486.jpg désigne l'image réduite de la première vue en utilisant l'identifiant 20412382737486).

Ts

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X	X		X	X		[Ts]	[Ts]	Alpha 14

Indicateur de date et d'heure (*timestamp*) de la dernière modification de la fiche.

Cet indicateur, au format AAAAMMJJHHMMSS GMT (par exemple 20030418173012), est géré automatiquement, il n'est pas nécessaire de le mettre à jour.

Note : Le champ Ts peut être utilisé dans le cadre d'un 4DBKSortSet/Ts< par exemple pour présenter les articles du panier de commande par ordre d'ajout dans le panier.

Layout

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X	X	X	X	X				Alpha 80

Nom du masque lié à la fiche article (par exemple *Standard*).

StActive

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X	X		X	X		[StActive]	[StActive]	Booléen

Indique si l'article est au catalogue ou non.

Cet état peut être soumis à un automatisme en fonction d'une période. Il est possible de rendre l'article actif ou inactif après ou avant une certaine date ou pendant une période donnée (après quoi l'article reprend son état initial).

Notes :
• Lors d'un 4DBKQuerySet sur un champ, les articles qui ne sont pas actifs sont automatiquement retirés de la sélection courante. Ce comportement par défaut peut être modifié dans la fenêtre de définition de la boutique, page **Données (2)**.
• Il est possible d'importer et d'exporter les données de programmation de l'état futur de l'article à l'aide du champ DaActive.
• Dans les fichiers d'import/export, le champ prend la valeur 0 ou 1.

C01

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X	X	X	X	X		nom [C01,masque]	nom [C01,masque]	Alpha 30

Code principal de l'article / clé primaire.

Ce code unique qui identifie l'article (par exemple *84203*) est utilisé dans de nombreuses commandes pour le désigner.

Notes :
• Dans le cadre de l'utilisation de fichiers d'import/export, l'en-tête correspondant à C01 peut apparaître sous la forme : nom_du_champ_dans_le_masque [C01,nom_masque] pour indiquer le masque de données auquel été associée la fiche article ainsi que le nom du champ dans ce masque. Par exemple, *code_produit [C01,Standard]* indique que le champ C01 s'appelle aussi code_produit et qu'il appartient au masque Standard. Il est donc possible d'utiliser la commande 4DBKField sous la forme 4DBKField/code_produit au lieu de 4DBKField/C01 pour afficher le code C01 de l'article courant.
• Pour supprimer un article existant dans la base de données depuis un fichier d'import, il faut suffixer le code C01 de la chaîne *DEL*.

Par exemple, 84203 ajoutera ou modifiera la fiche article identifiée par ce code, 84203*DEL* supprimera cette même fiche.

C02, C03, C04

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X	X	X	X	X		nom [Cnn]	nom [Cnn]	Alpha 30

Codes secondaires de l'article.

Ces codes secondaires peuvent être utilisés comme codes de référence additionnels, par exemple le code de référence dans la nomenclature du fabricant de l'article (*AP22696*) ou bien un code EAN/UPC (*EAN8590339342039*).
Ils peuvent également être utilisés comme zones de stockage pour une caractéristique discriminante de l'article comme la couleur (BLEU) ou la taille (XL).

Note : Dans le cadre de d'utilisation de fichiers d'import/export, l'entête correspondant à C02-C04 peut apparaître sous la forme nom_du_champ [C02-C04] pour indiquer le nom du champ. Par exemple, *taille [C02]* indique que le champ C02 s'appelle aussi taille. Il est donc possible d'utiliser la commande 4DBKField sous la forme 4DBKField/taille au lieu de 4DBKField/C02 pour afficher le code C02 de l'article courant.

T01

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X	X	X	X	X		nom [T01]	nom [T01]	Alpha 80

Désignation de l'article.

Ce champ désigne généralement le nom principal de l'article (par exemple *Savon de Marseille lavande*).

Note : Voir note concernant C02, C03, C04.

T02 à T16

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X	X	X	X	X		nom [Tnn]	nom [Tnn]	Alpha 80

Caractéristiques de l'article (par exemple: *12x15 cm*).

Ces textes peuvent être utilisés pour stocker tout type d'information ou de caractéristique sur l'article.

Par exemple, ces champs peuvent stocker le nombre de pièces contenues dans l'article s'il s'agit d'un lot (*x4*), la dimension d'un ouvrage (*12x15 cm*) ou la période de floraison d'une plante sous la forme d'un calendrier (*JANFEVMAR---------JULAUG------NOV---*) qu'il sera possible de décomposer et d'afficher dans les pages Web au moyen des formats d'affichage.

Note : Voir note concernant C02, C03, C04.

L01 à L05

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X	X	X	X	X		nom [Lnn]	nom [Lnn]	Alpha 80

Libellé de l'élément de la famille 01 à 05 lié à la fiche article.

Les familles sont utilisées pour classer les articles dans des catégories énumérées (par exemple : *Téléphones mobiles*).

Notes :
• Voir note concernant C02, C03, C04.
• Pour importer un libellé dans plusieurs langues, il faut utiliser la syntaxe suivante :
libellé_langue_1$$libellé_langue_2$$libellé_langue_3$$libellé_langue_4
Par exemple, avec *Téléphones mobiles$$Mobile phones*, *Téléphone mobile* deviendra le libellé pour la 1re langue et *Mobile phones* le libellé pour la 2e langue. Pour rappel, le libellé est affiché en fonction de la langue courante fixée par 4DBKPrefsSet/DataLanguage.

L01_ à L05_

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X	X	X	X	X		nom [Lnn]	nom [Lnn]	Alpha 30

Code de l'élement de la famille 01 à 05 lié à la fiche article.

Ce code complémentaire du libellé L01 à L05 permet de classifier plus finement les articles pour établir, par exemple, une hiérarchie entre plusieurs familles (en reprenant, dans le code, le code de la famille de niveau supérieur).

Si l'élément L01 associé à l'article est défini par SUR/Surgelés (code *SUR*, libellé *Surgelés*), l'élément L02 associé au même article pourrait être SUR*POI/Poisson (code *SUR*POI*, libellé *Poisson*) pour indiquer que l'article est un poisson surgelé.

En balayant tous les éléments de L02 commençant par *SUR*, il est ainsi possible d'obtenir facilement la liste des catégories d'articles surgelés.

Notes :
• Voir note concernant C02, C03, C04.
• Pour importer un code et un libellé, il faut utiliser la syntaxe code$libellé. Par exemple *MOB$Téléphones mobiles*.
• Pour importer un code et un libellé dans plusieurs langues, il faut utiliser la syntaxe suivante :
code$libellé_langue_1$$libellé_langue_2$$libellé_langue_3$$libellé_langue_4
Par exemple, avec *MOB$Téléphones mobiles$$Mobile phones*, *MOB* devriendra le code famille (L01_), *Téléphone mobile* le libellé pour la 1re langue (L01) et *Mobile phones* le libellé pour la 2e langue (L01). Pour rappel, le libellé est affiché en fonction de langue courante fixée par 4DBKPrefsSet/DataLanguage.

R01 à R05

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X	X		X	X		nom [Rnn]	nom [Rnn]	Réel

Nombres réels associés à l'article.

Ces types de champs sont utilisés pour stocker des nombres ou des montants (par exemple : *3 000 000*).

Note : Voir note concernant C02, C03, C04.

D01 à D05

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X	X		X	X		nom [Dnn]	nom [Dnn]	Date

Dates associées à l'article.

Ces champs sont utilisés pour stocker des dates (par exemple : *18/04/1966*).

Note : Voir note concernant C02, C03, C04.

B01 à B05

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X	X		X	X		nom [Bnn]	nom [Bnn]	Booléen

Valeurs booléennes ou binaires (valeurs ayant le statut vrai ou faux) associées à l'article.

Ces champs sont utilisés pour déterminer un choix oui/non qui peut être représenté, par exemple, sous forme d'une case à cocher.

Notes :
• Voir note concernant C02, C03, C04.
• Dans les fichiers d'import/export, le champ prend la valeur 0 (faux) ou 1 (vrai).

M01 à M05

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X	X			X		nom [Mnn]	nom [Mnn]	Liste alpha

Listes d'éléments liés à la fiche article.

Ce type de champ est utilisé pour associer une liste d'éléments énumérés à un article. Cette liste peut énumérer par exemple une particularité technique (*USB, USB2, Firewire*), une liste d'auteurs ou d'acteurs (*John Wayne,Burt Lancaster,* etc.).

Dans les pages Web ainsi que dans les fichiers d'import/export, les éléments de la liste doivent être séparés par une virgule. Par exemple : *Pomme, Poire, Banane.*

Le nombre d'éléments différents d'un champ Mnn pour tous les articles (par exemple, le nombre d'acteurs différents) peut être obtenu à l'aide de la commande 4DBKStoreMultipleSize et chaque élément (chaque acteur) à l'aide de 4DBKStoreMultiple.
Le nombre d'éléments différents d'un champ Mnn pour un seul article (par exemple, le nombre d'acteurs d'un film donné) peut être obtenu à l'aide de la commande 4DBKFieldMultipleSize et chaque élément de la liste (les acteurs du film) à l'aide de 4DBKFieldMultiple.

Notes :
• Voir note concernant C02, C03, C04.

• Pour importer un code et un libellé dans plusieurs langues, il faut utiliser la syntaxe suivante :
libellé_langue_1$$libellé_langue_2$$libellé_langue_3$$libellé_langue_4
Par exemple avec *Pomme$$Apple, Poire$$Pear, Banane$$Banana, Pomme, Poire* et *Banane* deviendront les libellés pour la 1re langue tandis que *Apple, Pear* et *Banana* deviendront les libellés pour la 2e langue. Pour rappel, le libellé est affiché en fonction de langue courante, fixée par 4DBKPrefsSet/DataLanguage.

Text ou Text1, Text2, Text3

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X	X			X		[Text] ou [Text1,2,3]	[Text] ou [Text1,2,3]	Alpha 30 Ko

Descriptifs de l'article.

Ces champs sont utilisés pour stocker les descriptions textuelles des articles. Ils peuvent contenir du code html.

A propos des recherches : il est possible d'effectuer des recherches dans ce type de champ (non indexé) à l'aide de 2 algorithmes différents.

- Le 1er algorithme utilise le nom standard du champ, par exemple : 4DBKQuerySet/Text1=@avion@.
- Pour le 2e algorithme, le nom du champ doit être suffixé avec _alt. Par exemple, 4DBKQuerySet/Text1_alt=@avion@. Il est possible avec cet algorithme d'utiliser plusieurs caractères joker @ au sein d'une chaîne. Par exemple, 4DBKQuerySet/Text2_alt=@télé@haute@tion@ trouvera les articles dont les descriptifs contiennent "télévision haute définition" ou "télévision haute résolution".

  Note : Dans les fichiers d'import/export, les caractères CR (retour chariot, code ASCII 13) sont codés **^r** et les caractères TAB (tabulation, code ASCII 09) **^t**.

Linked

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X						[Linked]	[Linked]	Alpha 400

Liste d'articles liés à l'article courant.

Il est possible de lier une liste d'articles à l'article courant en énumérant leurs codes principaux (C01). Par exemple, 0012, 0031, 0033 peuvent être liés à l'article courant.

Les articles liés peuvent permettre de prendre connaissance, lors de l'affichage de la fiche détail d'un article, des options et équipements associés.

Dans les pages Web ainsi que dans les fichiers d'import/export, les éléments de la liste doivent être séparés par une virgule.

Le nombre d'éléments de la liste (le nombre d'articles liés) peut être obtenu à l'aide de la commande 4DBKFieldLinkedSize et chaque élément (le code C01 de chaque article lié) à l'aide de 4DBKFieldLinked.

StNew

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X	X		X	X		[StNew]	[StNew]	Booléen

Indique si l'article est nouveau.

Note : Dans les fichiers d'import/export, le champ prend la valeur 0 (faux) ou 1 (vrai).

WgItem

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X	X		X	X		[WgItem]	[WgItem]	Réel

Indique le poids de l'article.

L'unité de saisie est libre (kg ou lb).

Note : Pour afficher ou importer/exporter le poids de l'article dans une unité autre que l'unité de saisie, il faut utiliser la syntaxe WgItem.coefficient. Par exemple, si le poids a été saisi en livres et que l'on souhaite l'afficher en kilogrammes, on utilisera *WgItem.2.22* (1 kg = 2,22 lb).

VlItem

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X	X		X	X		[VlItem]	[VlItem]	Réel

Indique le volume de l'article.

L'unité de saisie est libre (m^3 ou autre).

Note : Pour afficher ou importer/exporter le volume de l'article dans une unité autre que celle de saisie, il faut utiliser la syntaxe VlItem.coefficient. Par exemple, si le volume a été saisi en m3 et que l'on souhaite le diviser par 3 on utilisera *VlItem.0.33*.

StStatus1, StStatus2

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X	X		X	X		[StStatus1/2]	[StStatus1/2]	Booléen

Indique la valeur courante du statut 1 ou du statut 2 de l'article.

Ces statuts diffèrent des booléens B01 à B05 car ils peuvent être soumis à des automatismes en fonction de périodes. Il est ainsi possible de changer l'état du statut après ou avant une certaine date ou pendant une période donnée (après quoi le statut reprend son état initial).

Note : Dans les fichiers d'import/export, le champ prend la valeur 0 (faux) ou 1 (vrai).

VaCounter1, VaCounter2, VaCounter3

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X					X	[VaCountern]	[VaCountern]	Réel

Valeur des compteurs 1 à 3 de l'article.

Les compteurs des articles peuvent être utilisés pour comptabiliser, par exemple, le nombre de consultations de l'article ou le chiffre d'affaires lorsque cet article est commandé.

Pour lire un compteur, il faut utiliser la commande 4DBKCounter.

Pour incrémenter, décrémenter ou fixer la valeur d'un compteur, il faut utiliser la commande 4DBKCounterSet.

QtStock

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X	X		X	X		[QtStock]	[QtStock]	Entier long

Indique la quantité en stock de l'article.

PrItemEXV

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X	X		X	X		[PrItemEXV]	[PrItemEXV]	Réel

Indique le prix hors taxes de l'article.

Note : EXV signifie "EXcluding Vat".

PrSpecialEXV

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X	X		X	X		[PrSpecialEXV]	[PrSpecialEXV]	Réel

Indique le prix hors taxes de l'article lorsqu'il est en promotion.

Le champ StSpecialPrice détermine si l'article est en promotion ou non.

StSpecialPrice

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X	X		X	X		[StSpecialPrice]	[StSpecialPrice]	Booléen

Indique si l'article est ou non en promotion.

Note : Dans les fichiers d'import/export, le champ prend la valeur 0 (faux) ou 1 (vrai).

TyTax

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X	X		X	X		[TyTax]	[TyTax]	Entier/Alpha

Indique la catégorie de taxe de l'article.

Chaque catégorie (A,B,C,D ou E) correspond à des taux différents. Ces taux sont définis dans le module des taxes au niveau de chaque type de taxe.

Il est possible de définir plusieurs types de taxes pour une même zone géographique pour gérer, par exemple, une taxe provinciale et une taxe locale.

Note : Dans le fichier d'import, il est possible de préciser une valeur de A à E, ou bien un champ vide si l'article ne doit pas être taxé.

PrSaleEXV / VAT / INV
PcSaleVATRate

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X				X				Réel

Ces champs indiquent les montants associés à l'article, hors promotions et réductions, calculés en fonction du client.

Ces valeurs représentent le prix public de l'article. Elles tiennent compte des règles de taxe mais pas des remises. Elles sont utilisées pour calculer par différence, avec les champs PrCustomer, le montant de remise accordé.

- PrSaleEXV représente le montant hors taxes (EXcluding).
- PrSaleVAT représente le montant des taxes.
- PcSaleVATRate représente le taux de taxes.
- PrSaleINV représente le montant avec taxes (INcluding).

PrCustomerEXV / VAT / INV
PcCustomerVATRate

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X				X				Réel

Ces champs indiquent les montants courants associés à l'article, incluant les promotions et réductions, et calculé en fonction du client.

Ces valeurs représentent le prix normal de l'article à afficher. Elles tiennent compte des remises clients et des règles de taxe.

Il est aussi à noter que ces valeurs peuvent indiquer le prix du prochain article à commander. Par exemple, si le client a déjà mis 2 articles identiques dans son panier, et qu'il existe une remise par quantité à partir de 3 unités pour cet article, alors le prix rendu par PrCustomer tiendra compte de la remise par quantité.

- PrCustomerEXV représente le montant hors taxes (EXcluding).

- PrCustomerVAT représente le montant des taxes.
- PcCustomerVATRate représente le taux de taxes.
- PrCustomerINV représente le montant avec taxes (INcluding).

PrDeltaEXV / VAT / INV
PcDelta

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X				X				Réel

Ces champs indiquent l'économie réalisée.

Ces valeurs représentent la différence entre les champs de type PrCustomer et PrSale.

- PrDeltaEXV représente le montant hors taxes économisé (EXcluding).
- PrDeltaVAT représente le montant de taxes économisé.
- PrDeltaINV représente le montant avec taxes économisé (INcluding).
- PcDelta représente le taux d'économie.

DaDiscount1 à DaDiscount4

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
						[DaDiscount1-5]	[DaDiscount1-5]	Alpha

Ces champs indiquent les remises par quantité.

4D Business Kit permet de fixer jusqu'à quatre paliers de quantité pour lesquels le client bénéficiera d'un taux de remise différent. Par exemple, plus la quantité commandée augmente plus la remise est forte (de 10 à 50 : -10% ; de 51 à 100 : -15%, au-delà : -20%).

Dans les fichiers d'import/export, les remises doivent être spécifiées sous la forme Quantité basse/Quantité haute/Pourcentage de remise. Par exemple, *3/5/4* signifie que pour une quantité de 3 à 5 articles une remise de 4% sera appliquée.

TySaleMode

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
						[TySaleMode]	[TySaleMode]	Alpha

Indique la remise par client.

Chaque client peut se voir attribuer des remises personnalisées par article, plus exactement par *catégorie d'articles*. Par défaut, ces catégories sont libellées de A à E — ces libellés peuvent être personnalisés dans les paramètres de la boutique. Il est ensuite possible d'associer un article à une de ces catégories puis de préciser, pour chaque client, le montant de remise par catégorie.

Note : Dans le fichier d'import, il est possible de préciser une valeur de 1 à 5 ou de A à E, ou bien un champ vide si l'article ne doit pas être lié à une catégorie.

StNoShipping

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X	X		X	X		[StNoShipping]	[StNoShipping]	Booléen

Indique si l'article doit être ignoré lors du calcul des frais de port.

Si ce champ est à vrai, alors la quantité, le prix et le poids de l'article ne seront pas pris en compte pour la détermination des modes d'expédition possibles pour le panier et le client, et donc dans le calcul des frais de port.

Note : Dans les fichiers d'import/export, le champ prend la valeur 0 (faux) ou 1 (vrai).

DaActive

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
						[DaActive]	[DaActive]	Alpha

Fournit les données de programmation pour l'état futur de StActive.

Ce champ permet de fixer le moment où le champ StActive (qui indique si l'article est au catalogue ou non) passe dans un autre état.

Les données contenues dans le champ doivent être spécifiées sous la forme 0 ou 1/période début au format AAAAMMJJHHMMSS/période fin au format AAAAMMJJHHMMSS.

Exemples :

- 0/20031009083000/00000000000000 : signifie que StActive passe à 1 immédiatement (article disponible) puis sera mis à 0 automatiquement (article indisponible) à partir du 9 octobre 2003 à 08:30:00 et restera dans cet état.
- 1/20031009083000/00000000000000 : signifie que StActive passe à 0 immédiatement (article indisponible) puis sera mis à 1 automatiquement (article disponible) à partir du 9 octobre 2003 à 08:30:00 et restera dans cet état.
- 0/00000000000000/20031014233000 : signifie que StActive passe à 0 immédiatement (article indisponible) et restera dans cet état jusqu'au 14 octobre 2003 à 23:30:00 après quoi il passera à 1 (article disponible).
- 1/00000000000000/20031014233000 : signifie que StActive passe à 1 immédiatement (article disponible) et restera dans cet état jusqu'au 14 octobre 2003 à 23:30:00, après quoi il passera à 0 (article indisponible).
- 0/20031009083000/20031014233000 : signifie que StActive passe à 1 immédiatement (article disponible) et restera dans cet état jusqu'au 9 octobre 2003 à 08:30:00, puis il passera à 0 (article indisponible) jusqu'au 14 octobre 2003 à 23:30:00 après quoi il repassera à 1 (article disponible).
- 1/20031009083000/20031014233000 : signifie que StActive passe à 0 immédiatement (article indisponible) et restera dans cet état jusqu'au 9 octobre 2003 à 08:30:00, puis il passera à 1 (article disponible) jusqu'au 14 octobre 2003 à 23:30:00 après quoi il repassera à 0 (article indisponible).

DaSpecial

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
						[DaSpecial]	[DaSpecial]	Alpha

Fournit les données de programmation pour l'état futur de StSpecialPrice.

Ce champ permet de fixer le moment où le champ StSpecialPrice (qui indique si l'article est en promotion ou non) passe dans un autre état.

Les données contenues dans le champ doivent être spécifiées sous la forme 0 ou 1/période début au format AAAAMMJJHHMMSS/période fin au format AAAAMMJJHHMMSS.

- Exemples : voir les exemples du champ DaActive.

DaStatus1, DaStatus2

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
						[DaStatus1 - 2]	[DaStatus1 - 2]	Alpha

Fournit les données de programmation pour l'état futur de StStatus1 ou StStatus2.

Ces champs permettent de fixer le moment où respectivement les champ StStatus1 et StStatus2 (qui indiquent si le statut1 et le statut2 de l'article sont à vrai ou non) passent dans un autre état.

Les données contenues dans le champ doivent être spécifiées sous la forme 0 ou 1/période début au format AAAAMMJJHHMMSS/période fin au format AAAAMMJJHHMMSS.

- Exemples : voir les exemples du champ DaActive.

Vues

Chaque article possède au moins une représentation graphique appelée vue.

Il est possible d'associer jusqu'à 20 vues par article et chaque vue est constituée d'une image, d'une imagette et d'une icône (32x32) au format jpeg. Un texte peut venir compléter ces images pour les décrire (par exemple, chaque texte peut décrire les différentes vues d'un appartement à vendre).

Les images et textes doivent être placés dans un dossier nommé Pictures, situé au même niveau que le fichier d'import.

Lorsqu'une image est importée, elle est éventuellement redimensionnée et l'imagette ainsi que l'icône sont générées automatiquement selon les paramètres de redimensionnement et de compression définis pour la boutique.

Picture01 à Picture20

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
						[PictureNN]	[PictureNN]	Alpha

Indique le nom du fichier de l'image principale d'une vue de l'article (par exemple *renault12.jpg*).

Une fois importé, ce fichier génèrera une image, une imagette et une icône qui seront copiées dans le dossier Views de la boutique, sous la forme suivante :

- Pvv_ID.jpg pour les images (par exemple *P01_20030418173012.jpg*)
- Tvv_ID.jpg pour les imagettes (par exemple *T01_20030418173012.jpg*)
- Ivv_ID.jpg pour les icônes (par exemple *I01_20030418173012.jpg*)

... où vv est le numéro de la vue et ID est l'identifiant de l'article.

TextFile01 à TextFile20

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
						[TextFileNN]	[TextFileNN]	Alpha 30 Ko

Indique le nom du fichier texte associé à une vue de l'article (par exemple *renault12.txt*).

Une fois importé, ce fichier génèrera un texte qui sera copié dans le dossier Views de la boutique sous la forme suivante :

- Avv_ID.jpg (par exemple *A01_20030418173012.txt*)

... où vv est le numéro de la vue et ID est l'identifiant de l'article.

Note : Les caractères CR (retour chariot, code ASCII 13) sont codés **^r** et les caractères TAB (tabulation, code ASCII 09) **^t**.

Panier : lignes de commandes

Lorsqu'un article est placé dans le panier, les informations le concernant soumises à certaines conditions (remises, promotions) sont conservées pour être représentées plus tard — par exemple, sur un bordereau de commande récapitulatif.

Une commande est une suite de lignes de commandes, chacune étant représentée par un élément de la sélection B (contenu du panier). Chaque ligne possède ses caractéristiques propres : quantité, prix unitaire, poids unitaire, prix total, etc.

Lors de l'affichage d'une ligne de commande, il est possible d'utiliser tous les champs qui commençent par **Bsk** et ceux qui sont indiqués comme valides pour la sélection B (voir la colonne **Basket** dans le tableau).

Texte discriminant

Chaque article du panier est unique (il est désigné par son code principal C01) et apparaît sous la forme d'une ligne de commande.

Si un article identique est commandé à deux instants différents au sein de la même commande, les deux articles seront regroupés sur la même ligne.
Cependant, il peut être utile de considérer que 2 exemplaires du même article ne sont pas uniques au sein d'une même commande et qu'ils doivent apparaître sous 2 lignes de commandes différentes. Par exemple, si l'article est un billet d'avion à destination de Marseille et qu'au sein d'une même commande le même billet est ajouté au panier sous 2 noms différents, il faut alors qu'il apparaisse dans deux lignes différentes.
Autre exemple : l'article est un menu de restaurant et il est lui-même la combinaison d'autres articles. Lors de la même commande, un client A et un client B commandent le même menu à 10 euros mais prennent une entrée, un plat et un dessert différents. Le même article (le menu) doit donc apparaître sous deux lignes différentes avec pour chaque menu, le détail entrée-plat-dessert.
Pour prendre en charge ce type de situation, vous pouvez désigner un champ texte discriminant à l'aide de la commande 4DBKInfoSet avant d'ajouter l'article au panier par 4DBKBasketSet. Une fois ce texte discriminant positionné, deux articles identiques apparaîtront sur deux lignes de commande différentes si ce texte est différent pour chacun des articles.

Pour plus d'informations, consultez l'exemple d'affichage d'un panier, fourni dans le chapitre "Panier", page 83.

BskId

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
				X				Entier

Indique le numéro d'ordre de la ligne de commande courante.

Cette numérotation débute à 1 et est incrémentée à chaque fois qu'une ligne est créée.

Ce numéro est surtout utilisé lorsque l'on souhaite mettre à jour le texte discriminant d'un article du panier via la commande 4DBKInfoUpdate.

BskInfo ou BskLbInfo

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
				X				Alpha 30 Ko

Texte discriminant de la ligne de commande courante.

Ce texte peut être un nom, un numéro de série, une description ou toute chaîne alphanumérique. Il est fixé à l'aide de la commande 4DBKInfoSet et mis à jour à l'aide de la commande 4DBKInfoUpdate.

BskQtTotal

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
				X				Réel

Indique la quantité commandée dans la ligne de commande courante.

Cette quantité est fixée par la commande 4DBKQuantitySet.

BskPrUnitEXV / INV

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X				X				Réel

Indique le montant unitaire de l'article de la ligne de commande courante.

Lorsqu'un article est placé dans le panier, certaines conditions (remises, promotions) modifient son prix initial. Les champs BskPrUnit représentent le *prix moyen* de l'article, c'est-à-dire le montant total commandé BskPrSale pour cet article divisé par la quantité BskQtTotal.

Ce prix peut être comparé au prix de vente (PrSale ou PrItem) afin d'indiquer à l'acheteur les remises article par article dont il bénéficie (PrDelta pour un montant ou PcDelta pour un pourcentage).

- BskPrUnitEXV représente le montant hors taxes (EXcluding).
- BskPrUnitINV représente le montant avec taxes (INcluding).

BskPrSaleEXV / VAT / INV
BskPcSaleVATRate

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X				X				Réel

Indique le montant total de la ligne de commande courante.

Ce montant représente la somme des montants calculés pour une même ligne de commande. Il s'agit bien entendu du prix unitaire BskPrUnitEXV que multiplie la quantité BskQtTotal.

- BskPrSaleEXV représente le montant hors taxes (EXcluding).
- BskPrSaleVAT représente le montant des taxes.
- BskPcSaleVATRate représente le taux de taxes.
- BskPrSaleINV représente le montant avec taxes (INcluding).

BskWgTotal
BskVlTotal

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
				X				Réel

Ces champs indiquent le poids et le volume totaux de la ligne de commande courante.

- Pour BskWgTotal, il s'agit du poids unitaire WgItem que multiplie la quantité BskQtTotal.
- Pour BskVlTotal, il s'agit du volume unitaire VlItem que multiplie la quantité BskQtTotal.

Panier : ligne de frais de port

Les frais de port issus du choix d'un mode d'expédition doivent être affichés sur la commande.

Ils ne font néanmoins pas partie de la sélection B et doivent être affichés au moyen des commandes 4DBKShipping, 4DBKShippingLabel et des champs PrShippingEXV, PrShippingVAT, PrShippingINV et PcShippingVATRate.

PrShippingEXV / VAT / INV
PcShippingVATRate

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X								Réel

Ces champs indiquent le montant de la ligne de commande correspondant aux frais de port.

- PrShippingEXV représente le montant hors taxes (EXcluding).
- PrShippingVAT représente le montant des taxes.
- PrShippingVATRate représente le taux de taxes.
- PrShippingINV représente le montant avec taxes (INcluding).

Panier : totaux

PrItemsEXV / VAT / INV

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X								Réel

Ces champs indiquent le montant total des lignes de commande (donc hors frais de port).

Il s'agit de la somme de BskPrSale pour chaque ligne de commande.

- PrItemsEXV représente le montant hors taxes (EXcluding).
- PrItemsVAT représente le montant des taxes.
- PrItemsINV représente le montant avec taxes (INcluding).

PrOrderEXV / VAT / INV

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X								Réel

Ces champs indiquent le montant total de la commande (frais de port inclus).

Il s'agit de la somme de PrItems et de PrShipping.

- PrOrderEXV représente le montant hors taxes (EXcluding).
- PrOrderVAT représente le montant des taxes.
- PrOrderINV représente le montant avec taxes (INcluding).

DaOrderComment, DaOrderComment2, DaOrderComment3

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X								Texte

Ces champs indiquent les commentaires associés au panier d'une commande. Ils sont fixés par les commandes 4DBKOrderCommentSet, 4DBKOrderComment2Set et 4DBKOrderComment3Set afin d'être affichés, notamment, sur le bon de commande final.

QtItems / WgItems / VlItems

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X								Réel

Ces champs indiquent respectivement le nombre total, le poids total et le volume total des articles de la commande.

Panier : taux de taxes

Au sein d'une même commande, les articles peuvent être soumis à différents taux de taxes (biens de consommation courante, biens de luxe, exceptions douanières, etc.) ; en outre, chaque article peut lui-même être soumis à des règles de taxes différentes (taxe régionale, taxe fédérale, etc.).

Ces taux de taxes peuvent donc être représentés sous 2 formes :

- la liste des différents taux utilisés au sein d'une commande.
  Exemple :
  - pour le comté de Santa-Clara en Californie : un taux de 8,25%
  - pour la France : des taux de 5,5%, 19,6% et 33,33% en fonction de la nature des articles.
- la liste des taux de taxes par règle.
  Exemple :
  - pour le comté de Santa-Clara en Californie : 8,25% répartis en 7,25% pour la Californie et 1% pour le comté de Santa-Clara.
  - pour la France : une seule règle appelée TVA qui regroupe les 3 taux utilisés.

StOrderVATRate1 à 5

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X								Booléen

Ces champs indiquent l'existence de différents taux de taxes dans la commande.

4D Business Kit permet de gérer jusqu'à 5 taux par taxe. Pour des raisons pratiques d'affichage, il peut être utile de savoir ceux qui ont été appliqués.
Par exemple, si StOrderVATRate2 est vrai alors il faudra afficher le taux PcOrderVATRate2 et le montant PrOrderVATRate2 correspondants.

PrOrderVATRate1 à 5
PcOrderVATRate1 à 5

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X								Réel

Ces champs indiquent les taux (PrOrderVATRate) et montants (PcOrderVATRate) correspondant aux différents taux de taxes appliqués.

StOrderVATByType1 à 5

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X								Booléen

Ces champs indiquent l'application des différentes règles de taxe à la commande.

4D Business Kit permet de gérer jusqu'à 5 règles de taxe. Pour des raisons pratiques d'affichage, il peut être utile de savoir celles qui ont été appliquées.

Par exemple, si StOrderVATByType2 est vrai alors il faudra afficher le nom de la règle LbOrderVATByType2, le taux PcOrderVATByType2 et le montant PrOrderVATByType2 correspondants.

LbOrderVATByType1 à 5

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X								Alpha

Ces champs indiquent les libellés correspondant aux différentes règles de taxes appliquées.

PrOrderVATByType1 à 5
PcOrderVATByType1 à 5

Field	Query	Distinct	Sort	Basket	Set	Import	Export	Type
X								Réel

Ces champs indiquent les taux (PrOrderVATByType) et montants (PcOrderVATByType) correspondant aux différentes règles de taxes appliquées.

Note : Aucun taux ne sera fourni pour une règle où plusieurs taux sur les articles s'appliquent au sein des lignes de commande (produits taxés différement selon la nature des biens).

Clients

Les champs des clients peuvent être lus et écrits à l'aide des commandes 4DBKField et 4DBKFieldSet, des modules d'exportation et d'importation ou depuis une autre application 4D en utilisant le composant 4DBK.

Dans le cas d'une page Web, les champs ne peuvent être lus que si un client a été identifié au préalable à l'aide de la commande 4DBKCustomerLogin.

Pour créer une nouvelle fiche client depuis une page Web, il est nécessaire d'utiliser la commande de mise à jour 4DBKFieldSet en prenant soin de traiter le code client CusCode en premier (par exemple *4DBKFieldSet/CusCode=j.doe@net.com*). Le nouveau client sera alors identifié automatiquement (comme s'il l'avait été via 4DBKCustomerLogin) dès l'exécution de cette commande, de sorte que les 4DBKFieldSet suivants seront traités comme lors d'une mise à jour.

Certains comportements peuvent être également réglés à l'aide de la commande 4DBKPrefsSet.

Il existe deux méthodes d'utilisation de 4DBKFieldSet, l'une utilisant le langage Javascript, l'autre les événements de type HTTP POST.

> *Note :* Dans les tableaux suivants, la colonne Query s'utilise depuis une page Web ou depuis le composant 4DBK.

CusId

Field	Query	Set	Import	Export	Type
X			[Id]	[Id]	Alpha 14

Identifiant interne de la fiche client (par exemple : 20412382741486).

L'utilisation de cet identifiant n'est pas nécessaire dans la construction d'un site.

Ts

Field	Query	Set	Import	Export	Type
	X		[Ts]	[Ts]	Alpha 14

Indicateur de date et d'heure (*timestamp*) de la dernière modification de la fiche.

Cet indicateur, au format AAAAMMJJHHMMSS GMT (par exemple 20030418173012), est géré automatiquement, il n'est pas nécessaire de le mettre à jour.

Note : Le champ Ts peut être utilisé, dans le cadre d'une recherche depuis une autre application 4D équipée du composant 4DBK, pour télécharger les données des clients qui ont été créées ou mises à jour depuis une certaine date.

StDisabled

Field	Query	Set	Import	Export	Type
	X		[StDisabled]	[StDisabled]	Booléen

Indicateur de client radié.

Seuls les clients non radiés peuvent s'identifier via la commande 4DBKCustomerLogin.

Note : Dans les fichiers d'import/export, le champ prend la valeur 0 (faux) ou 1 (vrai).

CusCode

Field	Query	Set	Import	Export	Type
X	X	X	[CusCode]	[CusCode]	Alpha 80

Indique le code client.

Ce code identifie le client dans la commande 4DBKCustomerLogin. Il peut contenir une valeur numérique, un texte, une adresse E-mail, etc.

CusLabel

Field	Query	Set	Import	Export	Type
X	X	X	[CusLabel]	[CusLabel]	Alpha 80

Désigne le client.

Il s'agit en général de la raison sociale de l'entreprise.

CusPassword

Field	Query	Set	Import	Export	Type
X	X	X	[CusPassword]	[CusPassword]	Alpha 16

Indique le mot de passe client.

Ce code est utilisé pour identification par la commande 4DBKCustomerLogin.

CusPasswordInfo

Field	Query	Set	Import	Export	Type
X	X	X	[CusPasswordInfo]	[CusPasswordInfo]	Alpha 60

Texte permettant au client de se souvenir de son mot de passe en cas d'oubli.

Ce champ peut être utilisé avec les fonctions 4DBKCustomerMemo et 4DBKCustomerMailPwd.

CusTitle

Field	Query	Set	Import	Export	Type
X	X	X	[CusTitle]	[CusTitle]	Alpha 12

Indique le titre du client (M., Mme, Mlle, etc.).

CusFirstName

Field	Query	Set	Import	Export	Type
X	X	X	[CusFirstName]	[CusFirstName]	Alpha 40

Indique le prénom du client.

CusLastName

Field	Query	Set	Import	Export	Type
X	X	X	[CusLastName]	[CusLastName]	Alpha 60

Indique le nom du client.

CusStreetMain

Field	Query	Set	Import	Export	Type
X		X	[CusStreetMain]	[CusStreetMain]	Alpha 240

Indique la rue de l'adresse principale.

Note : Dans les fichiers d'import/export, les caractères CR (retour chariot, code ASCII 13) sont codés **^r** et les caractères ; (point virgule) sont codés **^s** (comme semicolon).

CusZIPMain

Field	Query	Set	Import	Export	Type
X	X	X	[CusZIPMain]	[CusZIPMain]	Alpha 12

Indique le code postal de l'adresse principale.

CusCityMain

Field	Query	Set	Import	Export	Type
X	X	X	[CusCityMain]	[CusCityMain]	Alpha 60

Indique la ville de l'adresse principale.

CusStateMain

Field	Query	Set	Import	Export	Type
X	X	X	[CusStateMain]	[CusStateMain]	Alpha 20

Indique l'état de l'adresse principale.

CusCountryMain

Field	Query	Set	Import	Export	Type
X	X	X	[CusCountryMain]	[CusCountryMain]	Alpha 60

Indique le pays de l'adresse principale.

CusLabelShip

Field	Query	Set	Import	Export	Type
X	X	X	[CusLabelShip]	[CusLabelShip]	Alpha 80

Désignation de l'adresse de livraison.

Il s'agit du nom du client (par exemple *M. John Doe*) ou, dans le cas d'une entreprise, du nom du client suivi de la raison sociale (par exemple *M. John Doe - SysTech Inc.*).

CusStreetShip

Field	Query	Set	Import	Export	Type
X		X	[CusStreetShip]	[CusStreetShip]	Alpha 240

Indique la rue de l'adresse de livraison.

Note : Dans les fichiers d'import/export, les caractères CR (retour chariot, code ASCII 13) sont codés **^r** et les caractères ; (point virgule) sont codés **^s** (s pour *semicolon*).

CusZIPShip

Field	Query	Set	Import	Export	Type
X	X	X	[CusZIPShip]	[CusZIPShip]	Alpha 12

Indique le code postal de l'adresse de livraison.

CusCityShip

Field	Query	Set	Import	Export	Type
X	X	X	[CusCityShip]	[CusCityShip]	Alpha 60

Indique la ville de l'adresse de livraison.

CusStateShip

Field	Query	Set	Import	Export	Type
X	X	X	[CusStateShip]	[CusStateShip]	Alpha 20

Indique l'état de l'adresse de livraison.

CusCountryShip

Field	Query	Set	Import	Export	Type
X	X	X	[CusCountryShip]	[CusCountryShip]	Alpha 60

Indique le pays de l'adresse de livraison.

CusLabelInvo

Field	Query	Set	Import	Export	Type
X	X	X	[CusLabelInvo]	[CusLabelInvo]	Alpha 80

Désignation de l'adresse de facturation.

Il s'agit du nom du client (par exemple *M. John Doe*) ou, dans le cas d'une entreprise, de la raison sociale (par exemple *SysTech Inc.*).

CusStreetInvo

Field	Query	Set	Import	Export	Type
X		X	[CusStreetInvo]	[CusStreetInvo]	Alpha 240

Indique la rue de l'adresse de facturation.

Note : Dans les fichiers d'import/export, les caractères CR (retour chariot, code ASCII 13) sont codés **^r** et les caractères ; (point virgule) sont codés **^s** (comme semicolon).

CusZIPInvo

Field	Query	Set	Import	Export	Type
X	X	X	[CusZIPInvo]	[CusZIPInvo]	Alpha 12

Indique le code postal de l'adresse de facturation.

CusCityInvo

Field	Query	Set	Import	Export	Type
X	X	X	[CusCityInvo]	[CusCityInvo]	Alpha 60

Indique la ville de l'adresse de facturation.

CusStateInvo

Field	Query	Set	Import	Export	Type
X	X	X	[CusStateInvo]	[CusStateInvo]	Alpha 20

Indique l'état de l'adresse de facturation.

CusCountryInvo

Field	Query	Set	Import	Export	Type
X	X	X	[CusCountryInvo]	[CusCountryInvo]	Alpha 60

Indique le pays de l'adresse de facturation.

CusPhone

Field	Query	Set	Import	Export	Type
X	X	X	[CusPhone]	[CusPhone]	Alpha 20

Indique le numéro de téléphone fixe du client.

CusFax

Field	Query	Set	Import	Export	Type
X	X	X	[CusFax]	[CusFax]	Alpha 20

Indique le numéro de télécopieur du client.

CusMobile

Field	Query	Set	Import	Export	Type
X	X	X	[CusMobile]	[CusMobile]	Alpha 20

Indique le numéro de téléphone mobile du client.

CusEMail

Field	Query	Set	Import	Export	Type
X	X	X	[CusEMail]	[CusEMail]	Alpha 20

Indique l'adresse E-mail du client.

CusBirthDate

Field	Query	Set	Import	Export	Type
X	X	X	[CusBirthDate]	[CusBirthDate]	Date

Indique la date de naissance du client.

CusNuVAT

Field	Query	Set	Import	Export	Type
X	X	X	[CusNuVAT]	[CusNuVAT]	Alpha 40

Numéro de TVA intracommunautaire du client (entreprise européenne uniquement).

Le numéro de TVA intracommunautaire permet à un client appartenant à l'Union Européenne de s'affranchir des taxes dans le pays vendeur pour les payer dans son pays.
Ce numéro peut être défini à l'aide de la commande 4DBKVATNumberSet.

CusPersoA1 à A5

Field	Query	Set	Import	Export	Type
X	X	X	[CusPersoA1]	[CusPersoA1]	Alpha 80

Données alphanumériques personnalisées.

Ces données sont utiles pour stocker diverses informations comme l'étage, le code d'entrée ou les préférences du client.

CusPersoN1 à N5

Field	Query	Set	Import	Export	Type
X	X	X	[CusPersoN1]	[CusPersoN1]	Réel

Données numériques personnalisées.

Ces données sont utiles pour stocker diverses informations numériques comme le chiffre d'affaire réalisé.

CusPersoT1 à T3

Field	Query	Set	Import	Export	Type
X	X	X	[CusPersoT1]	[CusPersoT1]	Alpha 30 Ko

Données texte personnalisées.

Ces données sont utiles pour stocker diverses informations longues ou pour des préférences du client.

CusDiscountA à E

Field	Query	Set	Import	Export	Type
X	X	X	[CusDiscountA]	[CusDiscountA]	Réel

Taux de remises du client par catégorie d'articles.

Reliés aux articles (TySaleMode), ces champs permettent d'attribuer à un client des remises spécifiques par article — ou plus exactement par *catégorie d'articles*. Les noms des catégories (par défaut, A, B, C, D et E) peuvent être personnalisés dans les préférences des boutiques. Chaque article peut se voir associer une catégorie de remise.

CusType

Field	Query	Set	Import	Export	Type
X	X	X	[CusType]	[CusType]	Alpha 80

Indique le type de client (par exemple *Particulier*, *Revendeur*, *Grossiste*, etc.).

CusSector

Field	Query	Set	Import	Export	Type
X	X	X	[CusSector]	[CusSector]	Alpha 80

Indique le secteur d'activité du client (par exemple, une zone géographique ou une profession).

CusPreferredShipping

Field	Query	Set	Import	Export	Type
X	X	X	[CusPreferredShipping]	[CusPreferredShipping]	Alpha 40

Indique le mode d'expédition préféré du client.
Il peut être stocké dans ce champ puis rappelé par défaut lors de commandes ultérieures de ce même client.

CusCCKey

Field	Query	Set	Import	Export	Type
X	X	X	[CusCCKey]	[CusCCKey]	Texte

Indique le numéro de carte bancaire du client.

Il est possible de stocker les coordonnées de carte bancaire du client dans les champs CC et de les rappeler lors de commandes ultérieures (commandes “one click”).

CusCCExpDate

Field	Query	Set	Import	Export	Type
X	X	X	[CusCCExpDate]	[CusCCExpDate]	Texte

Indique la date d’expiration de la carte bancaire du client.

CusCCNameOnCard

Field	Query	Set	Import	Export	Type
X	X	X	[CusCCNameOnCard]	[CusCCNameOnCard]	Texte

Indique le nom inscrit sur la carte bancaire du client.

CusCCZIPCode

Field	Query	Set	Import	Export	Type
X	X	X	[CusCCZIPCode]	[CusCCZIPCode]	Texte

Indique le code postal du client pour la vérification d’adresse des passerelles de paiement.

CusCCType

Field	Query	Set	Import	Export	Type
X	X	X	[CusCCType]	[CusCCType]	Texte

Indique le type de carte bancaire du client.

CusBAIBank

Field	Query	Set	Import	Export	Type
X	X	X	[CusBAIBank]	[CusBAIBank]	Texte

Indique la banque du compte du client.

Il est possible de stocker les coordonnées de compte bancaire du client dans les champs BAI (Bank Account Information) et de les rappeler lors de commandes ultérieures.

CusBAICity

Field	Query	Set	Import	Export	Type
X	X	X	[CusBAICity]	[CusBAICity]	Texte

Indique la ville du compte bancaire du client.

CusBAIPlace

Field	Query	Set	Import	Export	Type
X	X	X	[CusBAIPlace]	[CusBAIPlace]	Texte

Indique la ville de la banque du compte bancaire du client.

CusBAITeller

Field	Query	Set	Import	Export	Type
X	X	X	[CusBAITeller]	[CusBAITeller]	Texte

Indique le guichet de la banque du compte bancaire du client.

CusBAIAccount

Field	Query	Set	Import	Export	Type
X	X	X	[CusBAIAccount]	[CusBAIAccount]	Texte

Indique le numéro de compte bancaire du client.

CusBAIKey

Field	Query	Set	Import	Export	Type
X	X	X	[CusBAIKey]	[CusBAIKey]	Texte

Indique la clé du numéro de compte bancaire du client.

Commandes

Chaque fiche décrit une ligne de commande (par exemple 0124/1, 0124/2, 0124/3, etc.) et un ensemble de lignes de commande décrit une commande (ici 0124).

Les champs des commandes peuvent être lus à l'aide du module d'export ou depuis une autre application 4D en utilisant le composant 4DBK.

Note : Dans les tableaux suivants, la colonne Query s'utilise depuis le composant 4DBK.

Id

Field	Query	Set	Import	Export	Type
			[Id]	[Id]	Alpha 14

Identifiant interne de la fiche (par exemple *20412382737486*).

L'utilisation de cet identifiant n'est pas nécessaire dans la construction d'un site.

Ts

Field	Query	Set	Import	Export	Type
	X		[Ts]	[Ts]	Alpha 14

Indicateur de date et d'heure (*timestamp*) de la dernière modification de la fiche.

Cet indicateur, au format AAAAMMJJHHMMSS GMT (par exemple 20030418173012), est géré automatiquement, il n'est pas nécessaire de le mettre à jour.

Note : Le champ Ts peut être utilisé, dans le cadre d'une recherche depuis une autre application 4D équipée du composant 4DBK, pour télécharger les commandes créées après une certaine date.

StDone

Field	Query	Set	Import	Export	Type
	X			[StDone]	Booléen

Indique si la ligne de commande a été déclarée "traitée".

OrderCode

Field	Query	Set	Import	Export	Type
	X			[OrderCode]	Alpha 30

Indique le numéro de la ligne de commande.

OrderDate

Field	Query	Set	Import	Export	Type
	X			[OrderDate]	Date

Indique la date de création de la ligne de commande.

OrderTime

Field	Query	Set	Import	Export	Type
	X			[OrderTime]	Alpha 8

Indique l'heure de création de la ligne de commande.

CusCode

Field	Query	Set	Import	Export	Type
	X			[CusCode]	Alpha 80

Indique le code du client qui a passé la ligne de commande.

Note : Le champ CusCode peut être utilisé, dans le cadre d'une recherche depuis une autre application 4D équipée du composant 4DBK, pour télécharger les commande d'un client particulier.

CusEmail

Field	Query	Set	Import	Export	Type
	X			[CusEmail]	Alpha 80

Indique l'adresse E-mail du client qui a passé la ligne de commande.

CusAddrShip

Field	Query	Set	Import	Export	Type
				[CusAddrShip]	Texte

Indique l'adresse de livraison du client qui a passé la ligne de commande.

CusAddrInvo

Field	Query	Set	Import	Export	Type
				[CusAddrInvo]	Texte

Indique l'adresse de facturation du client qui a passé la ligne de commande.

C01

Field	Query	Set	Import	Export	Type
	X			[C01]	Alpha 30

Indique le code principal (C01) de l'article commandé.

T01

Field	Query	Set	Import	Export	Type
	X			[T01]	Alpha 80

Désignation de l'article commandé ou du mode d'expédition choisi (si StShipping vaut 1).

StShipping

Field	Query	Set	Import	Export	Type
	X			[StShipping]	Booléen

Indique si la ligne de commande correspond à des frais de port.

Note : Dans les fichiers d'export, le champ prend la valeur 0 (faux) ou 1 (vrai).

PrUnitEXV

Field	Query	Set	Import	Export	Type
	X			[PrUnitEXV]	Réel

Indique le prix public hors taxes de l'article (PrSaleEXV ou PrItemEXV).

PrDiscountEXV

Field	Query	Set	Import	Export	Type
	X			[PrDiscountEXV]	Réel

Indique le prix de vente hors taxes de l'article (BskPrUnitEXV).

RtDiscount

Field	Query	Set	Import	Export	Type
	X			[RtDiscount]	Réel

Indique le taux de remise entre le prix public et le prix de vente hors taxes de l'article.

QtItem

Field	Query	Set	Import	Export	Type
	X			[QtItem]	Entier

Indique le nombre d'articles de la ligne de commande.

PrFinalEXV / VAT / INV
RtVat

Field	Query	Set	Import	Export	Type
	X			[PrFinalEXV]	Entier

Indique le prix de vente de la ligne de commande.

- PrFinalEXV représente le montant hors taxes (EXcluding).
- PrFinalVAT représente le montant des taxes.
- RtVAT représente le taux de taxes.
- PrFinalINV représente le montant avec taxes (INcluding).

WgUnit / WgFinal

Field	Query	Set	Import	Export	Type
	X			[WgUnit]	Réel

Ces champs indiquent les poids unitaire et total de la ligne de commande.

DaOrderInfo

Field	Query	Set	Import	Export	Type
				[DaOrderInfo]	Alpha 30 Ko

Indique le texte discriminant BskInfo de la ligne de commande (par exemple un numéro de série de logiciel, le nom sur un billet d'avion, etc.).

Dans le cas d'une ligne de commande qui concerne les frais de port (StShipping = 1), ce champ est mis à jour par la commande 4DBKOrderShippingSet.

DaPaymentInfo

Field	Query	Set	Import	Export	Type
				[DaPaymentInfo]	Alpha 30 Ko

Indique le commentaire sur le mode de paiement (par exemple des coordonnées de carte bancaire).

Ce champ est mis à jour par la commande 4DBKOrderPaymentSet.

Dans le cas d'une ligne de commande qui concerne les frais de port (StShipping = 1), ce champ est vide.

DaComment / 2 / 3

Field	Query	Set	Import	Export	Type
				[DaComment]	Alpha 30 Ko

Contient des commentaires libres sur la ligne de commande (par exemple des informations particulières sur la commande ou le client).

Ces champs sont mis à jour par les commandes 4DBKOrderCommentSet, 4DBKOrderComment2Set, 4DBKOrderComment3Set.

LbVATByType1 à 5

Field	Query	Set	Import	Export	Type
	X			[LbVATByType1-5]	Alpha 80

Libellés correspondant aux différentes règles de taxes appliquées.

PcVATByType1 à 5
PrVATByType1 à 5

Field	Query	Set	Import	Export	Type
	X			[PcVATByType1-5]	Réel

Taux et montants correspondant aux différentes règles de taxes appliquées.

Note : Aucun taux ne sera fourni pour une règle où plusieurs taux sur les articles s'appliquent au sein des lignes de commande (produits taxés différement selon la nature des biens).

Règles de taxes

Les règles de taxes peuvent être importées ou exportées pour faciliter la mise en place d'un serveur 4DBK dans une région donnée. En effet, les règles de taxes sont propres à une zone géographique plus ou moins étendue, pouvant aller d'un pays à un découpage précis au sein d'une ville.

Il est possible de spécifier pour chaque paramètre de la zone (pays, états, codes postaux, villes) une liste de lieux à inclure et une liste de lieux à exclure.

Des particularités permettent de gérer le numéro de TVA intracommunautaire instauré par l'Union Européenne.

Champ	Import	Export	Description
LbTax	[LbTax]	[LbTax]	Nom de la règle
RtTaxA à E	[RtTaxA - E]	[RtTaxA - E]	Taux pour chaque catégorie
DaCountryList	[DaCountryList]	[DaCountryList]	Liste des pays d'application de la règle
DaStateList	[DaStateList]	[DaStateList]	Liste des états d'application de la règle
DaZIPList	[DaZIPList]	[DaZIPList]	Liste des codes postaux d'application de la règle
DaCityList	[DaCityList]	[DaCityList]	Liste des villes d'application de la règle
StVATNumberSet	[StVATNumberSet]	[StVATNumberSet]	La règle ne doit pas s'appliquer lorsqu'un numéro de TVA intracommunautaire a été défini via la commande 4DBKVATNumberSet
CdCountry	[CdCountry]	[CdCountry]	Code pays de départ des marchandises

Modes d'expédition

Les modes d'expédition peuvent être importés ou exportés pour faciliter la mise en place d'un serveur 4DBK dans une région donnée.

Les modes d'expédition vont générer les frais de port sur la commande. Ils peuvent s'appliquer, comme les règles de taxe, à une zone géographique plus ou moins étendue, pouvant aller d'un pays à un découpage précis au sein d'une ville. Il est possible de spécifier pour chaque paramètre de la zone (pays, états, codes postaux, villes) une liste de lieux à inclure et à exclure.

Les modes d'expédition sont analysés dès que le client s'identifie ou ajoute un article à son panier. Il est alors possible de présenter au client la liste des modes d'expédition envisageables (par exemple plusieurs vitesses d'acheminement) afin qu'il en sélectionne un.

Il est possible de fixer un seuil de montant de commande au-delà duquel les frais de port sont gratuits.

Les modes d'expédition sont définissables par nombre d'articles (montant du port pour le premier article, montant du port pour les suivants), par poids (une table fait correspondre des poids et des montants de port), par prix (une table fait correspondre des montants de commandes et des montants de port) ou par formule (il est possible de définir un calcul en fonction par exemple des catégories d'articles que le panier du client contient).

Champ	Import	Export	Description
Label	[Label:valeur]	[Label:valeur]	Désignation du mode d'expédition (langue1$$langue2$$langue3$$langue4)
Type	[Type:valeur]	[Type:valeur]	Type de mode (A = par article, W = par poids, P = par prix, F = par formule)
Code	[Code:valeur]	[Code:valeur]	Code du mode, utile pour tester une valeur non dépendante de la langue (par exemple : FX_FAST)
Countries	[Countries:valeur]	[Countries:valeur]	Liste des pays d'application du mode
State	[State:valeur]	[State:valeur]	Liste des états d'application du mode
ZIP	[ZIP:valeur]	[ZIP:valeur]	Liste des codes postaux d'application du mode
Cities	[Cities:valeur]	[Cities:valeur]	Liste des villes d'application du mode
TyTax	[TyTax:valeur]	[TyTax:valeur]	Type de taxe (fonctionnement identique à TyTax des articles)

Champ	Import	Export	Description
Comment	[Comment:valeur]	[Comment:valeur]	Zone d'information client (langue1$$langue2$$langue3$$langue4)
FreeOfCharge	[FreeOfCharge: valeur]	[FreeOfCharge: valeur]	Indique si le mode devient gratuit au-delà d'un certain montant de commande (0=0 : non actif, 1=montant : gratuit au-delà de montant)
Formula	[Formula:valeur]	[Formula:valeur]	Formule d'application si le type est F
V01 à V24	[Vnn:valeur1/ valeur2]	[Vnn:valeur1/ valeur2]	Elément de table pour les types A, W et P (jusqu'à valeur1, appliquer valeur2)

Boutiques

Vous pouvez afficher différentes informations concernant les boutiques, notamment dans les pages d'administration.

Champ	Field	Description
StoreMainName	X	Libellé de l'entité juridique de la boutique
StoreID	X	Code administratif de la boutique
StoreMainAddr	X	Adresse de l'entité juridique de la boutique
StoreMainPhone	X	Téléphone de l'entité juridique de la boutique
StoreMainFax	X	Fax de l'entité juridique de la boutique
StoreMainEMail	X	E-mail de l'entité juridique de la boutique
StoreMainWeb	X	Site Web de l'entité juridique de la boutique
StoreShopName	X	Libellé de la boutique
StoreShopAddr	X	Adresse de la boutique
StoreShopPhone	X	Téléphone de la boutique
StoreShopFax	X	Fax de la boutique
StoreShopEMail	X	E-mail de la boutique
NbItems	X	Nombre d'articles au catalogue (peut également être obtenu par 4DBKQuerySet/*all)

Connexions Web

Il s'agit des sessions uniques des utilisateurs Web.

Chaque ligne, inscrite lorsque la session se termine, comprend la synthèse de l'activité de l'utilisateur.

En retraitant ces informations dans une application spécifique, il est possible de fournir des diagrammes d'activité par période, par charge, par intérêt, etc.

Pour l'utilisation d'outils génériques (par exemple WebTrend), il sera nécessaire d'utiliser le fichier *Logweb.txt* qui est au format CLF.

Champ	Import	Export	Description
[TsLogStart]		X	Indicateur de date et heure GMT (*TimeStamp*) de début de la session (format AAAAMMJJHHMMSS)
[TsLogEnd]		X	Indicateur de date et heure GMT (*TimeStamp*) de fin de la session (format AAAAMMJJHHMMSS)
[VaTimeConnected]		X	Temps de durée de la session en minutes
[AdIP]		X	Adresse IP de l'utilisateur
[LbName]		X	Nom DNS de l'utilisateur (il faut l'avoir décodé automatiquement ou manuellement avant l'export)
[LbBrowser]		X	Désignation du navigateur
[LbReferer]		X	URL d'origine
[NbPages]		X	Nombre de pages vues
[NbObjects]		X	Nombre d'objets servis
[VaVolume]		X	Volume transféré en Mo
[VaCPUTime]		X	Minutes d'utilisation du processeur à 100%
[StFavorite]		X	L'utilisateur a ajouté ce site à ses favoris (IE Windows uniquement)

Requêtes Web

Il s'agit de la synthèse quotidienne de toutes les requêtes.

A chaque ligne correspond une journée et une boutique. Les requêtes qui ne concernent aucune boutique (virus, erreurs) sont classées sous le code **????**.

En retraitant ces informations dans une application spécifique, il est possible de fournir des diagrammes d'activité par période, par charge, par intérêt, etc.

Champ	Import	Export	Description
[DtDay]		X	Jour d'activité au format AAAAMMJJ
[NbUniqueVisitors_1_page]		X	Nombre quotidien d'utilisateurs uniques qui ont vu une seule page ("pas intéressés")
[NbUniqueVisitors_2_to_3_pages]		X	Nombre quotidien d'utilisateurs uniques qui ont vu 2 ou 3 pages (intéressés par le sujet mais pas par le site)
[NbUniqueVisitors_4_and_more_pages]		X	Nombre quotidien d'utilisateurs uniques qui ont vu 4 pages et plus (intéressés par le site)
[NbPages]		X	Nombre quotidien de pages vues
[NbObjects]		X	Nombre quotidien d'objets servis
[VaVolume]		X	Volume quotidien transféré en Mo
[VaCPUTime]		X	Minutes par jour d'utilisation du processeur à 100%

Divers

Champ	Field	Description
URL	X	URL courant (ex : /Test_Site/WebPagesUS/netparms.htm&&2BxBlu0obDuox2NvxMpoLGxolobo24kaluXauWkaTa)
Path	X	Chemin d'accès courant aux pages (ex : /Test_Site/WebPagesUS/)
Page	X	Page courante (ex : netparms.htm)
Host	X	Champ Host de la requête (ex : 192.168.3.104:8080)
Host,LST&1&:	X	Format pour obtenir la partie haute du champ Host, l'adresse IP (par ex : 192.168.3.104)
Host,LST&2&:	X	Format pour obtenir la partie basse du champ Host, le port (par ex : 8080)
SessionID	X	Identifiant de la session courante (par ex : %26%262BxBlu0obDuox2NvxMpoLGxolobo2bbBeJ1gS__a_kkaWkuaXXka_WTaWkXaq_Takkla6SkaTXWauu4akluaXXqaXka)
SessionID,X	X	Identifiant de la session courante commençant par __ au lieu de && (%26%26) pour la comptabilité avec certaines passerelles de paiement
Referer	X	Champ Referer de la requête (par ex : http://192.168.3.104:8080/test)
IP	X	Adresse IP de laquelle provient la requête
CdCurrency	X	Code devise de la boutique
RtCurrency	X	Taux de devise de la boutique par rapport à une autre devise (si la boutique est en dollars US, 4DBKField/RtCurrency.EUR,###0.00000 affichera le taux comparé à l'euro)
ErrorCode	X	Code erreur de la dernière opération

20 Formats d’affichage et fonctions

4D Business Kit permet de définir le **format d’affichage** des données présentes dans vos pages. Par exemple, il est possible d’afficher un nombre dans un format monétaire ou de limiter le nombre de caractères affichés dans un champ. Les formats possibles et la syntaxe à employer dépendent du type de données utilisées : numérique, date ou texte.

En outre, 4D Business Kit propose des **fonctions** permettant d’obtenir des valeurs issues d’un traitement de la valeur initiale. Par exemple, vous pouvez extraire du texte “Bonjour” la chaîne de caractères “jour”.

Les formats d’affichage et les fonctions peuvent être utilisés avec diverses commandes de 4D Business Kit, notamment 4DBKVar et 4DBKField (paramètre *format)*.

Valeurs numériques

Formats d’affichage

Le format d’affichage d’un champ numérique peut être défini à l’aide des caractères **#** (dièse) . (point) et , (virgule) sous les formes ### ### ##0,00 par exemple (la virgule comme séparateur décimal), ### ##0.00 (le point comme séparateur décimal) ou ###,###,##0.00 (la virgule comme séparateur de milliers et le point comme séparateur décimal).

Si des ambiguïtés peuvent survenir entre la virgule et le point comme type de séparateur sur des versions anglaises ou françaises de 4D Business Kit, il conviendra d’utiliser les caractères [et] pour séparer les parties entière et décimale du nombre sous la forme [partie entière][partie décimale].

En préfixant le format d'affichage de la lettre N, tous les espaces du format seront insécables et un montant ne sera pas coupé en deux au niveau d'un séparateur de milliers s'il est en fin de ligne.

Si l'on souhaite afficher un montant numérique dans une autre devise que celle de la boutique, il faut suffixer le champ de la mention .DEV (DEV représente le code de la devise).

▼ Par exemple, pour la variable VR01 contenant 1 233,14159265 :

```
4DBKVar/VR01,### ##0,00 FRF : affiche 1 233,14 FRF
4DBKVar/VR01,USD ###,##0.00 : affiche USD 1,233.14
4DBKVar/VR01,### ##0.0000 : affiche 1 233.1416
4DBKVar/VR01,00000,000000000 : affiche 01233,141592650
4DBKVar/VR01,[### ##0][,00] : affiche 1 233,14
4DBKVar/VR01,[###,##0][.0000] : 1,233.1416
4DBKVar/VR01,[### ##0] : affiche 1 233
4DBKVar/VR01,[###,##0] : affiche 1,233
4DBKVar/VR01,[][000] : affiche 141
4DBKVar/VR01,[########0] . [][00] : affiche 1223 . 14
4DBKVar/VR01,[#####0][00] : affiche 122314
4DBKVar/VR01,N### #####0,00 : affiche 1 223,14 avec des
     espaces insécables ( )
4DBKVar/VR01.GBP,###,##0.00 : si la devise de la
     boutique est le dollar US, VR01.GBP affichera le
     montant en livres anglaises.
```

Fonctions

Vous pouvez utiliser les fonctions suivantes avec les valeurs de type numérique :

F (*Full*)

Retourne un booléen permettant de tester si un nombre n'est pas nul. La valeur retournée est TRUE (nombre non nul) ou FALSE (nombre nul).

B (*Boolean*)

Retourne 0 ou 1 suivant que le nombre est nul ou non. La valeur retournée est 1 (nombre non nul) ou 0 (nombre nul).

▼ Par exemple, pour la variable VR01 contenant 1 233,14159265 :

```
4DBKVar/VR01,F : affiche true car VR01 n'est pas nul
     (false sinon)
4DBKVar/VR01,B : affiche 1 car VR01 n'est pas nul (0
     sinon)
```

Valeurs de type Date

Formats d'affichage

Pour un champ date, le format d'affichage peut être défini en utilisant d,m et y sous toutes les combinaisons :

- dd : le jour de la date sur 2 chiffres.
- mm : le mois de la date sur 2 chiffres.
- yy : l'année de la date sur 2 chiffres.
- yyyy : l'année de la date sur 4 chiffres.

Pour obtenir le nombre de secondes écoulées depuis une date donnée, le format est ES (pour *Elapsed Seconds*) suivi de la date au format AAAAMMJJ.

▼ Par exemple, si la date D01 est 03/12/2001 :

```
4DBKField/D01,dd/mm/yy : 03/12/01
4DBKField/D01,mm-yyyy : 12/2001
4DBKField/D01,mm dd yyyy : 12 03 2001
4DBKField/D01,yyyy : 2001
4DBKField/D01,mm yyyy (dd) : 12-2001 (03)
4DBKField/D01,ES19700101 : 1007337600 (secondes
     écoulées depuis le 01 01 1970)
```

Fonctions

Vous pouvez utiliser les fonctions suivantes avec les valeurs de type Date :

E (*Empty*)

Retourne un booléen permettant de tester si la date est nulle (00/00/00). La valeur retournée est TRUE (date nulle) ou FALSE (date non nulle).

F (*Full*)

Retourne un booléen permettant de tester si la date n'est pas nulle (00/00/00). La valeur retournée est TRUE (date non nulle) ou FALSE (date nulle).

B (*Boolean*)

Retourne 0 ou 1 suivant que la date est nulle (00/00/00) ou non. La valeur retournée est 1 (date non nulle) ou 0 (date nulle).

▼ Par exemple, si la date est 25/04/01 :

```
4DBKToday/E : False
4DBKToday/F : True
4DBKToday/B : 1
```

Valeurs de type Heure

Formats d'affichage

Pour une valeur de type heure (voir commandes "4DBKNow", page 163 et "4DBKTimeOffSet", page 164), le format d'affichage peut être défini en utilisant hh,mm et ss sous toutes les combinaisons.

- hh = l'heure sur 2 chiffres.
- mm = les minutes sur 2 chiffres.
- ss = les secondes sur 2 chiffres.

En préfixant le format du mot-clé GMT, on obtient l'heure au format GMT.

▼ Par exemple, si l'heure est 18:28:35 :

```
4DBKNow,hh:mm:ss : 18:28:35
4DBKNow,hh heures mm minutes : 18 heures 28 minutes
4DBKNow,GMThhHmm : 16H28
```

Pour obtenir l'heure courante en secondes depuis une heure donnée, le format est ES (pour *Elapsed Seconds*) ou GMTES pour le même résultat mais au format GMT (commande concernée : 4DBKNow).

▼ Par exemple, si l'heure est 11:23:57 (09:23:57 GMT) :

```
4DBKNow,ES080000 : 12237 secondes depuis 8H00
     (différence entre 11:23:57 et 08:00:00)
4DBKNow,GMTES080000 : 5037 secondes depuis 8H00
     (différence entre 09:23:57 et 08:00:00)
4DBKNow,ES140000 : -9363 secondes avant 14H00
     (différence entre 11:23:57 et 14:00:00)
4DBKNow,GMTES140000 : -16563 secondes avant 14H00
     (différence entre 09:23:57 et 14:00:00)
```

► Pour obtenir le nombre de secondes depuis le 1er janvier 1970 (certaines passerelles de paiement l'exigent) :

```
4DBKVarSet/VL01=4DBKToday,ES19700101
4DBKVarSet/VL01+=4DBKNow,ES000000
```

Valeurs de type Texte

Formats d'affichage

La syntaxe des formats d'affichage des champs de type texte est la suivante :

[Format][&Encodage][&NbCars][&NbMots]

OU BIEN :

[Format][:Encodage][:NbCars][:NbMots]

> *Note :* Le caractère **&** est utilisé dans les exemples de cette documentation. Pour plus d'informations sur l'emploi du caractère **:**, reportez-vous au paragraphe "Alternative de syntaxe & ou :", page 231.

Format

Le paramètre **Format** accepte les valeurs suivantes :

- **L** (Lower) : convertit le champ en minuscules
- **U** (Upper) : convertit le champ en majuscules
- **O** (Original) : le champ n'est pas converti en ISO
- **BR** (
) : les caractères CR (retours chariot) sont convertis en BR

Encodage

Le paramètre **Encodage** accepte les valeurs suivantes :

- **P** (Plus) : remplace les espaces par des +
- **L** (Link) : convertit le champ en URL ISO

NbCars

Le paramètre **NbCars** (nombre maximum de caractères) accepte les valeurs suivantes :

- **NN** : limite le champ à NN caractères
- **NN+** : limite le champ à NN caractères avec "..." (trois points) si le champ a été tronqué.

NbMots

Le paramètre **NbMots** (nombre maximum de mots) accepte les valeurs suivantes :

- **NN** : limite le champ à NN mots
- **NN+** : limite le champ à NN mots avec "..." (trois points) si le champ a été tronqué.

Exemples

Pour le champ T01 contenant le texte "Clé à Molette" :

```
4DBKField/T01,L : clé à molette
4DBKField/T01,U : CLE A MOLETTE
4DBKField/T01,&P : Clé+à+Molette
4DBKField/T01,U&P : CLE+A+MOLETTE
4DBKField/T01,&L : Cl%8E+%88+Molette
4DBKField/T01,&&5 : clé à
4DBKField/T01,&&5+ : clé à...
4DBKField/T01,&&&1 : clé
4DBKField/T01,&&&1+ : clé...
4DBKField/T01,U&L&&1+ : CL%8E%C9...
```

Fonctions

4D Business Kit vous permet d'appliquer diverses fonctions aux champs de texte. La syntaxe à employer est du type :

Fonction[**&Option1**][**&Option2**]

OU BIEN :

Fonction[**:Option1**][**:Option2**]

> *Note :* Le caractère **&** est utilisé dans les exemples de cette documentation. Pour plus d'informations sur l'emploi du caractère **:**, reportez-vous au paragraphe "Alternative de syntaxe & ou :", page 231.

E (*Empty*)

Retourne un booléen permettant de tester si le champ est vide. La valeur retournée est TRUE (0 caractère) ou FALSE (n caractères).

F (*Full*)

Retourne un booléen permettant de tester si le champ n'est pas vide. La valeur retournée est TRUE (n caractères) ou FALSE (0 caractère).

B (*Boolean*)

Retourne 0 ou 1 suivant que le champ est vide ou non. La valeur retournée est 1 (n caractères, champ non vide) ou 0 (0 caractère, champ vide).

S (*Size*) ou LEN (*Length*)

Retourne le nombre de caractères contenus dans le champ.

SUB (*Substring*)

Retourne une sous-chaîne du champ.
La syntaxe à employer est du type SUB&A[&B] où A = position du premier caractère et B = nombre de caractères à extraire. Si B est omis, la fonction retourne tous les caractères jusqu'à la fin du champ.

POS (*Position*)

Retourne la position d'une chaîne de caractères dans le champ.
La syntaxe à employer est du type POS&A où A = chaîne de caractères à rechercher.

LST (*List*)

Retourne le $n^{ième}$ mot d'une chaîne de caractères contenant une liste de mots. La syntaxe à employer est du type LST&A[&B] où A = indice du mot à retourner et B = séparateur de mots.

Si B est omis, 4D Business Kit considère la virgule comme séparateur de mots.

Vous pouvez utiliser tout type de caractère(s) comme séparateur. Toutefois, certains caractères "réservés" ou invisibles (par exemple l'espace ou le retour chariot) doivent être désignés à l'aide des mots-clés suivants :

- SPACE : espace
- COMMA : virgule (séparateur par défaut)
- CR : retour chariot
- LF : retour à la ligne
- CRLF : retour chariot + retour à la ligne
- AMP : & ("et commercial")
- COLON : **:** ("deux-points")
- BR : retour à la ligne au format HTML

ISONUM, ISOCD2, ISOCD3 (*Codes ISO*)

- ISONUM : retourne le numéro ISO d'un pays
- ISOCD2 : retourne le code ISO à deux lettres d'un pays
- ISOCD3 : retourne le code ISO à trois lettres d'un pays

Ces fonctions doivent être utilisées avec les champs contenant un nom de pays, c'est-à-dire :

- CusCountryMain (libellé du pays de l'adresse principale du client)
- CusCountryInvo (libellé du pays de l'adresse de facturation du client)
- CusCountryShip (libellé du pays de l'adresse de livraison du client)

Exemples

(1) Pour le champ T01 contenant le texte "Clé à Molette" :

```
4DBKField/T01,E : retourne false
4DBKField/T01,F : retourne true
4DBKField/T01,B : retourne 1
4DBKField/T01,S : retourne 13
4DBKField/T01,LEN : retourne 13
```

(2) Pour la variable VT01 contenant la chaîne "Tarte à la rhubarbe" :

```
4DBKVar/VT01,SUB&12 : retourne rhubarbe
4DBKVar/VT01,SUB&12&3 : retourne rhu
4DBKVar/VT01,POS&rhubarbe : retourne 12
4DBKVar/VT01,POS&a : retourne 2
4DBKVar/VT01,LST&2&SPACE : retourne à
4DBKVar/VT01,LST&1&b : retourne Tarte à la rhu
```

(3) Pour la variable VT01 contenant la chaîne "Lundi,Mardi,Mercredi" :

```
4DBKVar/VT01,LST&3&COMMA : retourne Mercredi
4DBKVar/VT01,LST&2 : retourne Mardi
```

(4) Pour le champ 4DBKField/CusStreetMain contenant les lignes suivantes :

```
Résidence des Lilas
60, rue Paul Hochon
Bât. C
```

... vous pouvez afficher la deuxième ligne à l'aide de l'instruction :

```
<--!4DBKField/CusStreetMain,LST&2&CR--> //affiche 60,
rue Paul Hochon
```

(5) Le champ CusCountryMain contient "France" :

```
4DBKField/CusCountryMain,ISONUM : retourne 250
4DBKField/CusCountryMain,ISOCD2 : retourne FR
4DBKField/CusCountryMain,ISOCD3 : retourne FRA
```

Alternative de syntaxe & ou :

4D Business Kit accepte le caractère “deux-points” (**:**) comme alternative au **&**. En effet, dans certains cas (par exemple avec la commande 4DBKIf), le caractère & peut être interprété de manière incorrecte.

Par exemple, il est possible d’écrire LST&6&SPACE ou LST:6:SPACE pour désigner le 6e caractère d’une liste de mots utilisant l’espace comme séparateur.

Dans le cas où le caractère **:** doit être utilisé comme paramètre (par exemple pour désigner un séparateur), le mot-clé COLON doit être utilisé. Pour reprendre l’exemple précédent, LST:6:COLON désigne le 6e caractère d’une liste de mots utilisant les deux-points comme séparateurs.

A Liste des erreurs

Cette annexe fournit décrit les messages d'erreurs pouvant être retournés par le serveur 4D Business Kit lors de l'exécution des instructions placées dans les pages HTML.

4D Business Kit affiche ces messages d'erreurs dans le navigateur. Le code à l'origine de l'erreur peut être examiné à l'aide des outils de débogage intégrés (cf. chapitre 3, "Aide au débogage du code 4D Business Kit" page 27).

Erreur	Description
can't launch a distinct filter on this field	on ne peut pas trouver de valeurs distinctes sur ce champ
command not found	la commande 4DBK utilisée comme paramètre n'existe pas
cursor parameter not found	la valeur du curseur n'a pas été spécifiée
invalid query parameter	le critère de recherche demandé est invalide
invalid selection parameter	la sélection spécifiée n'est pas 1,2... 5 ou B
layout not found	aucun masque n'a été trouvé pour la boutique courante
query parameter not found	aucun critère de recherche n'a été spécifié
field parameter not found	aucun champ n'a été spécifié
record number out of range	le nombre d'enregistrements demandé va au-delà de la taille de la sélection courante
record parameter not found	aucun numéro ou nombre d'enregistrements n'a été spécifié
selection parameter not found	le code sélection n'a pas été spécifié
sort parameter not found	aucun critère de tri n'a été spécifié
store parameter not found	le code 4 caractères de la boutique n'a pas été spécifié
this field can't be queried	on ne peut pas effectuer de recherche sur ce champ
this field can't be sorted	on ne peut pas effectuer de tri sur ce champ
unknown selection	il manque une instruction 4DBKSelectionSet pour spécifier la sélection courante

Erreur	Description
unknown store	il manque une instruction 4DBKStoreSet pour spécifier la boutique courante
unknown store in this database	la boutique spécifiée n'existe pas dans cet Espace multiboutique
unknown field	le champ spécifié n'existe pas
value parameter not found	aucune valeur n'a été spécifiée
this field can't be displayed	le champ spécifié ne peut être affiché
this field does not have a multiple data structure	il n'est pas possible d'effectuer des opérations concernant les données multiples sur ce champ
index parameter not found	il manque un numéro d'ordre
code parameter not found	aucun code n'a été spécifié
C01 missing	aucun code C01 n'a été spécifié
item xxx unknown	l'article xxx n'existe pas dans la base de données
invalid quantity parameter - should be +2,4 or =3	le paramètre n'a pas la forme correcte
quantity parameter not found	aucune quantité n'a été spécifiée
customer is not logged yet	aucun client n'est encore identifié
var parameter not found	aucune variable n'a été spécifiée
invalid var	la variable spécifiée est incorrecte
invalid operator	l'opérateur spécifié est incorrect
this field can't be displayed in a menu	il n'est pas possible d'afficher ce champ dans un menu
invalid preference parameter	ce paramètre de préférence est invalide
preference parameter not found	aucun paramètre de préférence n'a été spécifié

Index

Symboles

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

Z