# Guide de référence FormCalc

Adobe® LiveCycle® Designer

Version 8.0

Adobe® LiveCycle® Designer 8.0 - Guide de référence FormCalc pour Microsoft® Windows®
Octobre 2006



Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA.

# Sommaire

# Préface

Adobe® LiveCycle® Designer 8.0 offre un ensemble d'outils qui permet aux développeurs de formulaires de créer des documents professionnels intelligents. Les développeurs peuvent créer des formulaires en y intégrant des fonctions de calcul et des scripts pour que leurs destinataires bénéficient d'un confort d'utilisation inégalé. Par exemple, vous avez la possibilité d'utiliser des calculs simples pour assurer la mise à jour automatique du total d'un bon de commande ou bien des scripts complexes pour adapter la présentation d'un formulaire à la langue et au pays de l'utilisateur.

Pour faciliter la création de fonctions de calcul, LiveCycle Designer fournit FormCalc aux utilisateurs. FormCalc est un langage de calcul simple créé par Adobe, qui s'inspire des tableurs traditionnels. FormCalc est simple à utiliser et accessible aux développeurs qui n'ont aucune expérience dans la rédaction de scripts. De nombreuses règles et conventions sont communes aux autres langages de script pour que les développeurs expérimentés mettent à profit leurs connaissances lorsqu'ils utilisent FormCalc.

## Contenu du guide

Ce guide est conçu pour les développeurs de formulaires utilisant LiveCycle Designer, qui souhaitent intégrer des fonctions de calcul FormCalc dans leurs formulaires. Le présent guide est un ouvrage de référence sur les fonctions FormCalc, lesquelles sont regroupées en plusieurs chapitres correspondant à leur catégorie. Il propose également une introduction au langage FormCalc et à la création de blocs d'expressions FormCalc.

## A qui s'adresse ce guide ?

Ce guide s'adresse aux développeurs de formulaires qui souhaitent utiliser le langage FormCalc pour enrichir leurs conceptions de formulaire créées dans LiveCycle Designer avec des fonctions de calcul.

## Documentation connexe

Pour plus de détails sur l'utilisation des fonctions de calcul FormCalc dans des formulaires, voir *Création de calculs et de scripts* dans l'*Aide de LiveCycle Designer*.

Pour obtenir des informations plus techniques sur FormCalc, consultez la documentation *Adobe XML Forms 2.1 Specification* disponible sur le site Web Adobe Solutions Network (ASN).

# 1 Introduction à FormCalc

FormCalc est un langage de calcul simple mais puissant, fondé sur les tableurs traditionnels. Il permet de créer des formulaires, rapidement et efficacement, sans exiger la maîtrise des techniques ou des langages de script traditionnels. A l'aide de quelques fonctions intégrées, les nouveaux utilisateurs de FormCalc peuvent ainsi créer rapidement des formulaires pour éviter aux utilisateurs finaux d'effectuer certaines tâches fastidieuses, telles que les calculs, les validations et autres vérifications. De cette façon, un développeur de formulaires est capable d'intégrer une certaine intelligence à un formulaire au moment de la conception, de façon à ce que le formulaire interactif créé réagisse selon les données entrées.

FormCalc proposent de nombreuses fonctions portant sur des domaines variés, tels que les mathématiques, les dates et heures, les chaînes, la finance, la logique et le Web. Ces catégories représentent les types de données habituellement rencontrées dans les formulaires ; les fonctions permettent de manipuler rapidement et facilement les données à des fins utiles.

## A propos des scripts dans LiveCycle Designer

Dans LiveCycle Designer, FormCalc est le langage de script par défaut à tous les emplacements de script ; le langage JavaScript™ constitue l'autre choix possible. Les scripts s'appliquent aux différents événements qui accompagnent chaque objet d'un formulaire. Vous pouvez combiner les langages FormCalc et JavaScript dans vos formulaires interactifs. Toutefois, si vous utilisez un processus serveur, tel que LiveCycle Forms, pour créer des formulaires qui seront affichés dans un navigateur Web, les scripts FormCalc de certains événements rattachés aux objets des formulaires ne sont pas reflétés dans les formulaires HTML. Cette fonctionnalité permet d'éviter des erreurs de navigateur Internet lorsque les utilisateurs manipulent les formulaires remplis.

# 2 Section de référence sur le langage

## Blocs fonctionnels

Le langage FormCalc consiste en un certain nombre de blocs fonctionnels qui composent les expressions FormCalc. Chaque expression FormCalc est une séquence composée d'une combinaison de blocs fonctionnels.



### Littéraux

Les littéraux sont des valeurs constantes qui forment la base de toutes les valeurs transmises à FormCalc pour traitement. Il existe deux grands types de littéraux : les nombres et les chaînes.

**Littéraux numériques**

Un littéral numérique est une séquence composée surtout de chiffres ; elle comprend un ou plusieurs des caractères suivants : un entier, un point décimal, une fraction, un indicateur d'exposant (« e » ou « E ») et une valeur d'exposant avec signe facultatif. Voici quelques exemples de littéraux numériques :

- -12
- 1.5362
- 0.875
- 5.56e-2
- 1.234E10

Dans un littéral numérique, il est possible d'omettre la partie de l'entier ou de la fraction, mais non les deux. En outre, dans la partie de la fraction, on peut aussi omettre le point décimal ou la valeur de l'exposant, mais non les deux.

Tous les littéraux numériques sont convertis en interne en valeurs binaires 64 bits IEEE. Cependant, puisque les valeurs IEEE ne peuvent représenter qu'une quantité finie de nombres, certaines valeurs n'ont pas de représentation en fraction binaire. Cette situation est comparable au fait que certaines valeurs, comme 1/3, n'ont pas de représentation précise en fraction décimale (la valeur décimale aurait besoin d'un nombre infini de décimales pour être tout à fait exacte).

Ces valeurs qui ne possèdent pas d'équivalent en fraction binaire sont généralement les littéraux numériques comptant plus de 16 chiffres significatifs avant l'exposant. FormCalc arrondit ces valeurs à la valeur 64 bits IEEE représentable la plus proche selon les normes de l'IEEE. Par exemple, la valeur :

```
123456789.012345678
```

est arrondi à la valeur (la plus près) :

```
123456789.01234567
```

Autre exemple, le littéral numérique :

```
9999999999999999
```

est arrondi à la valeur (la plus près) :

```
100000000000000000
```

Cette façon de faire entraîne parfois des résultats surprenants. FormCalc possède une fonction, Round, qui renvoie le nombre indiqué arrondi au nombre indiqué de décimales. Lorsque le nombre indiqué est exactement à mi-chemin entre deux nombres représentables, il est arrondi au plus loin de zéro. Autrement dit, le nombre est arrondi à la hausse s'il est positif et à la baisse s'il est négatif. Considérez l'exemple suivant :

```
Round(0.124, 2)
```

renvoie `0.12`,

et

```
Round(.125, 2)
```

renvoie `0.13`.

D'après cette convention, vous pourriez penser que :

```
Round(0.045, 2)
```

renvoie `0.05`.

Cependant, la norme IEEE 754 stipule que le littéral numérique `0.045` reçoit la valeur approximative `0.0449999999999999`. Cette approximation est plus près de `0.04` que de `0.05`. C'est pourquoi :

```
Round(0.045, 2)
```

renvoie `0.04`.

Cette façon de faire est conforme à la norme IEEE 754.

Les valeurs 64 bits IEEE reconnaissent les représentations comme NaN (« not a number », n'est pas un nombre), +Inf (infini positif) et -Inf (infini négatif). FormCalc ne les gère pas ; toute expression dont l'évaluation a pour résultat NaN, +Inf ou -Inf entraîne une exception d'erreur, transmise au reste de l'expression.

### Littéraux chaînes

Un littéral chaîne est une séquence constituée de tout caractère Unicode, précédée et suivie de guillemets. Par exemple :

```
"The cat jumped over the fence."
"Number 15, Main street, California, U.S.A"
```

Le littéral chaîne `""` définit une séquence vide (sans caractères) appelée chaîne vide.

Pour mettre un élément entre guillemets au sein d'un littéral chaîne, vous devez faire précéder le guillemet d'une barre oblique inverse (\). Par exemple :

```
"The message reads: ""Warning: Insufficient Memory"""
```

Tous les caractères Unicode ont une séquence d'échappement de 6 caractères équivalente, composée de `\u` suivie de quatre chiffres hexadécimaux. Dans tout littéral chaîne, il est possible d'exprimer n'importe quel caractère, y compris les caractères de commande, au moyen de la séquence d'échappement Unicode équivalente. Par exemple :

```
"\u0047\u006f\u0066\u0069\u0073\u0068\u0021"
"\u000d" (carriage return)
"\u000a" (newline character)
```

## Opérateurs

FormCalc inclut un certain nombre d'opérateurs : unaires, multiplicatifs, additifs, relationnels, logiques, d'égalité et d'affectation.

Plusieurs opérateurs FormCalc ont un mot-clé mnémonique équivalent. Ces mots-clés sont utiles lorsque des expressions FormCalc sont incorporées dans du texte source HTML et XML, où les symboles « inférieur à » (<), « supérieur à » (>), et « et commercial » (&) ont des significations prédéfinies, et doivent donc être obtenus par des séquences d'échappement. Le tableau suivant dresse la liste de tous les opérateurs FormCalc accompagnés des formes symboliques et, le cas échéant, mnémoniques.

| **Type d'opérateur** | **Représentations** |
|---|---|
| Addition | + |
| Division | / |
| Egalité | == eq<br><> ne |
| ET logique | & and |
| OU logique | \| or |
| Multiplication | * |
| Expression relationnelle | < lt (« less than », inférieur à)<br>> gt (« greater than », supérieur à)<br><= le (« less than or equal to », inférieur ou égal à)<br>>= ge (« greater than or equal to », supérieur ou égal à) |
| Soustraction | - |
| Unaire | -<br>+<br>not |

# Commentaires

Les commentaires sont des sections de code que FormCalc n'exécute pas. Les commentaires contiennent normalement des renseignements ou des directives qui expliquent l'utilité d'un certain fragment de code. Au moment de l'exécution, FormCalc ne tient pas compte des informations stockées en commentaires.

Vous pouvez indiquer un commentaire en entrant soit un point-virgule (;), soit une paire de barres obliques (//). Dans FormCalc, un commentaire s'étend depuis son début jusqu'au prochain terminateur de ligne.

| Nom du caractère | Représentations |
|---|---|
| Commentaire | ;<br>// |

Par exemple :

```
// This is a type of comment
First_Name="Tony"
Initial="C" ;This is another type of comment
Last_Name="Blue"
```

### Commentaires de tous les calculs FormCalc pour un événement

L'ajout de commentaires à tous les calculs FormCalc pour un événement particulier entraîne une erreur lors de l'affichage de votre formulaire dans le panneau Aperçu PDF ou lors de l'affichage du PDF final. Chaque calcul FormCalc doit renvoyer une valeur, et FormCalc ne considère pas les commentaires comme étant des valeurs.

Pour empêcher que le code des commentaires FormCalc ne renvoie une erreur, effectuez l'une des opérations suivantes :

- Supprimez le code de commentaires de l'événement.
- Ajoutez une expression qui renvoie une valeur au code de FormCalc pour l'événement.

Pour empêcher que la valeur de l'expression n'entraîne des résultats indésirables sur votre formulaire, utilisez l'un des types d'expressions suivants :

- Une simple expression faite d'un caractère unique, comme suit :

```
// First_Name="Tony"
// Initial="C"
// Last_Name="Blue"
//
// The simple expression below sets the value of the event to zero.
0
```

- Une expression d'affectation conservant la valeur de l'objet. Utilisez ce type d'expression si le code de commentaire de FormCalc se trouve dans l'événement de calcul afin d'empêcher que la valeur réelle de l'objet ne soit modifiée, comme suit :

```
// First_Name="Tony"
// Initial="C"
// Last_Name="Blue"
//
// The assignment expression below sets the value of the current
// field equal to itself.
$.rawValue = $.rawValue
```

## Mots-clés

Les mots de passe FormCalc sont réservés et ne font pas la distinction entre majuscules et minuscules. Ces mots-clés sont réservés aux parties d'expressions, aux littéraux numériques spéciaux et aux opérateurs.

Le tableau suivant dresse la liste des mots-clés FormCalc. N'utilisez aucun de ces mots lorsque vous nommez un objet de votre conception de formulaire.

| | | | |
|---|---|---|---|
| and | endif | in | step |
| break | endwhile | infinity | then |
| continue | eq | le | this |
| do | exit | lt | throw |
| downto | for | nan | upto |
| else | foreach | ne | var |
| elseif | func | not | while |
| end | ge | null | |
| endfor | gt | or | |
| endfunc | if | return | |

## Identificateurs

Un identificateur est une séquence de caractères de longueur illimitée qui représente le nom d'une fonction ou d'une méthode. Un identificateur commence toujours par l'un des caractères suivants :

- N'importe quel caractère alphabétique (d'après le classement des lettres Unicode)
- Trait de soulignement (_)
- Signe du dollar ($)
- Point d'exclamation (!)

Les identificateurs FormCalc distinguent les lettres majuscules et les lettres minuscules. Autrement dit, les identificateurs dont les caractères ne diffèrent que par les majuscules ou les minuscules sont considérés comme distincts.

| **Nom du caractère** | **Représentations** |
|---|---|
| Identificateur | A..Z,a..z<br>$<br>!<br>_ |

Voici quelques exemples d'identificateurs valides :

```
GetAddr
$primary
_item
!dbresult
```

## Terminateurs de ligne

Les terminateurs de ligne sont utilisés pour séparer des lignes et améliorer la lisibilité.

Le tableau suivant dresse la liste des terminateurs de ligne valides FormCalc :

| Nom du caractère | Caractères Unicode |
|---|---|
| Retour chariot | #xD<br>U+000D |
| Changement de ligne | #xA<br><br>&#D; |

## Espace blanc

Les caractères d'espace blanc sont ceux qui séparent les objets et les opérations mathématiques. Ces caractères servent strictement à améliorer la lisibilité et n'ont aucune importance au moment du traitement dans FormCalc.

| Nom du caractère | Caractère Unicode |
|---|---|
| Saut de page | #xC |
| Tabulation horizontale | #x9 |
| Barre espace | #x20 |
| Tabulation verticale | #xB |

# Expressions

Les littéraux, opérateurs, commentaires, mots-clés, identificateurs, terminateurs de ligne et espaces blancs constituent ensemble une liste d'expressions, même lorsque la liste ne contient qu'une seule expression. En général, chaque expression de la liste renvoie une valeur ; la valeur de la liste dans son ensemble est celle de la dernière expression de la liste.

Considérez, par exemple, les deux champs suivants d'une conception de formulaire :

| Nom de champ | Calculs | Renvoie |
|---|---|---|
| `Field1` | `5 + Abs(Price)`<br>`« Hello World »`<br>`10 * 3 + 5 * 4` | `50` |
| `Field2` | `10 * 3 + 5 * 4` | `50` |

La valeur de `Field1` et de `Field2` après l'évaluation de l'expression de chaque champ est `50`.

FormCalc classe comme suit les différents types d'expressions qui forment une liste d'expressions :



## Simple

Dans leur forme leur plus simple, les expressions FormCalc sont des groupes d'opérateurs, mots-clés et littéraux assemblés de façon logique. Voici des exemples d'expressions simples :

```
2
"abc"
2 - 3 * 10 / 2 + 7
```

Chaque expression FormCalc renvoie une seule valeur, après exécution des opérations dans l'ordre traditionnel, même si cet ordre n'est pas toujours évident à déterminer d'après la syntaxe de l'expression. Ainsi, les ensembles d'expressions suivants, lorsqu'on les applique à des objets de conception de formulaire, produisent les mêmes résultats :

| Expression | Equivalent à | Renvoie |
|---|---|---|
| `"abc"` | `"abc"` | `abc` |
| `2 - 3 * 10 / 2 + 7` | `2 - (3 * (10 / 2)) + 7` | `-6` |
| `10 * 3 + 5 * 4` | `(10 * 3) + (5 * 4)` | `50` |
| `0 and 1 or 2 > 1` | `(0 and 1) or (2 >1)` | `1` (true) |
| `2 < 3 not 1 == 1` | `(2 < 3) not (1 == 1)` | `0` (false) |

Comme le suggère le tableau précédent, tous les opérateurs FormCalc possèdent un certain niveau de priorité lorsqu'ils apparaissent dans des expressions. Le tableau suivant illustre la hiérarchie des opérateurs :

| Priorité | Opérateur |
|---|---|
| La plus élevée | = |
| | (Unaire) - , + , not |
| | * , / |
| | + , - |
| | < , <= , > , >= , lt , le , gt , ge |
| | == , <> , eq , ne |
| | & , and |
| La plus basse | \| , or |

### Promotion d'opérandes

Dans le cas où une ou plusieurs opérandes d'une opération ne correspondent pas au type prévu pour cette opération, FormCalc promeut les opérandes pour qu'elles correspondent au type requis. La façon dont se fait cette promotion dépend du type d'opérande exigé par l'opération.

### Opérations numériques

Lorsque des opérandes non numériques se retrouvent dans des opérations numériques, elles sont d'abord promues à leur équivalent numérique. Si l'opérande non numérique n'est pas convertie correctement en valeur numérique, sa valeur devient zéro (0). La promotion d'opérandes dont la valeur est nulle en nombres donne toujours la valeur zéro.

Le tableau suivant donne quelques exemples de promotion d'opérandes non numériques.

| Expression | Equivalent à | Renvoie |
|---|---|---|
| `(5 - "abc") * 3` | `(5 - 0) * 3` | `15` |
| `"100" / 10e1` | `100 / 10e1` | `1` |
| `5 + null + 3` | `5 + 0 + 3` | `8` |

### Opération booléenne

Lorsque des opérandes non booléennes se retrouvent dans des opérations booléennes, elles sont d'abord promues à leur équivalent booléen. Si l'opérande non booléenne n'est pas convertie correctement en une valeur différente de zéro, sa valeur est Vrai (1) ; sinon, sa valeur est Faux (0). La promotion d'opérandes dont la valeur est nulle en valeur booléenne donne toujours la valeur Faux (0). Par exemple, l'expression :

```
"abc" | 2
```

renvoie `1`. Autrement dit, Faux | Vrai = Vrai, alors que

```
if ("abc") then
10
else
20
endif
```

renvoie `20`.

### Opérations de chaîne

Lorsque des opérandes non chaînes se retrouvent dans des opérations de chaîne, elles sont d'abord promues en chaînes tirées de leur valeur. La promotion d'opérandes dont la valeur est nulle en chaînes donne toujours la chaîne vide. Par exemple, l'expression :

```
concat("The total is ", 2, " dollars and ", 57, " cents.")
```

renvoie `"The total is 2 dollars and 57 cents."`

**Remarque :** si durant l'évaluation d'une expression, une étape intermédiaire génère NaN, +Inf ou -Inf, FormCalc génère une exception d'erreur et propage cette erreur dans le reste de l'expression. La valeur de l'expression est alors toujours 0. Par exemple :

```
3 / 0 + 1
```

renvoie 0.

## Affectation

Une expression d'affectation attribue la valeur d'une expression simple à la propriété identifiée par une syntaxe de référence donnée. Par exemple :

```
$template.purchase_order.name.first = "Tony"
```

Cette expression définit la valeur de la conception de formulaire « first » sur `Tony`.

Pour plus de détails sur l'utilisation de la syntaxe de référence, voir « Syntaxe de référence » à la page 24.

## OU logique

Une expression OU logique renvoie soit Vrai (1) si au moins l'une de ses opérandes est Vrai (1), soit Faux (0) si les deux opérandes sont Faux (0). Si les deux opérandes sont nulles, l'expression renvoie Nul.

| Expression | Représentation par des caractères |
|---|---|
| OU logique | \|<br>or |

Voici quelques exemples d'utilisation de l'expression OU logique :

| Expression | Renvoie |
|---|---|
| `1 or 0` | `1` (true) |
| `0 \| 0` | `0` (false) |
| `0 or 1 \| 0 or 0` | `1` (true) |

## ET logique

Une expression ET logique renvoie soit Vrai (1) si les deux opérandes sont Vrai (1), soit Faux (0) si au moins l'une de ses opérandes est Faux (0). Si les deux opérandes sont nulles, l'expression renvoie Nul.

| Expression | Représentation par des caractères |
|---|---|
| ET logique | &<br>and |

Voici quelques exemples d'utilisation de l'expression ET logique :

| Expression | Renvoie |
|---|---|
| `1 and 0` | `0` (false) |
| `0 & 0` | `1` (true) |
| `0 and 1 & 0 and 0` | `0` (false) |

## Unaire

Une expression unaire renvoie des résultats différents selon l'opérateur unaire utilisé.

| Expression | Représentation par des caractères | Renvoie |
|---|---|---|
| Unaire | - | La négation arithmétique de l'opérande, ou nul si l'opérande est nulle. |
| | + | La valeur arithmétique de l'opérande (inchangée), ou nul si l'opérande est nulle. |
| | not | La négation logique de l'opérande. |

**Remarque :** la négation arithmétique d'une opérande nulle donne le résultat nul, alors que la négation logique d'une opérande nulle renvoie le résultat booléen Vrai. Pour comprendre ces résultats, suivez le raisonnement suivant : Si « nul » signifie « rien », alors « pas rien » signifie « quelque chose ».

Voici quelques exemples d'utilisation de l'expression unaire :

| Expression | Renvoie |
|---|---|
| `-(17)` | `-17` |
| `-(-17)` | `17` |
| `+(17)` | `17` |
| `+(-17)` | `-17` |
| `not("true")` | `1` (true) |
| `not(1)` | `0` (false) |

## Egalité et inégalité

Les expressions d'égalité et d'inégalité renvoient le résultat d'une comparaison d'égalité de ses opérandes.

| Expression | Représentation par des caractères | Renvoie |
|---|---|---|
| Egalité | == eq | Vrai (1) lorsque les deux opérandes comparées sont identiques, et Faux (0) si elles sont différentes. |
| Inégalité | <> ne | Vrai (1) lorsque les deux opérandes comparées sont différentes, et Faux (0) si elles sont identiques. |

Les cas spéciaux suivants s'appliquent aussi à l'utilisation des opérateurs d'égalité :

- Si l'une des opérande est nulle, une comparaison de valeur nulle est exécutée. Si les deux opérandes sont nulles, le résultat est « identique » ; si l'une des opérandes n'est pas nulle, le résultat est « différent ».
- Si les deux opérandes sont des références, elles sont considérées identiques si elles se rapportent toutes deux au même objet, et différentes si elles ne se rapportent pas au même objet.
- Si les deux opérandes ont pour valeur une chaîne, une comparaison lexicale des chaînes, prenant en compte le paramètre régional, est exécutée sur les opérandes. Sinon, si elles ne sont pas nulles, les opérandes sont promues en valeurs numériques et une comparaison numérique a lieu.

Voici quelques exemples d'utilisation des expressions d'égalité et d'inégalité :

| Expression | Renvoie |
|---|---|
| `3 == 3` | `1` (true) |
| `3 <> 4` | `1` (true) |
| `"abc" eq "def"` | `0` (false) |
| `"def" ne "abc"` | `1` (true) |
| `5 + 5 == 10` | `1` (true) |
| `5 + 5 <> "10"` | `0` (false) |

## Expression relationnelle

Une expression relationnelle renvoie le résultat booléen de la comparaison relationnelle des opérandes.

| Expression | Représentation par des caractères | Renvoie |
|---|---|---|
| Expression relationnelle | < lt | Vrai (1) lorsque la première opérande est inférieure à la seconde ; faux (0) lorsque la première opérande est supérieure à la seconde. |
| | > gt | Vrai (1) lorsque la première opérande est supérieure à la seconde ; faux (0) lorsque la première opérande est inférieure à la seconde. |

| Expression | Représentation par des caractères | Renvoie |
|---|---|---|
| | <= le | Vrai (1) lorsque la première opérande est inférieure ou égale à la seconde ; faux (0) lorsque la première opérande est supérieure à la seconde. |
| | >= ge | Vrai (1) lorsque la première opérande est supérieure ou égale à la seconde ; faux (0) lorsque la première opérande est inférieure à la seconde. |

Les cas spéciaux suivants s'appliquent aussi à l'utilisation des opérateurs relationnels :

- Si l'une des opérandes est nulle, une comparaison de valeur nulle est exécutée. Les opérandes dont la valeur est nulle sont considérées identiques si les deux opérandes sont nulles et que l'opérateur relationnel est « inférieur ou égal à » ou « supérieur ou égal à » ; sinon, elles sont considérées différentes.
- Si les deux opérandes ont pour valeur une chaîne, une comparaison lexicale des chaînes, prenant en compte le paramètre régional, est exécutée sur les opérandes. Sinon, si elles ne sont pas nulles, les opérandes sont promues en valeurs numériques et une comparaison numérique a lieu.

Voici quelques exemples d'utilisation de l'expression relationnelle :

| Expression | Renvoie |
|---|---|
| `3 < 3` | `0` (false) |
| `3 > 4` | `0` (false) |
| `"abc" <= "def"` | `1` (true) |
| `"def" > "abc"` | `1` (true) |
| `12 >= 12` | `1` (true) |
| `"true" < "false"` | `0` (false) |

## Expressions if

Une expression (ou instruction) if évalue si une expression simple est vraie ; le résultat renvoyé est une liste d'expressions correspondant à la valeur Vrai. Si l'expression simple initiale est fausse (0), FormCalc vérifie les conditions elseif ou else et renvoie, le cas échéant, les résultats de leurs listes d'expressions.

| Expression | Syntaxe | Renvoie |
|---|---|---|
| If | `if ( simple expression ) then`<br>`  list of expressions`<br>*`elseif ( simple expression ) then`*<br>*`  list of expressions`*<br>*`else`*<br>*`  list of expressions`*<br>`endif` | Résultat de la liste d'expressions associée à toute condition correcte énoncée dans l'expression if.<br><br>**Remarque :** vous n'êtes pas tenu d'insérer des instructions elseif(...) ou else dans l'expression if. Cependant, vous devez signifier la fin de l'expression par endif. |

Voici quelques exemples d'utilisation de l'expression if :

| Expression | Renvoie |
|---|---|
| `if ( 1 < 2 ) then`<br>`  1`<br>`endif` | `1` |
| `if ( "abc" > "def") then`<br>`  1 and 0`<br>`else`<br>`  0`<br>`endif` | `0` |
| `if ( Field1 < Field2 ) then`<br>`  Field3 = 0`<br>`elseif ( Field1 > Field2 ) then`<br>`  Field3 = 40`<br>`elseif ( Field1 = Field2 ) then`<br>`  Field3 = 10`<br>`endif` | Varie avec les valeurs de `Field1` et de `Field2`. Par exemple, si Field1 est 20 et Field2 est 10, cette expression définit Field3 à 40. |

## Expressions while

Une expression while est une instruction itérative, aussi appelée instruction de boucle, qui évalue une expression simple donnée. Si le résultat de l'évaluation est Vrai (1), FormCalc examine continuellement la condition do et renvoie les résultats des listes d'expressions. Si le résultat est Faux (0), le contrôle passe alors à l'instruction suivante.

Une expression while est particulièrement adaptée aux situations où une répétition conditionnelle est nécessaire. En revanche, les situations demandant l'utilisation d'une répétition non conditionnelle sont souvent traitées plus efficacement avec une expression for.

| Expression | Syntaxe | Renvoie |
|---|---|---|
| While | `while ( simple expression ) do`<br>`  expression list`<br>`endwhile` | Le résultat de la liste d'expressions associée à la condition do. |

Dans l'exemple suivant, les valeurs des éléments sont ajoutées à une liste déroulante à partir d'un fichier XML. La méthode addItem est utilisée pour l'ensemble des éléments XML répertoriés sous list1 qui ne sont pas égaux à 3 :

```
var List = ref(xfa.record.lists.list1)
var i = 0
while ( List.nodes.item(i+1).value ne "3")do
$.addItem (List.nodes.item(i).value,List.nodes.item(i+1).value)
i = i + 2
endwhile
```

# Expressions for

Une expression for est une boucle ou une instruction itérative conditionnelle.

Une expression for est particulièrement adaptée aux situations de boucles où une répétition non conditionnelle est nécessaire. En revanche, les situations demandant l'utilisation d'une répétition conditionnelle sont souvent traitées plus efficacement avec une expression while.

La valeur de l'expression for correspond à la valeur de la dernière liste d'évaluation évaluée ou à Faux (0).

La condition for initialise une variable FormCalc qui contrôle l'action de boucle.

Dans la variante upto, la valeur de la variable de boucle effectue une itération de l'expression start à l'expression end, en utilisant pour cela des incréments de l'expression step. Si vous n'indiquez aucune valeur pour l'expression step, l'incrément step prend la valeur 1 par défaut.

Dans la variante downto, la valeur de la variable de boucle effectue une itération de l'expression start à l'expression end, en utilisant pour cela des décréments de l'expression step. Si vous n'indiquez aucun valeur pour l'expression step, les décréments step prennent la valeur -1 par défaut.

Les itérations de la boucle sont contrôlées par la valeur de l'expression end. Avant chaque itération, l'expression end est évaluée et comparée à la variable de boucle. Si le résultat est Vrai (1), la liste d'expressions est évaluée. Après chaque évaluation, l'expression step est évaluée et ajoutée à la variable de boucle.

Avant chaque itération, l'expression end est évaluée et comparée à la variable de boucle. De plus, après chaque évaluation de la condition do, l'expression step est évaluée et ajoutée à la variable de boucle.

Une boucle for se termine lorsque l'expression start a dépassé l'expression end. L'expression start peut dépasser l'expression end vers le haut, si vous utilisez la variante upto, ou vers le bas, si vous utilisez la variante downto.

| Expression | Syntaxe | Renvoie |
|---|---|---|
| For | `for variable = start expression`<br>`(upto \| downto ) end expression`<br>`(step step expression ) do`<br>`expression list`<br>`endfor`<br><br>**Remarque :** les expressions start, end et step doivent toutes être des expressions simples. | Le résultat de la liste d'expressions associée à la condition do. |

Dans l'exemple suivant, les valeurs des éléments sont ajoutées à une liste déroulante à partir d'un fichier XML. La méthode addItem est utilisée pour l'ensemble des éléments XML répertoriés sous list1 :

```
var List = ref(xfa.record.lists.list1)
for i=0 upto List.nodes.length - 1 step 2 do
$.addItem (List.nodes.item(i).value,"")
endfor
```

## Expressions foreach

Une expression foreach effectue une itération sur la liste d'expressions pour chaque valeur de sa liste d'arguments.

La valeur de l'expression foreach correspond à la valeur de la dernière liste d'expressions évaluée, ou à zéro (0) si aucune boucle n'a été indiquée.

La condition in, exécutée une seule fois (après déclaration de la variable de boucle), contrôle l'itération de la boucle. Avant chaque itération, des valeurs consécutives issues de la liste d'arguments sont affectées à la variable de boucle. La liste d'arguments ne peut pas être vide.

| Expression | Syntaxe | Renvoie |
|---|---|---|
| Foreach | `foreach variable in( argument list )do`<br>`  expression list`<br>`endfor`<br><br>**Remarque :** utilisez une virgule (,) pour séparer les expressions simples de la liste d'arguments. | La valeur de la dernière liste d'expressions évaluée ou zéro (0) si aucune boucle n'a été indiquée. |

Dans l'exemple suivant, seules les valeurs des éléments XML « display » sont ajoutées à la liste déroulante foreach.

```
foreach Item in (xfa.record.lists.list1.display[*]) do
$.addItem(Item,"")
endfor
```

## Expressions break

Une expression break provoque une sortie immédiate à partir de la boucle d'expression while, for ou foreach la plus centrale. Le contrôle est alors transmis à l'expression suivant la boucle terminée.

La valeur de l'expression break est toujours égale à zéro (0).

| Expression | Syntaxe | Renvoie |
|---|---|---|
| Saut | `break` | Transmet le contrôle à l'expression suivant la boucle terminée. |

Dans l'exemple suivant, une condition if est placée dans la boucle while pour vérifier si la valeur actuelle est égale à « Display data for 2 ». Si l'expression if est Vrai, l'expression break est exécutée et interrompt la boucle.

```
var List = ref(xfa.record.lists.list1)
var i=0
while (List.nodes.item(i+1).value ne "3") do
$.addItem(List.nodes.item(i).value,List.nodes.item(i+1).value)
i = i + 2
if (List.nodes.item(i) eq "Display data for 2" then
break
endif
endwhile
```

## Expressions continue

Une expression continue déclenche l'itération suivante de la boucle while, for ou foreach la plus centrale.

La valeur de l'expression continue est toujours égale à zéro (0).

| Expression | Syntaxe | Renvoie |
|---|---|---|
| Continue | `continue` | Lorsqu'elle est utilisée dans une expression while, le contrôle est transmis à la condition while. Lorsqu'elle est utilisée dans une expression for, le contrôle est transmis à l'expression step. |

L'objectif de l'exemple suivant est de remplir la liste déroulante à l'aide des valeurs d'un fichier XML. Si la valeur de l'élément XML actuel est « Display data for 3 », l'expression break entraîne la sortie de la boucle while. Si la valeur de l'élément XML actuel est « Display data for 2 », le script ajoute 2 à la variable `i` (le compteur) et la boucle passe immédiatement à son cycle suivant. Les deux dernières lignes ne sont pas prises en compte lorsque la valeur de l'élément XML actuel est « Display data for 2 ».

```
var List = ref(xfa.record.lists.list1)
var i = 0
while (List.nodes.item(i+1).value ne "5") do
if (List.nodes.item(i) eq "Display data for 3") then
break
endif
if (List.nodes.item(i) eq "Display data for 2" then
i=i+2
continue
endif
$.addItem(List.nodes.item(i).value,List.nodes.item(i+1).value)
i=i+2
endwhile
```

# Variables

Dans vos calculs, FormCalc permet de créer et de manipuler des variables pour stocker les données. Le nom que vous affectez à chaque variable que vous créez doit être un identificateur unique.

Par exemple, les expressions FormCalc suivantes définissent la variable userName et la valeur d'un champ de texte qui doit correspondre à la valeur de userName.

```
var userName = "Tony Blue"
TextField1.rawValue = userName
```

Vous pouvez référencer les variables que vous définissez dans le panneau Variables de la boîte de dialogue Propriétés du formulaire de la même façon. Les expressions FormCalc suivantes utilisent la fonction Concat pour définir la valeur d'un champ de texte à l'aide des variables de formulaire salutation et name.

```
TextField1.rawValue = Concat(salutation, name)
```

**Remarque :** une variable créée avec FormCalc a priorité sur une variable d'un nom semblable définie dans le panneau Variable de la boîte de dialogue Propriétés du formulaire.

# Syntaxe de référence

FormCalc donne accès aux propriétés et aux valeurs des objets de conception de formulaire au moyen d'une syntaxe de référence. L'exemple suivant illustre l'affectation et la récupération des valeurs d'un objet :

```
Invoice_Total.rawValue = Invoice_SubTotal.rawValue * (8 / 100)
```

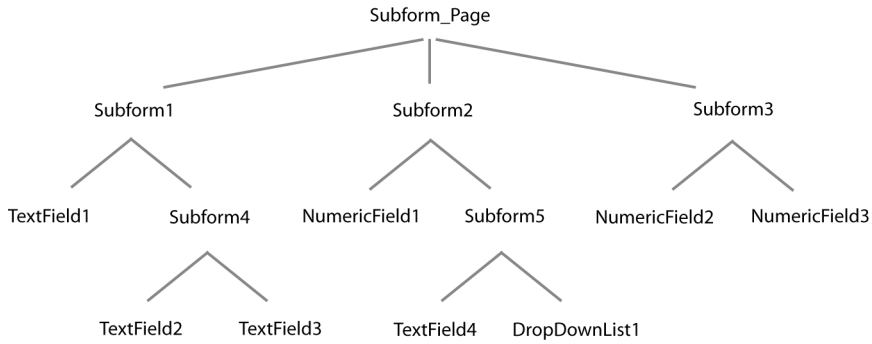
Dans ce cas, la syntaxe de référence `Invoice_Total` assigne la valeur de `Invoice_SubTotal * (8 / 100)` au champ `Invoice_Total`.

Dans le contexte des conceptions de formulaire, une syntaxe de référence complète permet l'accès à tous les objets.

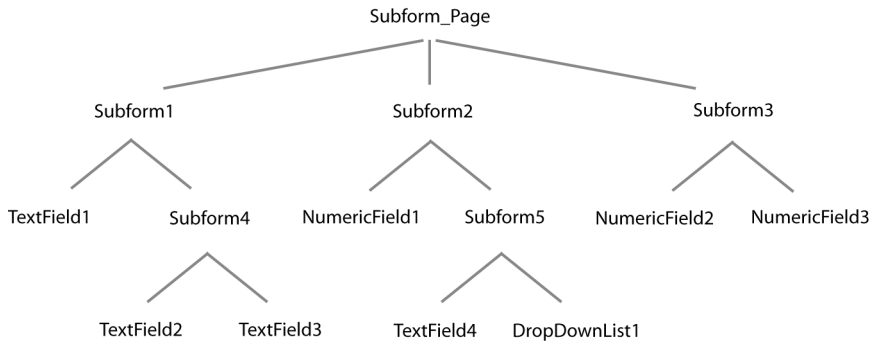
Pour faciliter l'accès aux propriétés et aux valeurs et limiter votre travail, FormCalc offre des raccourcis pratiques pour créer des références. Le tableau suivant décrit les raccourcis de la syntaxe de référence pour FormCalc.

| Notation | Description |
|---|---|
| `$` | Fait référence au champ ou à l'objet actuel, comme l'illustre cet exemple :<br>`$ = "Tony Blue"`<br>L'exemple ci-dessus définit la valeur du champ ou de l'objet actuel sur `Tony Blue`. |
| `$data` | Représente la racine du modèle de données `xfa.datasets.data`. Par exemple :<br>`$data.purchaseOrder.total`<br>est équivalent à<br>`xfa.datasets.data.purchaseOrder.total` |
| `$event` | Représente l'événement d'objet de formulaire actif. Par exemple :<br>`$event.name`<br>est équivalent à<br>`xfa.event.name` |
| `$form` | Représente la racine du modèle de formulaire `xfa.form`. Par exemple :<br>`$form.purchaseOrder.tax`<br>est équivalent à<br>`xfa.form.purchaseOrder.tax` |
| `$host` | Représente l'objet hôte. Par exemple :<br>`$host.messageBox("Hello world")`<br>est équivalent à<br>`xfa.host.messageBox("Hello world")` |
| `$layout` | Représente la racine du modèle de disposition `xfa.layout`. Par exemple :<br>`$layout.ready`<br>est équivalent à<br>`xfa.layout.ready` |

| Notation | Description |
|---|---|
| `$record` | Représente l'enregistrement actuel d'une collection de données, provenant d'un fichier XML par exemple. Par exemple :<br>`$record.header.txtOrderedByCity`<br>référence le noeud `txtOrderedByCity` dans le noeud `header` des données XML actuelles. |
| `$template` | Représente la racine de la conception de modèle `xfa.template`. Par exemple :<br>`$template.purchaseOrder.item`<br>est équivalent à<br>`xfa.template.purchaseOrder.item` |
| `!` | Représente la racine du modèle de données `xfa.datasets`. Par exemple :<br>`!data`<br>est équivalent à<br>`xfa.datasets.data` |
| `*` | Sélectionne tous les objets de formulaire dans un conteneur donné (tel qu'un sous-formulaire), quel que soit le nom, ou sélectionne tous les objets ayant un nom similaire.<br>Par exemple, l'expression suivante sélectionne tous les objets nommés `item` sur un formulaire :<br>`xfa.form.form1.item[*]` |

| Notation | Description |
|---|---|
| `..` | Vous pouvez utiliser deux points à un endroit quelconque de votre syntaxe de référence pour rechercher des objets faisant partie de n'importe quel sous-conteneur du conteneur actuel (tel qu'un sous-formulaire). Par exemple, l'expression `Subform_Page..Subform2` signifie rechercher le nœud `Subform_Page` (comme d'habitude) et trouver un descendant de `Subform_Page` appelé `Subform2`.<br>  <br><br>En se fondant sur l'arborescence ci-dessus,<br>`Subform_Page..TextField2`<br>est équivalent à<br>`Subform_Page.Subform1[0].Subform3.TextField2[0]`<br>puisque `TextField2[0]` est le premier nœud `Subform1` que FormCalc rencontre lors de sa recherche. Deuxième exemple :<br>`Subform_Page..Subform3[*]`<br>retourne les quatre instances de l'objet `TextField2`. |
| # | La notation # (signe dièse) est utilisée pour représenter l'un des éléments suivants dans une syntaxe de référence :<br>• Un objet sans nom. Ainsi, la syntaxe de référence suivante accède à un sous-formulaire sans nom :<br>`xfa.form.form1.#subform`<br>• Une propriété dans une syntaxe de référence si une propriété et un objet possèdent le même nom. La syntaxe de référence suivante accède, par exemple, à la propriété `name` d'un sous-formulaire si ce dernier contient également un champ appelé `name` :<br>`xfa.form.form1.#subform.#name` |

| Notation | Description |
|---|---|
| `[ ]` | Le crochet (`[ ]`) représente la valeur d'occurrence d'un objet. Pour créer une référence à une valeur d'occurrence, placez des crochets (`[ ]`) après un nom d'objet et insérez entre les crochets l'une des valeurs suivantes :<br>• `[ n ]`, où `n` est un numéro d'index d'occurrence absolu commençant à 0. Un numéro d'occurrence hors limite renvoie une erreur. Par exemple :<br>`xfa.form.form1.#subform.Quantity[3]`<br>`refers to the fourth occurrence of the Quantity object.`<br>• `[ +/- n ]`, où `n` indique une occurrence relative à l'occurrence de l'objet porteur de la référence. Les valeurs positives retournent des nombres d'occurrences plus élevés alors que les valeurs négatives retournent des nombres d'occurrences plus bas. Par exemple :<br>`xfa.form.form1.#subform.Quantity[+2]`<br>Cette référence retourne l'occurrence de `Quantity` dont le nombre d'occurrences est deux de plus que le nombre d'occurrences du conteneur qui constitue la référence. Si, par exemple, cette référence était jointe à l'objet `Quantity[2]`, la référence serait la même que :<br>`xfa.template.Quantity[4]`<br>Si le numéro d'index calculé est hors limite, la référence renvoie une erreur.<br>Cette syntaxe sert surtout à localiser l'occurrence précédente ou suivante d'un objet en particulier. Par exemple, chaque occurrence de l'objet `Quantity` (sauf la première) peut utiliser `Quantity[-1]` pour obtenir la valeur de l'objet `Quantity` précédent.<br>• `[*]` indique plusieurs occurrences d'un objet. Le premier objet nommé est trouvé, et les objets portant le même nom et ayant le même parent immédiat sont retournés. Le résultat de cette notation est une collection d'objets. Par exemple :<br>`xfa.form.form1.#subform.Quantity[*]`<br>Cette expression fait référence à tous les objets portant le nom `Quantity` et ayant le même parent immédiat que la première occurrence `Quantity` trouvée par la référence.<br>**Remarque :** dans les formulaires en arabe, hébreu, thaï et vietnamien, la syntaxe de référence se trouve toujours à droite (même pour les langues s'écrivant de droite à gauche).<br>**Remarque :** LiveCycle Forms 7.1 ne prend pas en charge les conceptions de formulaire contenant des polices relatives à l'arabe, l'hébreu, le thaï et le vietnamien. |

| Notation | Description |
|---|---|
| `[ ]`<br>`(suite)` |  <br>L'arborescence servant de référence, les expressions suivantes retournent les résultats indiqués :<br>• `Subform_Page.Subform1[*]` renvoie les deux objets `Subform1`.<br>• `Subform_Page.Subform1.Subform3.TextField2[*]` renvoie les deux objets `TextField2`. `Subform_Page.Subform1` a pour résultat le premier objet `Subform1` à gauche et `TextField2[*]` est évalué relativement à l'objet `Subform3`.<br>• `Subform_Page.Subform1[*].TextField1` renvoie les deux instances `TextField1`. `Subform_Page.Subform1[*]` a pour résultat les deux objets `Subform1` et `TextField1` est évalué relativement aux objets `Subform1`.<br>• `Subform_Page.Subform1[*].Subform3.TextField2[1]` renvoie le deuxième et le quatrième objet `TextField2` à partir de la gauche. `Subform_Page.Subform1[*]` a pour résultat les deux objets `Subform1` et `TextField2[1]` est évalué relativement aux objets `Subform3`.<br>• `Subform_Page.Subform1[*].Subform3[*]` retourne les deux instances de l'objet `Subform3`.<br>• `Subform_Page.*` retourne les deux objets `Subform1` et l'objet `Subform2`.<br>• `Subform_Page.Subform2.*` retourne les deux instances de l'objet `NumericField2`. |

# Appel des propriétés et des méthodes

LiveCycle Designer définit différentes méthodes et propriétés pour tous les objets de conception de formulaire. FormCalc donne accès à ces propriétés et méthodes et permet de les utiliser pour changer l'aspect et le comportement des objets sur votre formulaire. Tout comme l'appel de fonction, l'appel de méthode et de propriété se fait par la transmission d'arguments selon un ordre spécifique. Le nombre et le type d'arguments de chaque méthode et propriété sont propres à chaque type d'objet.

**Remarque :** en effet, différents objets de conception de formulaire correspondent à différentes méthodes et propriétés. Pour obtenir une liste exhaustive des méthodes et des propriétés gérées par les objets, reportez-vous au document *Adobe XML Form Object Model Reference* à partir du site LiveCycle Designer Developer Center à l'adresse suivante : www.adobe.com/devnet/livecycle/designing_forms.html.

# Appels de fonctions intégrées

FormCalc prend en charge un vaste ensemble de fonctions intégrées avec une multitude de possibilités. Le nom de ces fonctions ne fait pas la distinction entre les majuscules et les minuscules, mais contrairement aux mots-clés, FormCalc ne réserve pas le nom des fonctions. Autrement dit, dans les formulaires, les calculs d'objets dont les noms coïncident avec ceux des fonctions FormCalc ne génèrent pas de conflits.

Selon la fonction utilisée, vous pouvez être amené à indiquer un ensemble d'arguments pour qu'une valeur soit renvoyée. Beaucoup de fonctions comportent des arguments facultatifs : selon la situation, vous devez décider des arguments à utiliser.

FormCalc évalue tous les arguments de fonction dans l'ordre, de gauche à droite. Si vous tentez de transmettre à une fonction un nombre d'arguments inférieur à celui requis, la fonction génère une exception d'erreur.

Chaque fonction exige que ses arguments soient entrés selon un format précis, sous la forme d'un littéral numérique ou d'un littéral chaîne. Si la valeur d'un argument ne correspond pas à ce qu'attend la fonction, FormCalc convertit la valeur. Par exemple :

```
Len(35)
```

La fonction Len exige un littéral chaîne. Dans ce cas, FormCalc convertit l'argument, du nombre 35 à la chaîne « 35 », et la fonction retourne 2.

Cependant, la conversion d'un littéral chaîne en littéral numérique est plus complexe. Par exemple :

```
Abs("abc")
```

La fonction Abs exige un littéral numérique. FormCalc convertit la valeur de tous les littéraux chaînes à la valeur 0. Cette conversion peut générer des problèmes dans les fonctions où la valeur 0 entraîne une erreur, comme dans le cas de la fonction Apr.

Les arguments de certaines fonctions doivent obligatoirement être des entiers. Dans ce cas, les arguments transmis sont toujours convertis en entiers, la partie fractionnaire étant tronquée.

Voici le résumé des principales propriétés des fonctions intégrées :

- Le nom des fonctions intégrées ne fait pas la distinction entre les minuscules et les majuscules.
- Même si les fonctions intégrées sont prédéfinies, leurs noms ne constituent pas des mots réservés. Autrement dit, la fonction intégrée Max, par exemple, n'entre jamais en conflit avec un objet, une propriété d'objet ou une méthode d'objet nommée.
- Beaucoup de fonctions intégrées exigent la saisie d'un certain nombre d'arguments, parfois suivis d'arguments facultatifs.
- Quelques fonctions intégrées, telles que Avg, Count, Max, Min, Sum et Concat, acceptent un nombre indéfini d'arguments.

Pour consulter la liste complète des fonctions FormCalc, voir « Liste alphabétique des fonctions » à la page 30.

# 3 Liste alphabétique des fonctions

Le tableau suivant contient toutes les fonctions FormCalc, la description de chaque fonction et la catégorie à laquelle elle appartient.

| Fonction | Description | Type |
|---|---|---|
| « Abs » à la page 34 | Retourne la valeur absolue d'une valeur ou d'une expression numérique. | Arithmétique |
| « Apr » à la page 64 | Renvoie le taux annuel d'un prêt. | Financière |
| « At » à la page 83 | Repère la position du premier caractère d'une chaîne dans une autre chaîne. | Chaîne |
| « Avg » à la page 35 | Evalue un ensemble de valeurs et/ou d'expressions numériques et renvoie la moyenne des éléments non nuls de cet ensemble. | Arithmétique |
| « Ceil » à la page 36 | Renvoie le nombre entier supérieur ou égal au nombre fourni. | Arithmétique |
| « Choose » à la page 74 | Sélectionne une valeur dans un ensemble de paramètres donné. | Logique |
| « Concat » à la page 83 | Renvoie la concaténation des chaînes indiquées. | Chaîne |
| « Count » à la page 36 | Evalue un ensemble de valeurs et/ou d'expressions et renvoie le nombre d'éléments non nuls contenus dans cet ensemble. | Arithmétique |
| « CTerm » à la page 65 | Renvoie le nombre de périodes nécessaires pour qu'un investissement, dont le taux d'intérêt est fixe mais composé, atteigne une valeur capitalisée. | Financière |
| « Date » à la page 53 | Renvoie la date active du système sous la forme du nombre de jours écoulés depuis l'époque. | Date et heure |
| « Date2Num » à la page 53 | Renvoie le nombre de jours écoulés depuis l'époque, à partir d'une chaîne de date. | Date et heure |
| « DateFmt » à la page 54 | Renvoie une chaîne de format de date, à partir d'un style de format de date. | Date et heure |
| « Decode » à la page 84 | Renvoie la version décodée d'une chaîne donnée. | Chaîne |
| « Encode » à la page 85 | Renvoie la version codée d'une chaîne donnée. | Chaîne |
| « Eval » à la page 78 | Renvoie la valeur d'un calcul de formulaire donné. | Divers |
| « Exists » à la page 75 | Détermine si le paramètre donné est une syntaxe de référence à un objet existant. | Logique |
| « Floor » à la page 37 | Renvoie le plus grand nombre entier inférieur ou égal à la valeur donnée. | Arithmétique |

| Fonction | Description | Type |
|---|---|---|
| « Format » à la page 86 | Formate les données fournies selon la chaîne de format d'image indiquée. | Chaîne |
| « FV » à la page 66 | Renvoie la valeur capitalisée de paiements fixes effectués à intervalles réguliers, compte tenu d'un taux d'intérêt fixe. | Financière |
| « Get » à la page 99 | Télécharge le contenu de l'adresse URL indiquée. | URL |
| « HasValue » à la page 76 | Détermine si le paramètre indiqué est un mécanisme d'accès dont la valeur n'est ni nulle, ni vide, ni un blanc. | Logique |
| « IPmt » à la page 67 | Renvoie le montant de l'intérêt payé pour un prêt au cours d'une période donnée. | Financière |
| « IsoDate2Num » à la page 55 | Renvoie le nombre de jours écoulés depuis l'époque, à partir d'une chaîne de date valide. | Date et heure |
| « IsoTime2Num » à la page 55 | Renvoie le nombre de millisecondes écoulées depuis l'époque, à partir d'une chaîne d'heure valide. | Date et heure |
| « Left » à la page 87 | Extrait un nombre spécifique de caractères d'une chaîne, en commençant par le premier caractère à gauche. | Chaîne |
| « Len » à la page 87 | Renvoie le nombre de caractères d'une chaîne donnée. | Chaîne |
| « LocalDateFmt » à la page 56 | Renvoie une chaîne de format de date localisée, à partir d'un style de format de date. | Date et heure |
| « LocalTimeFmt » à la page 57 | Renvoie une chaîne de format d'heure localisée, à partir d'un style de format d'heure. | Date et heure |
| « Lower » à la page 88 | Convertit en minuscules toutes les majuscules d'une chaîne donnée. | Chaîne |
| « Ltrim » à la page 89 | Renvoie une chaîne sans aucun caractère d'espace blanc à gauche. | Chaîne |
| « Max » à la page 38 | Renvoie la valeur maximale des éléments non nuls d'un ensemble donné de nombres. | Arithmétique |
| « Min » à la page 39 | Renvoie la valeur minimale des éléments non nuls d'un ensemble donné de nombres. | Arithmétique |
| « Mod » à la page 40 | Renvoie le reste d'un nombre divisé par un autre. | Arithmétique |
| « NPV » à la page 68 | Renvoie la valeur nette actualisée d'un investissement, compte tenu d'une série de mouvements futurs de l'encaisse et d'un taux d'actualisation. | Financière |
| « Null » à la page 79 | Renvoie la valeur nulle. La valeur nulle signifie aucune valeur. | Divers |
| « Num2Date » à la page 58 | Renvoie une chaîne de date, à partir du nombre de jours écoulés depuis l'époque. | Date et heure |

| Fonction | Description | Type |
|---|---|---|
| « Num2GMTime » à la page 59 | Renvoie une chaîne d'heure UT, à partir du nombre de millisecondes écoulées depuis l'époque. | Date et heure |
| « Num2Time » à la page 60 | Renvoie une chaîne d'heure, à partir du nombre de millisecondes écoulées depuis l'époque. | Date et heure |
| « Oneof » à la page 76 | Retourne vrai (1) si une valeur se trouve dans un ensemble donné, et faux (0) dans le cas contraire. | Logique |
| « Parse » à la page 89 | Analyse les données fournies selon le format d'image donné. | Chaîne |
| « Pmt » à la page 69 | Renvoie le remboursement d'un prêt basé sur des versements fixes et sur un taux d'intérêt fixe. | Financière |
| « Post » à la page 100 | Place à l'adresse URL indiquée les données fournies. | URL |
| « PPmt » à la page 70 | Renvoie le montant du principal payé pour un prêt au cours d'une période. | Financière |
| « Put » à la page 101 | Télécharge les données fournies vers l'adresse URL indiquée. | URL |
| « PV » à la page 71 | Renvoie la valeur actualisée d'un investissement réalisé par versements fixes périodiques avec un taux d'intérêt fixe. | Financière |
| « Rate » à la page 72 | Renvoie le taux d'intérêt composé par période nécessaire pour qu'un investissement d'une valeur actualisée atteigne une valeur capitalisée au cours d'une période donnée. | Financière |
| « Ref » à la page 79 | Renvoie une référence à un objet existant. | Divers |
| « Replace » à la page 90 | Remplace toutes les occurrences d'une chaîne par une autre, dans la chaîne indiquée. | Chaîne |
| « Right » à la page 91 | Extrait un nombre spécifique de caractères dans une chaîne donnée, en commençant par le dernier caractère à droite. | Chaîne |
| « Round » à la page 41 | Evalue une valeur ou expression numérique donnée et renvoie un nombre arrondi comportant le nombre de décimales défini. | Arithmétique |
| « Rtrim » à la page 91 | Renvoie une chaîne sans aucun caractère d'espace blanc à droite. | Chaîne |
| « Space » à la page 92 | Renvoie une chaîne composée d'un nombre donné d'espaces blancs. | Chaîne |
| « Str » à la page 93 | Convertit un nombre en une chaîne de caractères. FormCalc formate le résultat selon la largeur indiquée et l'arrondit au nombre de décimales défini. | Chaîne |
| « Stuff » à la page 94 | Insère une chaîne dans une autre chaîne. | Chaîne |

| Fonction | Description | Type |
|---|---|---|
| « Substr » à la page 95 | Extrait une partie d'une chaîne donnée. | Chaîne |
| « Sum » à la page 42 | Renvoie la somme des éléments non nuls d'un ensemble donné de nombres. | Arithmétique |
| « Term » à la page 73 | Renvoie le nombre de périodes nécessaires pour atteindre la valeur capitalisée donnée, compte tenu de paiements fixes périodiques versés sur un compte portant intérêt. | Financière |
| « Time » à la page 61 | Renvoie l'heure actuelle du système sous la forme du nombre de millisecondes écoulées depuis l'époque. | Date et heure |
| « Time2Num » à la page 61 | Renvoie le nombre de millisecondes écoulées depuis l'époque, à partir d'une chaîne d'heure. | Date et heure |
| « TimeFmt » à la page 62 | Renvoie un format d'heure, à partir d'un style de format d'heure. | Date et heure |
| « UnitType » à la page 80 | Renvoie les unités d'une étendue d'unité. Une étendue d'unité est une chaîne composée d'un nombre suivi d'un nom d'unité. | Divers |
| « UnitValue » à la page 81 | Retourne la valeur numérique d'une mesure avec l'étendue d'unité qui lui est associée, après une conversion d'unité facultative. | Divers |
| « Upper » à la page 96 | Convertit en majuscules toutes les minuscules d'une chaîne. | Chaîne |
| « Uuid » à la page 96 | Renvoie une chaîne UUID qui sert de méthode d'identification. | Chaîne |
| « Within » à la page 77 | Retourne vrai (1) si une valeur de contrôle se trouve dans une plage donnée, et faux (0) dans le cas contraire. | Logique |
| « WordNum » à la page 97 | Renvoie le texte équivalent à un nombre donné. | Chaîne |

# 4 Fonctions arithmétiques

Ces fonctions exécutent différentes opérations mathématiques.

**Fonctions**



## Abs

Renvoie la valeur absolue d'une valeur ou expression numérique, ou renvoie « nul » si la valeur ou expression est nulle.

**Syntaxe**

`Abs(`*`n1`*`)`

**Paramètres**

| Paramètre | Description |
|---|---|
| `n1` | Valeur ou expression numérique à évaluer. |

**Remarque :** FormCalc respecte la norme internationale IEEE-754 en matière de gestion des valeurs numériques à virgule flottante. Pour plus de détails, voir « Littéraux numériques » à la page 8.

**Exemples**

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Abs` :

| Expression | Renvoie |
|---|---|
| `Abs(1.03)` | `1.03` |
| `Abs(-1.03)` | `1.03` |
| `Abs(0)` | `0` |

# Avg

Evalue un ensemble de valeurs et/ou d'expressions numériques et renvoie la moyenne des éléments non nuls de cet ensemble.

### Syntaxe

`Avg(`*`n1`* `[, `*`n2`* `...])`

### Paramètres

| Paramètre | Description |
|---|---|
| `n1` | Première valeur ou expression numérique de l'ensemble. |
| `n2` (facultatif) | Autres valeurs ou expressions numériques. |

**Remarque :** FormCalc respecte la norme internationale IEEE-754 en matière de gestion des valeurs numériques à virgule flottante. Pour plus de détails, voir « Littéraux numériques » à la page 8.

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Avg` :

| Expression | Renvoie |
|---|---|
| `Avg(0, 32, 16)` | `16` |
| `Avg(2.5, 17, null)` | `9.75` |
| `Avg(Price[0], Price[1], Price[2], Price[3])` | Moyenne des quatre premières occurrences non nulles de `Price`. |
| `Avg(Quantity[*])` | Moyenne de toutes les occurrences non nulles de `Quantity`. |

# Ceil

Renvoie le nombre entier supérieur ou égal au nombre fourni, ou renvoie « nul » si le paramètre est nul.

### Syntaxe

`Ceil(n)`

### Paramètres

| Paramètre | Description |
|---|---|
| `n` | N'importe quelle valeur ou expression numérique.<br>La fonction renvoie la valeur `0` si `n` n'est pas une valeur ou expression numérique. |

**Remarque :** FormCalc respecte la norme internationale IEEE-754 en matière de gestion des valeurs numériques à virgule flottante. Pour plus de détails, voir « Littéraux numériques » à la page 8.

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Ceil` :

| Expression | Renvoie |
|---|---|
| `Ceil(2.5875)` | `3` |
| `Ceil(-5.9)` | `-5` |
| `Ceil("abc")` | `0` |
| `Ceil(A)` | 100 si la valeur de `A` est égale à 99,999. |

# Count

Evalue un ensemble de valeurs et/ou expressions, et renvoie le nombre d'éléments non nuls contenus dans cet ensemble.

### Syntaxe

`Count(n1 [, n2 ...])`

### Paramètres

| Paramètre | Description |
|---|---|
| `n1` | Valeur ou expression numérique. |
| `n2` (facultatif) | Autres valeurs et/ou expressions numériques. |

**Remarque :** FormCalc respecte la norme internationale IEEE-754 en matière de gestion des valeurs numériques à virgule flottante. Pour plus de détails, voir « Littéraux numériques » à la page 8.

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Count` :

| Expression | Renvoie |
|---|---|
| `Count("Tony", "Blue", 41)` | `3` |
| `Count(Customers[*])` | Nombre d'occurrences non nulles de `Customers`. |
| `Count(Coverage[2], "Home", "Auto")` | `3`, à condition que la troisième occurrence de `Coverage` ne soit pas nulle. |

# Floor

Renvoie le plus grand nombre entier inférieur ou égal à la valeur donnée.

### Syntaxe
`Floor(`*n*`)`

### Paramètres

| Paramètre | Description |
|---|---|
| `n` | N'importe quelle valeur ou expression numérique. |

**Remarque :** FormCalc respecte la norme internationale IEEE-754 en matière de gestion des valeurs numériques à virgule flottante. Pour plus de détails, voir « Littéraux numériques » à la page 8.

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Floor` :

| Expression | Renvoie |
|---|---|
| `Floor(21.3409873)` | `21` |
| `Floor(5.999965342)` | `5` |
| `Floor(3.2 * 15)` | `48` |

# Max

Renvoie la valeur maximale des éléments non nuls d'un ensemble donné de nombres.

### Syntaxe

`Max(n1 [, n2 ...])`

### Paramètres

| Paramètre | Description |
|---|---|
| `n1` | Valeur ou expression numérique. |
| `n2` (facultatif) | Autres valeurs et/ou expressions numériques. |

**Remarque :** FormCalc respecte la norme internationale IEEE-754 en matière de gestion des valeurs numériques à virgule flottante. Pour plus de détails, voir « Littéraux numériques » à la page 8.

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Max` :

| Expression | Renvoie |
|---|---|
| `Max(234, 15, 107)` | `234` |
| `Max("abc", 15, "Tony Blue")` | `15` |
| `Max("abc")` | `0` |
| `Max(Field1[*], Field2[0])` | Évalue les occurrences non nulles de `Field1`, ainsi que la première occurrence de `Field2`, et retourne la valeur la plus élevée. |
| `Max(Min(Field1[*], Field2[0]), Field3, Field4)` | La première expression évalue les occurrences non nulles de `Field1`, ainsi que la première occurrence de `Field2`, et renvoie la valeur la plus basse. Le résultat final est le maximum de cette valeur comparativement aux valeurs de `Field3` et de `Field4`.<br>Voir aussi « Min » à la page 39. |

# Min

Renvoie la valeur minimale des éléments non nuls d'un ensemble donné de nombres.

### Syntaxe

`Min(`*`n1`* `[, `*`n2`* `...])`

### Paramètres

| Paramètre | Description |
|---|---|
| `n1` | Valeur ou expression numérique. |
| `n2` (facultatif) | Autres valeurs et/ou expressions numériques. |

**Remarque :** FormCalc respecte la norme internationale IEEE-754 en matière de gestion des valeurs numériques à virgule flottante. Pour plus de détails, voir « Littéraux numériques » à la page 8.

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Min` :

| Expression | Renvoie |
|---|---|
| `Min(234, 15, 107)` | `15` |
| `Min("abc", 15, "Tony Blue")` | `15` |
| `Min("abc")` | `0` |
| `Min(Field1[*], Field2[0])` | Évalue les occurrences non nulles de `Sales_July`, ainsi que la première occurrence de `Sales_August`, et retourne la valeur la plus basse. |
| `Min(Max(Field1[*], Field2[0]), Field3, Field4)` | La première expression évalue les occurrences non nulles de `Field1`, ainsi que la première occurrence de `Field2`, et renvoie la valeur la plus élevée. Le résultat final est le minimum de cette valeur comparativement aux valeurs de `Field3` et de `Field4`.<br>Voir aussi « Max » à la page 38. |

# Mod

Renvoie le reste d'un nombre divisé par un autre. Il s'agit du reste de la division du dividende par le diviseur. Le signe du reste est toujours le même que celui du dividende.

### Syntaxe

`Mod(`*`n1, n2`*`)`

### Paramètres

| Paramètre | Description |
|---|---|
| `n1` | Le dividende : valeur ou expression numérique. |
| `n2` | Le diviseur : valeur ou expression numérique. |

Si `n1` et/ou `n2` ne sont pas des valeurs ou des expressions numériques, la fonction renvoie 0.

**Remarque :** FormCalc respecte la norme internationale IEEE-754 en matière de gestion des valeurs numériques à virgule flottante. Pour plus de détails, voir « Littéraux numériques » à la page 8.

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Mod` :

| Expression | Renvoie |
|---|---|
| `Mod(64, -3)` | `1` |
| `Mod(-13,3)` | `-1` |
| `Mod("abc", 2)` | `0` |
| `Mod(X[0], Y[9])` | La première occurrence de `X` sert de dividende et la dixième occurrence de `Y` est utilisée comme diviseur. |
| `Mod(Round(Value[4], 2), Max(Value[*]))` | La cinquième occurrence de `Value` arrondie à deux décimales sert de dividende et la valeur la plus élevée parmi toutes les occurrences non nulles de `Value` est utilisée comme diviseur.<br>Voir aussi « Max » à la page 38 et « Round » à la page 41. |

# Round

Evalue une valeur ou expression numérique donnée et renvoie un nombre arrondi comportant un nombre de décimales défini.

### Syntaxe

`Round(n1 [, n2])`

### Paramètres

| Paramètre | Description |
|---|---|
| `n1` | Valeur ou expression numérique à évaluer. |
| `n2` (facultatif) | Nombre de décimales avec lesquelles `n1` est évalué (maximum de 12).<br>Si vous omettez la valeur `n2`, ou si `n2` n'est pas valide, la fonction ne retournera aucune décimale. |

**Remarque :** FormCalc respecte la norme internationale IEEE-754 en matière de gestion des valeurs numériques à virgule flottante. Pour plus de détails, voir « Littéraux numériques » à la page 8.

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Round` :

| Expression | Renvoie |
|---|---|
| `Round(12.389764537, 4)` | `12.3898` |
| `Round(20/3, 2)` | `6.67` |
| `Round(8.9897, "abc")` | `9` |
| `Round(FV(400, 0.10/12, 30*12), 2)` | `904195.17`. Cette fonction prend la valeur obtenue par la fonction `FV` et l'arrondit à deux décimales.<br>Voir aussi « FV » à la page 66. |
| `Round(Total_Price, 2)` | Arrondit la valeur de `Total_Price` à deux décimales. |

# Sum

Renvoie la somme des éléments non nuls d'un ensemble donné de nombres.

### Syntaxe

`Sum(`*`n1`* `[, `*`n2`* `...])`

### Paramètres

| Paramètre | Description |
|---|---|
| `n1` | Valeur ou expression numérique. |
| `n2` (facultatif) | Autres valeurs et/ou expressions numériques. |

**Remarque :** FormCalc respecte la norme internationale IEEE-754 en matière de gestion des valeurs numériques à virgule flottante. Pour plus de détails, voir « Littéraux numériques » à la page 8.

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Sum` :

| Expression | Renvoie |
|---|---|
| `Sum(2, 4, 6, 8)` | `20` |
| `Sum(-2, 4, -6, 8)` | `4` |
| `Sum(4, 16, "abc", 19)` | `39` |
| `Sum(Amount[2], Amount[5])` | Totaux des troisième et sixième occurrences de `Amount`. |
| `Sum(Round(20/3, 2), Max(Amount[*]), Min(Amount[*]))` | Effectue le total de la valeur de 20/3 arrondie à deux décimales, et des occurrences non nulles les plus et moins élevées de `Amount`.<br>Voir aussi « Max » à la page 38, « Min » à la page 39 et « Round » à la page 41. |

# 5 Fonctions de date et d'heure

Les fonctions décrites dans la présente section portent spécifiquement sur la création et la gestion des valeurs de date et d'heure.

**Fonctions**



## Structuration des dates et des heures

### Paramètres régionaux

Un *paramètre régional* est un terme standard utilisé, dans le développement de normes internationales, pour identifier une nation (une langue, un pays ou les deux). En ce qui concerne FormCalc, un *paramètre régional* définit le format de date et d'heure ainsi que le format numérique et monétaire d'une nation ou d'une région, pour permettre aux utilisateurs de se servir des formats auxquels ils sont habitués.

Chaque paramètre régional comprend une chaîne de caractères unique appelée *identificateur de paramètre régional*. La composition de ces chaînes est gérée par le groupe IETF (Internet Engineering Task Force) de l'Organisation mondiale de normalisation (ISO), organe de la Société Internet (www.isoc.org).

Les identificateurs de paramètre régional se composent d'une partie langue, d'une partie pays ou des deux. Le tableau suivant présente la liste des paramètres régionaux valides avec la présente version de LiveCycle Designer.

| **Langage** | **Pays** | **Code ISO** |
|---|---|---|
| Allemand | Autriche | de_AT |
| Allemand | Allemagne | de_DE |
| Allemand | Luxembourg | de_LU |
| Allemand | Suisse | de_CH |
| Anglais | Australie | en_AU |
| Anglais | Canada | en_CA |
| Anglais | Inde | en_IN |
| Anglais | Irlande | en_IE |
| Anglais | Nouvelle-Zélande | en_NZ |
| Anglais | Afrique du Sud | en_ZA |
| Anglais | Royaume-Uni | en_GB |
| Anglais | Royaume-Uni (Europe) | en_GB_EURO |
| Anglais | Etats-Unis | en_US |
| Arabe | Emirats arabes unis | ar_AE |
| Arabe | Bahreïn | ar_BH |
| Arabe | Algérie | ar_DZ |
| Arabe | Egypte | ar_EG |
| Arabe | Iraq | ar_IQ |
| Arabe | Jordanie | ar_JO |
| Arabe | Koweït | ar_KW |
| Arabe | Liban | ar_LB |
| Arabe | Libye | ar_LY |
| Arabe | Maroc | ar_MA |
| Arabe | Oman | ar_OM |
| Arabe | Qatar | ar_QA |
| Arabe | Arabie Saoudite | ar_SA |
| Arabe | Soudan | ar_SD |
| Arabe | Syrie | ar_SY |
| Arabe | Tunisie | ar_TN |

| Langage | Pays | Code ISO |
|---|---|---|
| Arabe | Yémen | ar_YE |
| Bulgare | Bulgarie | bg_BG |
| Chinois | Hong Kong | zh_HK |
| Chinois | Chinois simplifié (République populaire de Chine) | zh_CN |
| Chinois | Chinois traditionnel | zh_TW |
| Coréen | République de Corée | ko_KR |
| Coréen | Corée (caractères Hanja) | ko_KR_HANI |
| Croate | Croatie | hr_HR |
| Danois | Danemark | da_DK |
| Espagnol | Equateur | es_EC |
| Espagnol | El Salvador | es_SV |
| Espagnol | Guatemala | es_GT |
| Espagnol | Honduras | es_HN |
| Espagnol | Nicaragua | es_NI |
| Espagnol | Panama | es_PA |
| Espagnol | Paraguay | es_PY |
| Espagnol | Porto Rico | es_PR |
| Espagnol | Uruguay | es_UY |
| Espagnol | Argentine | es_AR |
| Espagnol | Bolivie | es_BO |
| Espagnol | Chili | es_CL |
| Espagnol | Colombie | es_CO |
| Espagnol | Costa Rica | es_CR |
| Espagnol | République dominicaine | es_DO |
| Espagnol | Mexique | es_MX |
| Espagnol | Pérou | es_PE |
| Espagnol | Espagne | es_ES |
| Espagnol | Venezuela | es_VE |
| Estonien | Estonie | et_EE |
| Finnois | Finlande | fi_FI |
| Français | Belgique | fr_BE |

| Langage | Pays | Code ISO |
|---|---|---|
| Français | Canada | fr_CA |
| Français | France | fr_FR |
| Français | Luxembourg | fr_LU |
| Français | Suisse | fr_CH |
| Grec | Grèce | el_GR |
| Hébreu | Israël | he_IL |
| Hongrois | Hongrie | hu_HU |
| Indonésien | Indonésie | id_ID |
| Italien | Italie | it_IT |
| Italien | Suisse | it_CH |
| Japonais | Japon | ja_JP |
| Letton | Lettonie | lv_LV |
| Lituanien | Lituanie | lt_LT |
| Malais | Malaisie | ms_MY |
| Néerlandais | Belgique | nl_BE |
| Néerlandais | Pays-Bas | nl_NL |
| Norvégien (Bokmal) | Norvège | nb_NO |
| Norvégien (Nynorsk) | Norvège | nn_NO |
| Polonais | Pologne | pl_PL |
| Portugais | Brésil | pt_BR |
| Portugais | Portugal | pt_PT |
| Roumain | Roumanie | ro_RO |
| Russe | Russie | ru_RU |
| Serbo-croate | Bosnie-Herzégovine | sh_BA |
| Serbo-croate | Croatie | sh_HR |
| Serbo-croate | Serbie-et-Monténégro | sh_CS |
| Slovaque | Slovaquie | sk_SK |
| Slovène | Slovénie | sl_SI |
| Suédois | Suède | sv_SE |
| Tchèque | République tchèque | cs_CAZ |
| Thaï | Thaïlande | th_TH |

| Langage | Pays | Code ISO |
|---|---|---|
| Thaï | Royaume de Thaïlande | th_TH_TH |
| Turc | Turquie | tr_TR |
| Ukrainien | Ukraine | uk_UA |
| Vietnamien | Vietnam | vi_VN |

Habituellement, les deux éléments d'un paramètre régional sont importants. Par exemple, le nom des jours de la semaine et des mois en anglais pour le Canada et en anglais pour la Grande-Bretagne ont le même format, mais les dates sont formatées différemment. Ainsi, l'indication de la langue anglaise comme paramètre régional n'est pas suffisante. De même, l'indication du seul pays comme paramètre régional n'est pas suffisante. Par exemple, le Canada possède deux formats de date, l'un pour l'anglais et l'autre pour le français.

En général, chaque application fonctionne dans un environnement où se trouve un paramètre régional. Ce paramètre régional est *ambiant*. Dans certaines circonstances, une application peut fonctionner sur un système, ou dans un environnement, dans lequel ne se trouve aucun paramètre régional. Dans ces rares cas, le paramètre régional ambiant par défaut est l'anglais des Etats-Unis (en-US). Ce paramètre régional est celui *par défaut*.

## Epoque

On associe aux valeurs de date et d'heure une origine, dite *époque*, qui sert de point de départ chronologique. Toute valeur de date ou d'heure située avant cette époque n'est pas valide.

L'unité de valeur pour toutes les fonctions de date est le nombre de jours calculé depuis l'époque. L'unité de valeur pour toutes les fonctions d'heure est le nombre de millisecondes calculé depuis l'époque.

LiveCycle Designer définit le jour 1 de l'époque pour toutes les fonctions de date comme étant le 1er janvier 1900, et la milliseconde 1 de l'époque pour toutes les fonctions d'heure comme étant minuit, 00:00:00, en temps universel coordonné (UTC), aussi connu sous le nom de « temps moyen de Greenwich » (GMT). Cette définition signifie que les utilisateurs des fuseaux horaires situés à l'Est de l'UTC peuvent recevoir des valeurs d'heure négatives.

## Formats de date

Un *format de date* est une notation abrégée déterminant la façon dont la date s'affiche. Elle comprend différents signes de ponctuation et symboles représentant le formatage utilisé par la date. Le tableau suivant fournit des exemples de formats de date :

| Format de date | Exemple |
|---|---|
| MM/DD/YY | 11/11/78 |
| DD/MM/YY | 25/07/85 |
| MMMM DD, YYYY | Mars 10, 1964 |

Le format de date est géré par une norme ISO. Chaque pays ou région spécifie ses propres formats de date. Les quatre catégories générales de formats de date correspondent aux formats court, moyen, long et

complet. Le tableau ci-dessous indique quelques exemples de formats de date, dans différents paramètres régionaux, pour chacune des quatre catégories.

| Identificateur de paramètre régional et description | Format de date (catégorie) | Exemple |
|---|---|---|
| en_GB<br>Anglais (Royaume-Uni) | DD/MM/YY (Court) | 08/18/92<br>08/04/05 |
| fr_CA<br>Français (Canada) | YY-MM-DD (Moyen) | 92-08-18 |
| de_DE<br>Allemand (Allemagne) | D. MMMM YYYY (Long) | 17. Juni 1989 |
| fr_FR<br>Français (France) | EEEE, ' le ' D MMMM YYYY (Complet) | Lundi, le 29 Octobre, 1990 |

## Formats d'heure

Un *format d'heure* est une notation abrégée déterminant la façon dont l'heure s'affichera. Elle comprend des signes de ponctuation, des littéraux et des symboles. Le tableau suivant propose des exemples de formats d'heure.

| Format d'heure | Exemple |
|---|---|
| h:MM A | 7:15 PM |
| HH:MM:SS | 21:35:26 |
| HH:MM:SS 'o''clock' A Z | 14:20:10 o'clock PM EDT |

Le format d'heure est géré par une norme ISO. Chaque nation définit ses formats d'heure par défaut, court, moyen, long, et complet. Le paramètre régional identifie le format d'heure conforme aux normes de cette nation.

Le tableau ci-dessous indique quelques exemples de formats d'heure, dans différents paramètres régionaux, pour chacune des quatre catégories.

| Identificateur de paramètre régional et description | Format d'heure (Catégorie) | Exemple |
|---|---|---|
| en_GB<br>Anglais (Royaume-Uni) | HH:MM (Court) | 14:13 |
| fr_CA<br>Français (Canada) | HH:MM:SS (Moyen) | 12:15:50 |
| de_DE<br>Allemand (Allemagne) | HH:MM:SS z (Long) | 14:13:13 -0400 |
| fr_FR<br>Français (France) | HH ' h ' MM Z (Complet) | 14 h 13 GMT-04:00 |

## Formats d'image de date et d'heure

Vous devez utiliser les symboles suivants pour créer des formats de date et heure pour les champs de date/heure. Certains symboles de dates sont uniquement utilisés dans les paramètres régionaux correspondant aux langues chinoise, japonaise et coréenne. Ces symboles sont également indiqués ci-dessous.

**Remarque :** la virgule (,), le tiret (-), le deux-points (:), la barre oblique (/), le point (.) et l'espace ( ) sont considérés comme des valeurs littérales qui peuvent être incluses n'importe où dans le format. Pour inclure du texte dans un format, délimitez la chaîne de texte par des guillemets simples ('). Par exemple, `'Your payment is due no later than' MM-DD-YY` peut être spécifié comme format d'affichage.

| **Symboles de date** | **Description** | **Valeur formatée pour un paramètre régional Anglais (Etats-Unis) dont la valeur d'entrée tributaire du format du paramètre régional est 1/1/08 (Janvier 1, 2008)** |
|---|---|---|
| `D` | Jour du mois à 1 ou 2 chiffres (1-31) | 1 |
| `DD` | Jour du mois à 2 chiffres avec zéro de remplissage (01-31) | 01 |
| `J` | Jour de l'année à 1, 2 ou 3 chiffres (1-366) | 1 |
| `JJJ` | Jour de l'année à 3 chiffres avec zéros de remplissage (001-366) | 001 |
| `M` | Mois de l'année à 1 ou 2 chiffres (1-12) | 1 |
| `MM` | Mois de l'année à 2 chiffres avec zéro de remplissage (01-12) | 01 |
| `MMM` | Nom abrégé du mois | Jan |
| `MMMM` | Nom complet du mois | Janvier |
| `E` | Jour de la semaine à 1 chiffre (1-7), où 1=Dimanche | 3 (le 1er janvier 2008 correspond à un mardi) |
| `EEE` | Nom abrégé du jour de la semaine | Mar (le 1er janvier 2008 correspond à un mardi) |
| `EEEE` | Nom complet du jour de la semaine | Mardi (le 1er janvier 2008 correspond à un mardi) |
| `YY` | Année à 2 chiffres, où les nombres inférieurs à 30 se situent après l'an 2000 et où les nombres égaux et supérieurs à 30 se situent avant l'an 2000. Par exemple, 00=2000, 29=2029, 30=1930 et 99=1999. | 08 |
| `YYYY` | Année à 4 chiffres | 2008 |
| `G` | Nom de l'ère (av. J.-C. ou ap. J.-C.) | ap. J.-C. |

| Symboles de date | Description | Valeur formatée pour un paramètre régional Anglais (Etats-Unis) dont la valeur d'entrée tributaire du format du paramètre régional est 1/1/08 (Janvier 1, 2008) |
|---|---|---|
| `w` | Semaine du mois à 1 chiffre (0-5), où la semaine 1 est le premier groupe de 4 jours qui se termine un samedi | 1 |
| `WW` | Semaine de l'année à 2 chiffres (01-53) ISO-8601, où la semaine 1 représente la semaine qui contient le 4 janvier | 01 |

Vous disposez de plusieurs autres modèles de dates correspondant aux paramètres régionaux chinois, japonais et coréen.

Les ères japonaises sont représentables à l'aide de plusieurs symboles différents. Les quatre derniers symboles d'ère constituent d'autres symboles possibles pour représenter les ères japonaises.

| Symbole de date CJK | Description |
|---|---|
| `DDD` | Jour du mois du paramètre régional, exprimé en valeur numérique idéographique |
| `DDDD` | Jour du mois du paramètre régional, exprimé en valeur numérique idéographique (règle des dixièmes) |
| `YYY` | Année du paramètre régional, exprimée en valeur numérique idéographique |
| `YYYYY` | Année du paramètre régional, exprimée en valeur numérique idéographique (règle des dixièmes) |
| `g` | Autre nom possible désignant l'ère du paramètre régional. Pour l'ère actuelle japonaise, Heisei, ce modèle affiche la lettre H ASCII (U+48) |
| `gg` | Autre nom possible désignant l'ère du paramètre régional. Pour l'ère actuelle japonaise, ce modèle affiche l'idéogramme représenté par le symbole Unicode (U+5E73) |
| `ggg` | Autre nom possible désignant l'ère du paramètre régional. Pour l'ère actuelle japonaise, ce modèle affiche les idéogrammes représentés par les symboles Unicode (U+5E73 U+6210) |
| `g` | Autre nom possible désignant l'ère du paramètre régional. Pour l'ère actuelle japonaise, ce modèle affiche la lettre H à pleine largeur (U+FF28) |
| `g g` | Autre nom possible désignant l'ère du paramètre régional. Pour l'ère actuelle japonaise, ce modèle affiche l'idéogramme représenté par le symbole Unicode (U+337B) |

| Symbole de l'heure | Description | Valeur d'entrée tributaire du format du paramètre régional | Valeur formatée du paramètre régional Anglais (Etats-Unis) |
|---|---|---|---|
| h | Heure au format du jour (AM/PM) à 1 ou à 2 chiffres (1-12) | 12:08 AM ou 2:08 PM | 12 ou 2 |
| hh | Heure au format du jour (AM/PM) à 2 chiffres avec zéro de remplissage (01-12) | 12:08 AM ou 2:08 PM | 12 ou 02 |
| k | Heure au format du jour (AM/PM) à 0 ou à 2 chiffres (1-11) | 12:08 AM ou 2:08 PM | 0 ou 2 |
| kk | Heure au format du jour (AM/PM) à 2 chiffres (00-11) | 12:08 AM ou 2:08 PM | 00 ou 02 |
| H | Heure du jour à 1 ou 2 chiffres (0-23) | 12:08 AM ou 2:08 PM | 0 ou 14 |
| HH | Heure du jour à 2 chiffres avec zéro de remplissage (00-23) | 12:08 AM ou 2:08 PM | 00 ou 14 |
| K | Heure du jour à 1 ou 2 chiffres (1-24) | 12:08 AM ou 2:08 PM | 24 ou 14 |
| KK | Heure du jour à 2 chiffres avec zéro de remplissage (01-24) | 12:08 AM ou 2:08 PM | 24 ou 14 |
| M | Minute de l'heure à 1 ou 2 chiffres (0-59)<br>**Remarque :** vous devez utiliser ce symbole dans un symbole d'heure. | 2:08 PM | 8 |
| MM | Minute de l'heure à 2 chiffres avec zéro de remplissage (00-59)<br>**Remarque :** vous devez utiliser ce symbole dans un symbole d'heure. | 2:08 PM | 08 |
| S | Seconde de la minute à 1 ou 2 chiffres (0-59)<br>**Remarque :** vous devez utiliser ce symbole dans un symbole d'heure et de minutes. | 2:08:09 PM | 9 |
| SS | Seconde de la minute à 2 chiffres avec zéro de remplissage (00-59)<br>**Remarque :** vous devez utiliser ce symbole dans un symbole d'heure et de minutes. | 2:08:09 PM | 09 |

| Symbole de l'heure | Description | Valeur d'entrée tributaire du format du paramètre régional | Valeur formatée du paramètre régional Anglais (Etats-Unis) |
|---|---|---|---|
| FFF | Millième de la seconde à 3 chiffres (000-999)<br><br>**Remarque :** vous devez utiliser ce symbole dans un symbole d'heure, de minute et de seconde. | 2:08:09 PM | 09 |
| A | Partie de la journée comprise entre minuit et midi (AM) ou midi et minuit (PM) | 2:08:09 PM | PM |
| z | Format de fuseau horaire ISO-8601 (par exemple, Z, +0500, -0030, -01, +0100)<br><br>**Remarque :** vous devez utiliser ce symbole dans un symbole d'heure. | 2:08:09 PM | -0400 |
| zz | Format de fuseau horaire ISO-8601 secondaire (par exemple, Z, +05:00, -00:30, -01, +01:00)<br><br>**Remarque :** vous devez utiliser ce symbole dans un symbole d'heure. | 2:08:09 PM | -04:00 |
| Z | Nom abrégé de fuseau horaire, par exemple, TMG, TMG+05:00, TMG-00:30, HNE, HAP<br><br>**Remarque :** vous devez utiliser ce symbole dans un symbole d'heure. | 2:08:09 PM | EDT |

### Symboles réservés

Les symboles qui suivent ont une signification particulière et ne peuvent pas être utilisés comme texte littéral.

| | |
|---|---|
| ? | Lorsqu'il est utilisé, ce symbole représente n'importe quel caractère. Lorsqu'il est fusionné pour l'affichage, il devient un espace. |
| * | Lorsqu'il est utilisé, ce symbole représente 0 ou le caractère d'espace Unicode. Lorsqu'il est fusionné pour l'affichage, il devient un espace. |
| + | Lorsqu'il est utilisé, ce symbole représente un ou plusieurs caractères d'espace Unicode. Lorsqu'il est fusionné pour l'affichage, il devient un espace. |

# Date

Renvoie la date active du système sous la forme du nombre de jours écoulés depuis l'époque.

### Syntaxe
`Date()`

### Paramètres

Non

### Exemples

L'expression suivante constitue un exemple d'utilisation de la fonction `Date` :

| Expression | Renvoie |
|---|---|
| `Date()` | `37875` (le nombre de jours depuis l'époque jusqu'au 12 septembre 2003) |

# Date2Num

Renvoie le nombre de jours écoulés depuis l'époque, à partir d'une chaîne de date.

### Syntaxe
`Date2Num(`*`d`* `[, `*`f`* `[, `*`k`* `]])`

### Paramètres

| Paramètre | Description |
|---|---|
| `d` | Chaîne de date respectant le format indiqué par `f` qui par ailleurs se conforme au paramètre régional indiqué par `k`. |
| `f` (facultatif) | Chaîne de format de date. Si `f` est omis, le format de date par défaut `MMM D, YYYY` est utilisé. |
| `k` (facultatif) | Chaîne d'identification du paramètre régional qui se conforme aux normes de dénomination de ce dernier. Si `k` est omis (ou n'est pas valide), le paramètre régional ambiant est utilisé. |

La fonction renvoie la valeur `0` si l'une des conditions suivantes est vraie :

- Le format de la date fournie ne correspond pas au format indiqué dans la fonction.
- Le paramètre régional ou le format de date fourni dans la fonction n'est pas valide.

Les informations fournies sont insuffisantes pour déterminer un jour unique calculé depuis l'époque (autrement dit, une information relative à la date est manquante ou incomplète).

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Date2Num` :

| Expression | Renvoie |
|---|---|
| `Date2Num("Mar 15, 1996")` | `35138` |
| `Date2Num("1/1/1900", "D/M/YYYY")` | `1` |
| `Date2Num("03/15/96", "MM/DD/YY")` | `35138` |
| `Date2Num("Aug 1,1996", "MMM D, YYYY")` | `35277` |
| `Date2Num("96-08-20", "YY-MM-DD", "fr_FR")` | `35296` |
| `Date2Num("1/3/00", "D/M/YY") - Date2Num("1/2/00", "D/M/YY")` | `29` |

## DateFmt

Renvoie une chaîne de format de date, à partir d'un style de format de date.

### Syntaxe

`DateFmt([`*n* `[, `*k* `]])`

### Paramètres

| Paramètre | Description |
|---|---|
| `n` (facultatif) | Entier qui identifie le style de format d'heure propre au paramètre régional, de la façon suivante :<br>• 1 (style court)<br>• 2 (style moyen)<br>• 3 (style long)<br>• 4 (style complet)<br>Si `n` est omis (ou n'est pas valide), la valeur de style par défaut est 0. |
| `k` (facultatif) | Chaîne d'identification du paramètre régional qui se conforme aux normes de dénomination de ce dernier. Si `k` est omis (ou n'est pas valide), le paramètre régional ambiant est utilisé. |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `DateFmt` :

| Expression | Renvoie |
|---|---|
| `DateFmt(1)` | `M/D/YY` (si le paramètre régional en_US est défini) |
| `DateFmt(2, "fr_CA")` | `YY-MM-DD` |
| `DateFmt(3, "de_DE")` | `D. MMMM YYYY` |
| `DateFmt(4, "fr_FR")` | `EEEE D' MMMM YYYY` |

# IsoDate2Num

Renvoie le nombre de jours écoulés depuis l'époque, à partir d'une chaîne de date valide.

### Syntaxe

`IsoDate2Num(d)`

### Paramètres

| Paramètre | Description |
|---|---|
| `d` | Chaîne de date valide. |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `IsoDate2Num` :

| Expression | Renvoie |
|---|---|
| `IsoDate2Num("1900")` | `1` |
| `IsoDate2Num("1900-01")` | `1` |
| `IsoDate2Num("1900-01-01")` | `1` |
| `IsoDate2Num("19960315T20:20:20")` | `35138` |
| `IsoDate2Num("2000-03-01") - IsoDate2Num("20000201")` | `29` |

# IsoTime2Num

Renvoie le nombre de millisecondes écoulées depuis l'époque, à partir d'une chaîne d'heure valide.

### Syntaxe

`IsoTime2Num(d)`

### Paramètres

| Paramètre | Description |
|---|---|
| `d` | Chaîne d'heure valide. |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `IsoTime2Num` :

| Expression | Renvoie |
|---|---|
| `IsoTime2Num("00:00:00Z")` | `1`, pour un utilisateur situé dans le fuseau horaire de l'heure normale de l'Est. |
| `IsoTime2Num("13")` | `64800001`, pour un utilisateur situé à Boston, aux États-Unis. |
| `IsoTime2Num("13:13:13")` | `76393001`, pour un utilisateur situé en Californie. |
| `IsoTime2Num("19111111T131313+01")` | `43993001`, pour un utilisateur situé dans le fuseau horaire de l'heure normale de l'Est. |

# LocalDateFmt

Renvoie une chaîne de format de date localisée, à partir d'un style de format de date.

### Syntaxe

`LocalDateFmt([n [, k ]])`

### Paramètres

| Paramètre | Description |
|---|---|
| `n` (facultatif) | Entier qui identifie le style de format de date propre au paramètre régional, de la façon suivante :<br>• 1 (style court)<br>• 2 (style moyen)<br>• 3 (style long)<br>• 4 (style complet)<br>Si `n` est omis (ou n'est pas valide), la valeur de style par défaut est `0`. |
| `k` (facultatif) | Chaîne d'identification du paramètre régional qui se conforme aux normes de dénomination de ce dernier. Si `k` est omis (ou n'est pas valide), le paramètre régional ambiant est utilisé. |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `LocalDateFmt` :

| Expression | Renvoie |
|---|---|
| `LocalDateFmt(1, "de_DE")` | `tt.MM.uu` |
| `LocalDateFmt(2, "fr_CA")` | `aa-MM-jj` |
| `LocalDateFmt(3, "de_CH")` | `t. MMMM jjjj` |
| `LocalDateFmt(4, "fr_FR")` | `EEEE j MMMM aaaa` |

# LocalTimeFmt

Renvoie une chaîne de format d'heure localisée, à partir d'un style de format d'heure.

### Syntaxe

`LocalTimeFmt([n [, k ]])`

### Paramètres

| Paramètre | Description |
|---|---|
| `n` (Facultatif) | Entier qui identifie le style de format d'heure propre au paramètre régional, de la façon suivante :<br>• 1 (style court)<br>• 2 (style moyen)<br>• 3 (style long)<br>• 4 (style complet)<br>Si `n` est omis (ou n'est pas valide), la valeur de style par défaut est `0`. |
| `k` (Facultatif) | Chaîne d'identification du paramètre régional qui se conforme aux normes de dénomination de ce dernier. Si `k` est omis (ou n'est pas valide), le paramètre régional ambiant est utilisé. |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `LocalTimeFmt` :

| Expression | Renvoie |
|---|---|
| `LocalTimeFmt(1, "de_DE")` | `HH:mm` |
| `LocalTimeFmt(2, "fr_CA")` | `HH:mm:ss` |
| `LocalTimeFmt(3, "de_CH")` | `HH:mm:ss z` |
| `LocalTimeFmt(4, "fr_FR")` | `HH' h 'mm z` |

# Num2Date

Renvoie une chaîne de date, à partir du nombre de jours écoulés depuis l'époque.

### Syntaxe

`Num2Date(n [,f [, k ]])`

### Paramètres

| Paramètre | Description |
|---|---|
| `n` | Entier représentant le nombre de jours.<br>Si `n` n'est pas valide, la fonction renvoie une erreur. |
| `f` (Facultatif) | Chaîne de format de date. Si vous omettez la valeur `f`, la fonction utilise le format de date par défaut `MMM D, YYYY`. |
| `k` (Facultatif) | Chaîne d'identification du paramètre régional qui se conforme aux normes de dénomination de ce dernier. Si vous omettez la valeur `k`, ou si `k` n'est pas valide, la fonction utilise le paramètre régional ambiant. |

La fonction renvoie la valeur `0` si l'une des conditions suivantes est vraie :

- Le format de la date fournie ne correspond pas au format indiqué dans la fonction.
- Le paramètre régional ou le format de date fourni dans la fonction n'est pas valide.

Les informations fournies sont insuffisantes pour déterminer un jour unique calculé depuis l'époque (autrement dit, une information relative à la date est manquante ou incomplète).

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Num2Date` :

| Expression | Renvoie |
|---|---|
| `Num2Date(1, "DD/MM/YYYY")` | `01/01/1900` |
| `Num2Date(35139, "DD-MMM-YYYY", "de_DE")` | `16-Mrz-1996` |
| `Num2Date(Date2Num("Mar 15, 2000") - Date2Num("98-03-15", "YY-MM-DD", "fr_CA"))` | `Jan 1, 1902` |

# Num2GMTime

Renvoie une chaîne d'heure UT, à partir du nombre de millisecondes écoulées depuis l'époque.

### Syntaxe

`Num2GMTime(`*`n`* `[,`*`f`* `[, `*`k`* `]])`

### Paramètres

| Paramètre | Description |
|---|---|
| `n` | Entier représentant le nombre de millisecondes.<br>Si `n` n'est pas valide, la fonction renvoie une erreur. |
| `f` (Facultatif) | Chaîne de format d'heure. Si vous omettez la valeur `f`, la fonction utilise le format d'heure par défaut `H:MM:SS A`. |
| `k` (Facultatif) | Chaîne d'identification du paramètre régional qui se conforme aux normes de dénomination de ce dernier. Si vous omettez la valeur `k`, ou si `k` n'est pas valide, la fonction utilise le paramètre régional ambiant. |

La fonction renvoie la valeur `0` si l'une des conditions suivantes est vraie :

- Le format de l'heure fournie ne correspond pas au format indiqué dans la fonction.
- Le paramètre régional ou le format d'heure fourni dans la fonction n'est pas valide.

Les informations fournies sont insuffisantes pour déterminer une heure unique calculée depuis l'époque (autrement dit, une information relative à l'heure est manquante ou incomplète).

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Num2GMTime` :

| Expression | Renvoie |
|---|---|
| `Num2GMTime(1, "HH:MM:SS")` | `00:00:00` |
| `Num2GMTime(65593001, "HH:MM:SS Z")` | `18:13:13 GMT` |
| `Num2GMTime(43993001, TimeFmt(4, "de_DE"), "de_DE")` | `12.13 Uhr GMT` |

# Num2Time

Renvoie une chaîne d'heure, à partir du nombre de millisecondes écoulées depuis l'époque.

### Syntaxe

`Num2Time(`*`n`* `[,`*`f`* `[, `*`k`* `]])`

### Paramètres

| Paramètre | Description |
|---|---|
| `n` | Entier représentant le nombre de millisecondes.<br>Si `n` n'est pas valide, la fonction renvoie une erreur. |
| `f` (Facultatif) | Chaîne de format d'heure. Si vous omettez la valeur `f`, la fonction utilise le format d'heure par défaut `H:MM:SS A`. |
| `k` (Facultatif) | Chaîne d'identification du paramètre régional qui se conforme aux normes de dénomination de ce dernier. Si vous omettez la valeur `k`, ou si `k` n'est pas valide, la fonction utilise le paramètre régional ambiant. |

La fonction renvoie la valeur `0` si l'une des conditions suivantes est vraie :

- Le format de l'heure fournie ne correspond pas au format indiqué dans la fonction.
- Le paramètre régional ou le format d'heure fourni dans la fonction n'est pas valide.

Les informations fournies sont insuffisantes pour déterminer une heure unique calculée depuis l'époque (autrement dit, une information relative à l'heure est manquante ou incomplète).

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Num2Time` :

| Expression | Renvoie |
|---|---|
| `Num2Time(1, "HH:MM:SS")` | `00:00:00` à Greenwich, en Angleterre, et `09:00:00` à Tokyo. |
| `Num2Time(65593001, "HH:MM:SS Z")` | `13:13:13` EST à Boston, aux États-Unis. |
| `Num2Time(65593001, "HH:MM:SS Z", "de_DE")` | `13:13:13 GMT-05:00` pour un utilisateur suisse allemand à Boston, aux Etats-Unis. |
| `Num2Time(43993001, TimeFmt(4, "de_DE"), "de_DE")` | `13.13 Uhr GMT+01:00` pour un utilisateur situé à Zurich, en Suisse. |
| `Num2Time(43993001, "HH:MM:SSzz")` | `13:13+01:00` pour un utilisateur situé à Zurich, en Suisse. |

# Time

Renvoie l'heure actuelle du système sous la forme du nombre de millisecondes écoulées depuis l'époque.

### Syntaxe
`Time()`

### Paramètres

Non

### Exemples

L'expression suivante constitue un exemple d'utilisation de la fonction `Time` :

| Expression | Renvoie |
|---|---|
| `Time()` | `71533235` à précisément 15:52:15 le 15 septembre 2003 pour un utilisateur situé dans le fuseau horaire de l'heure normale de l'Est (HNE). |

# Time2Num

Renvoie le nombre de millisecondes écoulées depuis l'époque, à partir d'une chaîne d'heure.

### Syntaxe
`Time2Num(`*`d`* `[, `*`f`* `[, `*`k`* `]])`

### Paramètres

| Paramètre | Description |
|---|---|
| `d` | Chaîne d'heure respectant le format indiqué par `f` qui par ailleurs se conforme au paramètre régional indiqué par `k`. |
| `f` (Facultatif) | Chaîne de format d'heure. Si vous omettez la valeur `f`, la fonction utilise le format d'heure par défaut `H:MM:SS A`. |
| `k` (Facultatif) | Chaîne d'identification du paramètre régional qui se conforme aux normes de dénomination de ce dernier. Si vous omettez la valeur `k`, ou si `k` n'est pas valide, la fonction utilise le paramètre régional ambiant. |

La fonction renvoie la valeur `0` si l'une des conditions suivantes est vraie :

- Le format de l'heure fournie ne correspond pas au format indiqué dans la fonction.
- Le paramètre régional ou le format d'heure fourni dans la fonction n'est pas valide.

Les informations fournies sont insuffisantes pour déterminer une heure unique calculée depuis l'époque (autrement dit, une information relative à l'heure est manquante ou incomplète).

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Time2Num` :

| Expression | Renvoie |
|---|---|
| `Time2Num("00:00:00 GMT", "HH:MM:SS Z")` | `1` |
| `Time2Num("1:13:13 PM")` | `76393001` pour un utilisateur situé en Californie qui suit l'heure normale du Pacifique, et `76033001` lorsque le même utilisateur suit l'heure avancée du Pacifique. |
| `Time2Num("13:13:13", "HH:MM:SS") - Time2Num("13:13:13 GMT", "HH:MM:SS Z")) / (60 * 60 * 1000)` | `8` pour un utilisateur situé à Vancouver et `5` pour un utilisateur situé à Ottawa, à l'heure normale. A l'heure avancée, les valeurs renvoyées sont `7` et `4`, respectivement. |
| `Time2Num("13:13:13 GMT", "HH:MM:SS Z", "fr_FR")` | `47593001` |

## TimeFmt

Renvoie un format d'heure, à partir d'un style de format d'heure.

### Syntaxe

`TimeFmt([`*n* `[, `*k* `]])`

### Paramètres

| Paramètre | Description |
|---|---|
| `n` (Facultatif) | Entier qui identifie le style de format d'heure propre au paramètre régional, de la façon suivante :<br>• 1 (style court)<br>• 2 (style moyen)<br>• 3 (style long)<br>• 4 (style complet)<br>Si vous omettez la valeur `n`, ou si `n` n'est pas valide, la fonction utilise la valeur de style par défaut. |
| `k` (Facultatif) | Chaîne d'identification du paramètre régional qui se conforme aux normes de dénomination de ce dernier. Si `k` est omis (ou n'est pas valide), le paramètre régional ambiant est utilisé. |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `TimeFmt` :

| Expression | Renvoie |
|---|---|
| `TimeFmt(1)` | `h:MM A` (if en_US locale is set) |
| `TimeFmt(2, "fr_CA")` | `HH:MM:SS` |
| `TimeFmt(3, "fr_FR")` | `HH:MM:SS Z` |
| `TimeFmt(4, "de_DE")` | `H.MM' Uhr 'Z` |

# 6 Fonctions financières

Ces fonctions effectuent différents calculs d'intérêt, de principal et d'évaluation liés au secteur financier.

**Fonctions**



## Apr

Renvoie le taux annuel d'un prêt.

**Remarque :** les méthodes de calcul du taux d'intérêt varient d'un pays à l'autre. Cette fonction calcule le taux d'intérêt d'après les normes américaines.

**Syntaxe**

`Apr(n1, n2, n3)`

**Paramètres**

| Paramètre | Description |
|---|---|
| `n1` | Valeur ou expression numérique représentant le montant du principal du prêt. |
| `n2` | Valeur ou expression numérique représentant le montant du versement envers le remboursement du prêt. |
| `n3` | Valeur ou expression numérique représentant le nombre de périodes correspondant à la durée du prêt. |

Si un paramètre est nul, la fonction renvoie `null`. Si un paramètre est négatif ou équivalent à 0, la fonction renvoie une erreur.

**Remarque :** FormCalc respecte la norme internationale IEEE-754 en matière de gestion des valeurs numériques à virgule flottante. Pour plus de détails, voir « Littéraux numériques » à la page 8.

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Apr` :

| Expression | Renvoie |
|---|---|
| `Apr(35000, 269.50, 360)` | `0.08515404566`pour un prêt de 35 000 $ remboursé à raison de 269.50 $ par mois pendant 30 ans. |
| `Apr(210000 * 0.75, 850 + 110, 25 * 26)` | `0.07161332404` |
| `Apr(-20000, 250, 120)` | Erreur |
| `Apr(P_Value, Payment, Time)` | Cet exemple utilise des variables au lieu de valeurs ou expressions numériques. |

## CTerm

Renvoie le nombre de périodes nécessaires pour qu'un investissement, dont le taux d'intérêt est fixe mais composé, atteigne une valeur capitalisée.

**Remarque :** les méthodes de calcul du taux d'intérêt varient d'un pays à l'autre. Cette fonction calcule le taux d'intérêt d'après les normes américaines.

### Syntaxe

`CTerm(`*`n1`*`, `*`n2`*`, `*`n3`*`)`

### Paramètres

| Paramètre | Description |
|---|---|
| `n1` | Valeur ou expression numérique représentant le taux d'intérêt par période. |
| `n2` | Valeur ou expression numérique représentant la valeur capitalisée de l'investissement. |
| `n3` | Valeur ou expression numérique représentant le montant de l'investissement de départ. |

Si un paramètre est nul, la fonction renvoie null. Si un paramètre est négatif ou équivalent à 0, la fonction renvoie une erreur.

**Remarque :** FormCalc respecte la norme internationale IEEE-754 en matière de gestion des valeurs numériques à virgule flottante. Pour plus de détails, voir « Littéraux numériques » à la page 8.

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `CTerm` :

| Expression | Renvoie |
|---|---|
| `CTerm(0.02, 1000, 100)` | `116.2767474515` |
| `CTerm(0.10, 500000, 12000)` | `39.13224648502` |
| `CTerm(0.0275 + 0.0025, 1000000, 55000 * 0.10)` | `176.02226044975` |
| `CTerm(Int_Rate, Target_Amount, P_Value)` | Cet exemple utilise des variables au lieu de valeurs ou expressions numériques. |

# FV

Renvoie la valeur capitalisée de paiements fixes effectués à intervalles réguliers, compte tenu d'un taux d'intérêt fixe.

**Remarque :** les méthodes de calcul du taux d'intérêt varient d'un pays à l'autre. Cette fonction calcule le taux d'intérêt d'après les normes américaines.

### Syntaxe

`FV(n1, n2, n3)`

### Paramètres

| Paramètre | Description |
|---|---|
| `n1` | Valeur ou expression numérique représentant le montant du versement. |
| `n2` | Valeur ou expression numérique représentant l'intérêt par période de l'investissement. |
| `n3` | Valeur ou expression numérique représentant le nombre total de périodes de versement. |

La fonction renvoie une erreur si l'une des conditions suivantes est vraie :

- `n1` ou `n3` est négatif ou équivalent à 0.
- `n2` est négatif.

Si un paramètre est nul, la fonction renvoie nul.

**Remarque :** FormCalc respecte la norme internationale IEEE-754 en matière de gestion des valeurs numériques à virgule flottante. Pour plus de détails, voir « Littéraux numériques » à la page 8.

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `FV` :

| Expression | Renvoie |
|---|---|
| `FV(400, 0.10 / 12, 30 * 12)` | `904195.16991842445`. Valeur, après 30 ans, d'un investissement de 400 $ par mois rapportant 10 % annuellement. |
| `FV(1000, 0.075 / 4, 10 * 4)` | `58791.96145535981`. Valeur, après 10 ans, d'un investissement de 1 000 $ par mois rapportant 7,5 % par trimestre. |
| `FV(Payment[0], Int_Rate / 4, Time)` | Cet exemple utilise des variables au lieu de valeurs ou expressions numériques. |

# IPmt

Renvoie le montant de l'intérêt payé pour un prêt au cours d'une période donnée.

**Remarque :** les méthodes de calcul du taux d'intérêt varient d'un pays à l'autre. Cette fonction calcule le taux d'intérêt d'après les normes américaines.

### Syntaxe

`IPmt(n1, n2, n3, n4, n5)`

### Paramètres

| Paramètre | Description |
|---|---|
| `n1` | Valeur ou expression numérique représentant le montant du principal du prêt. |
| `n2` | Valeur ou expression numérique représentant le taux d'intérêt annuel de l'investissement. |
| `n3` | Valeur ou expression numérique représentant le montant du versement mensuel. |
| `n4` | Valeur ou expression numérique représentant le premier mois où un paiement est effectué. |
| `n5` | Valeur ou expression numérique représentant le nombre de mois à prendre en compte dans le calcul. |

La fonction renvoie une erreur si l'une des conditions suivantes est vraie :

- `n1`, `n2` ou `n3` est négatif ou équivalent à 0.
- `n4` ou `n5` est négatif.

Si un paramètre est nul, la fonction renvoie null. Si le montant du versement (`n3`) est inférieur aux intérêts mensuels, la fonction renvoie `0`.

**Remarque :** FormCalc respecte la norme internationale IEEE-754 en matière de gestion des valeurs numériques à virgule flottante. Pour plus de détails, voir « Littéraux numériques » à la page 8.

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `IPmt` :

| Expression | Renvoie |
|---|---|
| `IPmt(30000, 0.085, 295.50, 7, 3)` | `624.8839283142`. Montant des intérêts versés sur un prêt de 30 000 $ contracté à 8,5 %, pour les trois mois situés entre les septième et dixième mois du terme du prêt. |
| `IPmt(160000, 0.0475, 980, 24, 12)` | `7103.80833569485`. Montant du principal remboursé pendant la troisième année du prêt. |
| `IPmt(15000, 0.065, 65.50, 15, 1)` | `0`, puisque le versement mensuel est inférieur aux intérêts courus chaque mois. |

# NPV

Renvoie la valeur nette actualisée d'un investissement, compte tenu d'une série de mouvements futurs de l'encaisse et d'un taux d'actualisation.

**Remarque :** les méthodes de calcul du taux d'intérêt varient d'un pays à l'autre. Cette fonction calcule le taux d'intérêt d'après les normes américaines.

### Syntaxe

`NPV(n1, n2 [, ...])`

### Paramètres

| Paramètre | Description |
|---|---|
| `n1` | Valeur ou expression numérique représentant le taux d'actualisation au cours d'une période. |
| `n2` | Valeur ou expression numérique représentant une valeur de mouvement d'encaisse, qui doit survenir à la fin de la période. Il est important que les valeurs indiquées dans `n2` et au-delà soient dans l'ordre adéquat. |

La fonction renvoie une erreur si la valeur de `n1` est négative ou égale à 0. Si l'un des paramètres est nul, la fonction renvoie « null ».

**Remarque :** FormCalc respecte la norme internationale IEEE-754 en matière de gestion des valeurs numériques à virgule flottante. Pour plus de détails, voir « Littéraux numériques » à la page 8.

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `NPV` :

| Expression | Renvoie |
|---|---|
| `NPV(0.065, 5000)` | `4694.83568075117`, qui est la valeur nette actualisée d'un investissement rapportant 6,5 % par an et qui générera 5 000 $. |
| `NPV(0.10, 500, 1500, 4000, 10000)` | `11529.60863329007`, qui est la valeur nette actualisée d'un investissement rapportant 10 % par année et qui générera 500 $, 1 500 $, 4 000 $ et 10 000 $ pour chacune des quatre prochaines années. |
| `NPV(0.0275 / 12, 50, 60, 40, 100, 25)` | `273.14193838457`, qui est la valeur nette actualisée d'un investissement rapportant 2,75 % par an et qui générera 50 $, 60 $, 40 $, 100 $ et 25 $ pour chacun des cinq prochains mois. |

# Pmt

Renvoie le remboursement d'un prêt basé sur des versements fixes et sur un taux d'intérêt fixe.

**Remarque :** les méthodes de calcul du taux d'intérêt varient d'un pays à l'autre. Cette fonction calcule le taux d'intérêt d'après les normes américaines.

### Syntaxe

`Pmt(n1, n2, n3)`

### Paramètres

| Paramètre | Description |
|---|---|
| `n1` | Valeur ou expression numérique représentant le montant du principal du prêt. |
| `n2` | Valeur ou expression numérique représentant le taux d'intérêt par période de l'investissement. |
| `n3` | Valeur ou expression numérique représentant le nombre total de périodes de versement. |

La fonction renvoie une erreur si l'un des paramètres est négatif ou égal à 0. Si l'un des paramètres est nul, la fonction renvoie « null ».

**Remarque :** FormCalc respecte la norme internationale IEEE-754 en matière de gestion des valeurs numériques à virgule flottante. Pour plus de détails, voir « Littéraux numériques » à la page 8.

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Pmt` :

| Expression | Renvoie |
|---|---|
| `Pmt(150000, 0.0475 / 12, 25 * 12)` | `855.17604207164`, qui est le versement mensuel d'un prêt de 150 000 $ assorti d'un taux d'intérêt annuel de 4,75 %, remboursable en 25 ans. |
| `Pmt(25000, 0.085, 12)` | `3403.82145169876`, qui est le versement annuel d'un prêt de 25 000 $ assorti d'un taux d'intérêt annuel de 8,5 %, remboursable en 12 ans. |

# PPmt

Renvoie le montant du principal payé pour un prêt au cours d'une période.

**Remarque :** les méthodes de calcul du taux d'intérêt varient d'un pays à l'autre. Cette fonction calcule le taux d'intérêt d'après les normes américaines.

### Syntaxe

`PPmt(n1, n2, n3, n4, n5)`

### Paramètres

| Paramètre | Description |
|---|---|
| `n1` | Valeur ou expression numérique représentant le montant du principal du prêt. |
| `n2` | Valeur ou expression numérique représentant le taux d'intérêt annuel. |
| `n3` | Valeur ou expression numérique représentant le montant du versement mensuel. |
| `n4` | Valeur ou expression numérique représentant le premier mois où un paiement est effectué. |
| `n5` | Valeur ou expression numérique représentant le nombre de mois à prendre en compte dans le calcul. |

La fonction renvoie une erreur si l'une des conditions suivantes est vraie :

- `n1`, `n2` ou `n3` est négatif ou équivalent à 0.
- `n4` ou `n5` est négatif.

Si un paramètre est nul, la fonction renvoie null. Si le montant du versement (`n3`) est inférieur aux intérêts mensuels, la fonction renvoie `0`.

**Remarque :** FormCalc respecte la norme internationale IEEE-754 en matière de gestion des valeurs numériques à virgule flottante. Pour plus de détails, voir « Littéraux numériques » à la page 8.

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `PPmt` :

| Expression | Renvoie |
|---|---|
| `PPmt(30000, 0.085, 295.50, 7, 3)` | `261.6160716858`, qui est le montant du principal remboursé sur un prêt de 30 000 $ contracté à 8,5 %, pour les trois mois situés entre les septième et dixième mois du terme du prêt. |
| `PPmt(160000, 0.0475, 980, 24, 12)` | `4656.19166430515`, qui est le montant du principal remboursé pendant la troisième année du prêt. |
| `PPmt(15000, 0.065, 65.50, 15, 1)` | `0`, car dans le cas présent, le versement mensuel est inférieur aux intérêts courus chaque mois et aucune partie du principal n'a été remboursée. |

# PV

Renvoie la valeur actualisée d'un investissement réalisé par versements fixes périodiques avec un taux d'intérêt fixe.

**Remarque :** les méthodes de calcul du taux d'intérêt varient d'un pays à l'autre. Cette fonction calcule le taux d'intérêt d'après les normes américaines.

### Syntaxe

`PV(n1, n2, n3)`

### Paramètres

| Paramètre | Description |
|---|---|
| `n1` | Valeur ou expression numérique représentant le montant du versement. |
| `n2` | Valeur ou expression numérique représentant l'intérêt par période de l'investissement. |
| `n3` | Valeur ou expression numérique représentant le nombre total de périodes de versement. |

La fonction renvoie une erreur si la valeur de `n1` ou de `n3` est négative ou égale à 0. Si l'un des paramètres est nul, la fonction renvoie « null ».

**Remarque :** FormCalc respecte la norme internationale IEEE-754 en matière de gestion des valeurs numériques à virgule flottante. Pour plus de détails, voir « Littéraux numériques » à la page 8.

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `PV` :

| Expression | Renvoie |
|---|---|
| `PV(400, 0.10 / 12, 30 * 12)` | `45580.32799074439`. Valeur, après 30 ans, d'un investissement de 400 $ par mois rapportant 10 % annuellement. |
| `PV(1000, 0.075 / 4, 10 * 4)` | `58791.96145535981`. Valeur, après dix ans, d'un investissement de 1 000 $ par mois rapportant 7,5 % par trimestre. |
| `PV(Payment[0], Int_Rate / 4, Time)` | Cet exemple utilise des variables au lieu de valeurs ou expressions numériques. |

# Rate

Renvoie le taux d'intérêt composé par période nécessaire pour qu'un investissement d'une valeur actualisée atteigne une valeur capitalisée au cours d'une période donnée.

**Remarque :** les méthodes de calcul du taux d'intérêt varient d'un pays à l'autre. Cette fonction calcule le taux d'intérêt d'après les normes américaines.

### Syntaxe

`Rate(n1, n2, n3)`

### Paramètres

| Paramètre | Description |
|---|---|
| `n1` | Valeur ou expression numérique représentant la valeur capitalisée de l'investissement. |
| `n2` | Valeur ou expression numérique représentant la valeur actualisée de l'investissement. |
| `n3` | Valeur ou expression numérique représentant le nombre total de périodes de versement. |

La fonction renvoie une erreur si l'un des paramètres est négatif ou égal à 0. Si l'un des paramètres est nul, la fonction renvoie « null ».

**Remarque :** FormCalc respecte la norme internationale IEEE-754 en matière de gestion des valeurs numériques à virgule flottante. Pour plus de détails, voir « Littéraux numériques » à la page 8.

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Rate` :

| Expression | Renvoie |
|---|---|
| `Rate(12000, 8000, 5)` | `0.0844717712` (ou 8,45 %), qui est le taux d'intérêt par période nécessaire pour que la valeur actuelle de 8 000 $ atteigne 12 000 $ en cinq périodes. |
| `Rate(10000, 0.25 * 5000, 4 * 12)` | `0.04427378243` (ou 4,43 %), qui est le taux d'intérêt mensuel nécessaire pour que la valeur actuelle atteigne 10 000 $ en quatre ans. |
| `Rate(Target_Value, Pres_Value[*], Term * 12)` | Cet exemple utilise des variables au lieu de valeurs ou expressions numériques. |

# Term

Renvoie le nombre de périodes nécessaires pour atteindre la valeur capitalisée donnée, compte tenu de paiements fixes périodiques versés sur un compte portant intérêt.

**Remarque :** les méthodes de calcul du taux d'intérêt varient d'un pays à l'autre. Cette fonction calcule le taux d'intérêt d'après les normes américaines.

### Syntaxe

`Term(n1, n2, n3)`

### Paramètres

| Paramètre | Description |
|---|---|
| `n1` | Valeur ou expression numérique représentant le montant du versement effectué à la fin de chaque période. |
| `n2` | Valeur ou expression numérique représentant le taux d'intérêt par période de l'investissement. |
| `n3` | Valeur ou expression numérique représentant la valeur capitalisée de l'investissement. |

La fonction renvoie une erreur si l'un des paramètres est négatif ou égal à 0. Si l'un des paramètres est nul, la fonction renvoie « null ».

**Remarque :** FormCalc respecte la norme internationale IEEE-754 en matière de gestion des valeurs numériques à virgule flottante. Pour plus de détails, voir « Littéraux numériques » à la page 8.

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Term` :

| Expression | Renvoie |
|---|---|
| `Term(475, .05, 1500)` | `3.00477517728` (soit environ 3), qui est le nombre de périodes nécessaires pour qu'un versement de 475 $ atteigne 1 500 $, avec un taux d'intérêt de 5 % par période. |
| `Term(2500, 0.0275 + 0.0025, 5000)` | `1.97128786369`, qui est le nombre de périodes nécessaires pour que des versements de 2 500 $ atteignent 5 000 $, avec un taux d'intérêt de 3 % par période. |
| `Rate(Inv_Value[0], Int_Rate + 0.0050, Target_Value)` | Cet exemple utilise des variables au lieu de valeurs ou expressions numériques. Dans le présent cas, la première occurrence de la variable `Inv_Value` sert de montant de versement, un demi point de pourcentage est ajouté à la variable `Int_Rate` qui représente le taux d'intérêt, et la variable `Target_Value` correspond à la valeur capitalisée de l'investissement. |

# 7 Fonctions logiques

Ces fonctions sont utiles pour tester et analyser des informations dans le but d'obtenir un résultat vrai ou faux.

**Fonctions**



## Choose

Sélectionne une valeur dans un ensemble de paramètres donné.

**Syntaxe**

`Choose(n, s1 [, s2 ...])`

**Paramètres**

| Paramètre | Description |
|---|---|
| `n` | Position de la valeur que vous souhaitez sélectionner dans l'ensemble. Si cette valeur n'est pas un nombre entier, la fonction arrondit `n` en faisant fi des décimales.<br>Cette fonction renvoie une chaîne vide si l'une ou l'autre des conditions suivantes est vraie :<br>• `n` est inférieur à 1.<br>• `n` est supérieur au nombre d'éléments contenus dans la liste.<br>Si `n` est nul, la fonction renvoie « null ». |
| `s1` | Première valeur de l'ensemble de valeurs. |
| `s2` (Facultatif) | Autres valeurs de l'ensemble. |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Choose` :

| Expression | Renvoie |
|---|---|
| `Choose(3, "Taxes", "Price", "Person", "Teller")` | `Person` |
| `Choose(2, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1)` | `9` |
| `Choose(Item_Num[0], Items[*])` | Renvoie la valeur de l'ensemble `Items` qui correspond à la position définie par la première occurrence de `Item_Num`. |
| `Choose(20/3, "A", "B", "C", "D", "E", "F", "G", "H")` | `F` |

# Exists

Détermine si le paramètre donné est une syntaxe de référence à un objet existant.

### Syntaxe
`Exists(`*v*`)`

### Paramètres

| Paramètre | Description |
|---|---|
| `v` | Expression de syntaxe de référence valide.<br>Si `v` n'est pas une syntaxe de référence, la fonction renvoie « false » (`0`). |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Exists` :

| Expression | Renvoie |
|---|---|
| `Exists(Item)` | True (`1`) si l'objet `Item` existe et false (`0`) dans le cas contraire. |
| `Exists("hello world")` | False (`0`). La chaîne n'est pas une syntaxe de référence. |
| `Exists(Invoice.Border.Edge[1].Color)` | True (`1`) si l'objet `Invoice` existe et possède la propriété `Border`, laquelle est définie au moins par une propriété `Edge`, qui dispose quant à elle de la propriété `Color`. Sinon, la fonction renvoie false (`0`). |

# HasValue

Détermine si le paramètre indiqué est une syntaxe de référence dont la valeur n'est ni nulle, ni vide, ni un blanc.

### Syntaxe

`HasValue(v)`

### Paramètres

| Paramètre | Description |
|---|---|
| `v` | Expression de syntaxe de référence valide.<br>Si `v` n'est pas une syntaxe de référence, la fonction renvoie « false » (`0`). |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `HasValue`.

| Expression | Renvoie |
|---|---|
| `HasValue(2)` | True (`1`) |
| `HasValue(" ")` | False (`0`) |
| `HasValue(Amount[*])` | Erreur |
| `HasValue(Amount[0])` | Evalue la première occurrence de `Amount` et renvoie true (`1`) si sa valeur n'est ni nulle, ni vide, ni un blanc. |

# Oneof

Détermine si la valeur fournie fait partie d'un ensemble.

### Syntaxe

`Oneof(s1, s2 [, s3 ...])`

### Paramètres

| Paramètre | Description |
|---|---|
| `s1` | Position de la valeur que vous souhaitez sélectionner dans l'ensemble. Si cette valeur n'est pas un nombre entier, la fonction arrondit `s1` en faisant fi des décimales. |
| `s2` | Première valeur de l'ensemble de valeurs. |
| `s3` (Facultatif) | Autres valeurs de l'ensemble. |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Oneof` :

| Expression | Renvoie |
|---|---|
| `Oneof(3, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1)` | True (`1`) |
| `Oneof("John", "Bill", "Gary", "Joan", "John", "Lisa")` | True (`1`) |
| `Oneof(3, 1, 25)` | False (`0`) |
| `Oneof("loan", Fields[*])` | Détermine si une occurrence de `Fields` a la valeur de `loan`. |

## Within

Détermine si la valeur fournie est dans une plage donnée.

### Syntaxe

`Within(`*`s1`*`, `*`s2`*`, `*`s3`*`)`

### Paramètres

| Paramètre | Description |
|---|---|
| `s1` | Valeur à tester.<br>Si `s1` est un nombre, la comparaison d'ordonnancement est numérique.<br>Si `s1` n'est pas un nombre, la comparaison d'ordonnancement utilise la séquence de classement du paramètre régional actuel. Pour plus de détails, voir « Paramètres régionaux » à la page 43.<br>Si `s1` est nul, la fonction renvoie « null ». |
| `s2` | Limite inférieure de la plage de test. |
| `s3` | Limite supérieure de la plage de test. |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Within` :

| Expression | Renvoie |
|---|---|
| `Within("C", "A", "D")` | True (`1`) |
| `Within(1.5, 0, 2)` | True (`1`) |
| `Within(-1, 0, 2)` | False (`0`) |
| `Within($, 1, 10)` | True (`1`) si la valeur actuelle se situe entre 1 et 10. |

# 8 Fonctions diverses

Les fonctions décrites dans la présente section ne font partie d'aucune catégorie de fonctions particulière, et sont utiles dans de multiples applications.

**Fonctions**



## Eval

Renvoie la valeur d'un calcul de formulaire donné.

**Syntaxe**

`Eval(s)`

**Paramètres**

| Paramètre | Description |
|---|---|
| `s` | Chaîne valide représentant une expression ou une liste d'expressions.<br><br>**Remarque :** la fonction `Eval` ne peut pas faire référence à des variables ou à des fonctions définies par l'utilisateur. Par exemple :<br><br>`var s = "var t = concat(s, ""hello"")"`<br>`eval(s)`<br><br>Dans ce cas, la fonction `Eval` ne reconnaît pas `s` et renvoie une erreur. Toutes les fonctions suivantes faisant référence à la variable `s` échouent également. |

**Exemples**

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Eval` :

| Expression | Renvoie |
|---|---|
| `eval("10*3+5*4")` | `50` |
| `eval("hello")` | `error` |

# Null

Renvoie la valeur nulle. La valeur nulle signifie aucune valeur.

### Définition
`Null()`

### Paramètres

Non

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Null` :

| Expression | Renvoie |
|---|---|
| `Null()` | `null` |
| `Null() + 5` | `5` |
| `Quantity = Null()` | Assigne la valeur `null` à l'objet `Quantity`. |
| `Concat("ABC", Null(), "DEF")` | `ABCDEF`<br>Voir aussi « Concat » à la page 83. |

# Ref

Renvoie une référence à un objet existant.

### Définition
`Ref(v)`

### Paramètres

| Paramètres | Description |
|---|---|
| `v` | Chaîne valide représentant une syntaxe de référence, une propriété, une méthode ou une fonction.<br>**Remarque :** si un paramètre est nul, la fonction renvoie null. Pour tous les autres paramètres, la fonction génère une erreur. |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Ref` :

| Expressions | Renvoie |
|---|---|
| `Ref("10*3+5*4")` | `10*3+5*4` |
| `Ref("hello")` | `hello` |

# UnitType

Renvoie les unités d'une étendue d'unité. Une étendue d'unité est une chaîne composée d'un nombre suivi d'un nom d'unité.

### Syntaxe

`UnitType(s)`

### Paramètres

| Paramètre | Description |
|---|---|
| `s` | Chaîne valide contenant une valeur numérique et une unité de mesure valide (unitspan). Unités de mesure reconnues :<br>• in, inches (pouces)<br>• mm, millimètres<br>• cm, centimètres<br>• pt, picas, points<br>• mp, millipoints<br>Si `s` n'est pas valide, la fonction renvoie `in`. |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `UnitType` :

| Expression | Résultats |
|---|---|
| `UnitType("36 in")` | `in` |
| `UnitType("2.54centimeters")` | `cm` |
| `UnitType("picas")` | `pt` |
| `UnitType("2.cm")` | `cm` |
| `UnitType("2.zero cm")` | `in` |
| `UnitType("kilometers")` | `in` |
| `UnitType(Size[0])` | Renvoie la valeur de la mesure de la première occurrence de `Size`. |

# UnitValue

Renvoie la valeur numérique d'une mesure avec l'étendue d'unité qui lui est associée, après une conversion d'unité facultative. Une étendue d'unité est une chaîne composée d'un nombre suivi d'une unité de mesure valide.

### Syntaxe

`UnitValue(`*`s1`* `[, `*`s2`* `])`

### Paramètres

| Paramètres | Description |
|---|---|
| `s1` | Chaîne valide contenant une valeur numérique et une unité de mesure valide (unitspan). Unités de mesure reconnues :<br>• in, inches (pouces)<br>• mm, millimètres<br>• cm, centimètres<br>• pt, picas, points<br>• mp, millipoints |
| `s2` (facultatif) | Chaîne contenant une unité de mesure valide. La fonction convertit l'unitspan indiquée dans `s1` en cette nouvelle unité de mesure.<br>Si vous n'incluez pas une valeur pour `s2`, la fonction utilise l'unité de mesure indiquée dans `s1`. Si `s2` n'est pas valide, la fonction convertit `s1` en pouces. |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `UnitValue` :

| Expression | Renvoie |
|---|---|
| `UnitValue("2in")` | `2` |
| `UnitValue("2in", "cm")` | `5.08` |
| `UnitValue("6", "pt")` | `432` |
| `UnitValue("A", "cm")` | `0` |
| `UnitValue(Size[2], "mp")` | Renvoie la valeur de la mesure de la troisième occurrence de `Size` convertie en millipoints. |
| `UnitValue("5.08cm", "kilograms")` | `2` |

# 9 Fonctions de chaîne

Les fonctions décrites dans la présente section portent sur la manipulation, l'évaluation et la création de valeurs de chaîne.

**Fonctions**

# At

Repère la position du premier caractère d'une chaîne dans une autre chaîne.

### Syntaxe

`At(s1, s2)`

### Paramètres

| Paramètre | Description |
|---|---|
| `s1` | Chaîne source. |
| `s2` | Chaîne de recherche.<br>Si `s2` n'est pas une partie de `s1`, la fonction renvoie `0`.<br>Si `s2` est vide, la fonction renvoie `1`. |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `At` :

| Expression | Renvoie |
|---|---|
| `At("ABCDEFGH", "AB")` | `1` |
| `At("ABCDEFGH", "F")` | `6` |
| `At(23412931298471, 29)` | `5`, position du premier caractère de la première occurrence de `29` à l'intérieur de la chaîne source. |
| `At(Ltrim(Cust_Info[0]), "555")` | Emplacement de la chaîne `555` à l'intérieur de la première occurrence de `Cust_Info`.<br>Voir aussi « Ltrim » à la page 89. |

# Concat

Renvoie la concaténation des chaînes indiquées.

### Syntaxe

`Concat(s1 [, s2 ...])`

### Paramètres

| Paramètre | Description |
|---|---|
| `s1` | Première chaîne de l'ensemble. |
| `s2` (Facultatif) | Autres chaînes à ajouter à l'ensemble. |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Concat` :

| Expression | Renvoie |
|---|---|
| `Concat("ABC", "DEF")` | `ABCDEF` |
| `Concat("Tony", Space(1), "Blue")` | `Tony Blue`<br>Voir aussi « Space » à la page 92. |
| `Concat("You owe ", WordNum(1154.67, 2), ".")` | `You owe One Thousand One Hundred Fifty-four Dollars And Sixty-seven Cents.`<br>Voir aussi « WordNum » à la page 97. |

# Decode

Renvoie la version décodée d'une chaîne donnée.

### Syntaxe

`Decode(`*`s1`* `[, `*`s2`* `])`

### Paramètres

| Paramètre | Description |
|---|---|
| `s1` | Chaîne à décoder. |
| `s2` (Facultatif) | Chaîne identifiant le type de décodage à effectuer. Chaînes de décodage valides :<br>• url (décodage URL)<br>• html (décodage HTML)<br>• xml (décodage XML)<br>Si vous omettez la valeur `s2`, la fonction utilise le décodage URL. |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Decode` :

| Expression | Renvoie |
|---|---|
| `Decode("&AElig;&Aacute;&Acirc;&Aacute;&Acirc;", "html")` | `ÆÁÂÁÂ` |
| `Decode("~!@#$%^&*()_+\|`{"}[]<>?,./;':", "xml")` | `~!@#$%^&*()_+\|`{""}[]<>?,./;':` |

# Encode

Renvoie la version codée d'une chaîne donnée.

### Syntaxe

`Encode(`*`s1`* `[, `*`s2`* `])`

### Paramètres

| Paramètre | Description |
|---|---|
| `s1` | Chaîne à coder. |
| `s2` (Facultatif) | Chaîne identifiant le type de codage à effectuer. Chaînes de codage valides :<br>• url (codage URL)<br>• html (codage HTML)<br>• xml (codage XML)<br>Si vous omettez la valeur `s2`, la fonction utilise le codage URL. |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Encode` :

| Expression | Renvoie |
|---|---|
| `Encode("""hello, world!""", "url")` | `%22hello,%20world!%22` |
| `Encode("ÁÂÃÄÅÆ", "html")` | `&#xc1;&#Xc2;&#Xc3;&#xc4;&#xc5;&#xc6;` |

# Format

Formate les données fournies selon la chaîne de format d'image indiquée.

### Syntaxe

`Format(s1, s2 [, s3 ...])`

### Paramètres

| Paramètre | Description |
|---|---|
| `s1` | Chaîne de format d'image, qui peut être un format de date ou d'heure prenant en compte le paramètre régional. Voir « Paramètres régionaux » à la page 43. |
| `s2` | Données source à formater.<br>Pour les formats d'image de date, les données source doivent être soit une chaîne de date/heure ISO, soit une chaîne de date ISO dans l'un ou l'autre des formats suivants :<br>• YYYY[MM[DD]]<br>• YYYY[-MM[-DD]]<br>Pour les formats d'image de date, les données source doivent être soit une chaîne de date/heure ISO, soit une chaîne de date ISO dans l'un ou l'autre des formats suivants :<br>• HH[MM[SS[.FFF][z]]]<br>• HH[MM[SS[.FFF][+HH[MM]]]]<br>• HH[MM[SS[.FFF][-HH[MM]]]]<br>• HH[:MM[:SS[.FFF][z]<br>• HH[:MM[:SS[.FFF][-HH[:MM]]]]<br>• HH[:MM[:SS[.FFF][+HH[:MM]]]]<br>Pour les formats d'image de date-heure, les données source doivent être une chaîne de date-heure ISO.<br>Pour les formats d'image numériques, les données source doivent être numériques.<br>Pour les formats d'image textuels, les données source doivent être textuelles.<br>Pour les formats d'image composés, le nombre d'arguments des données source doit correspondre au nombre de sous-éléments de l'image. |
| `s3` (Facultatif) | Autres données source à formater. |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Format` :

| Expression | Renvoie |
|---|---|
| `Format("MMM D, YYYY", "20020901")` | `Sep 1, 2002` |
| `Format("$9,999,999.99", 1234567.89)` | `$1,234,567.89` aux États-Unis et `1 234 567,89` euros en France. |

# Left

Extrait un nombre spécifique de caractères d'une chaîne, en commençant par le premier caractère à gauche.

### Syntaxe

`Left(s, n)`

### Paramètres

| Paramètre | Description |
|---|---|
| `s` | Chaîne dans laquelle l'extraction est effectuée. |
| `n` | Nombre de caractères à extraire.<br>Si le nombre de caractères à extraire est supérieur à la taille de la chaîne, la fonction renvoie toute la chaîne.<br>Si le nombre de caractères à extraire est 0 ou moins, la fonction renvoie une chaîne vide. |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Left` :

| Expression | Renvoie |
|---|---|
| `Left("ABCDEFGH", 3)` | `ABC` |
| `Left("Tony Blue", 5)` | `"Tony "` |
| `Left(Telephone[0], 3)` | Les trois premiers caractères de la première occurrence de `Telephone`. |
| `Left(Rtrim(Last_Name), 3)` | Les trois premiers caractères de `Last_Name`.<br>Voir aussi « Rtrim » à la page 91. |

# Len

Renvoie le nombre de caractères d'une chaîne donnée.

### Syntaxe

`Len(s)`

### Paramètres

| Paramètre | Description |
|---|---|
| `s` | Chaîne à évaluer. |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Len` :

| Expression | Renvoie |
|---|---|
| `Len("ABDCEFGH")` | `8` |
| `Len(4)` | `1` |
| `Len(Str(4.532, 6, 4))` | `6`<br>Voir aussi « Str » à la page 93. |
| `Len(Amount[*])` | Nombre de caractères de la première occurrence de `Amount`. |

## Lower

Convertit en minuscules toutes les majuscules d'une chaîne donnée.

### Syntaxe

`Lower(`*`s`*`, [, `*`k`*` ])`

### Paramètres

| Paramètre | Description |
|---|---|
| `s` | Chaîne à convertir. |
| `k` (Facultatif) | Chaîne représentant un paramètre régional valide. Si vous omettez la valeur `k`, la fonction utilise le paramètre régional ambiant.<br>Voir aussi « Paramètres régionaux » à la page 43.<br>**Remarque :** cette fonction convertit uniquement les caractères Unicode U+41 à U+5A (du jeu de caractères ASCII), ainsi que les caractères U+FF21 à U+FF3A (du jeu de caractères pleine largeur). |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Lower` :

| Expression | Renvoie |
|---|---|
| `Lower("ABC")` | `abc` |
| `Lower("21 Main St.")` | `21 main st.` |
| `Lower(15)` | `15` |
| `Lower(Address[0])` | Cet exemple convertit en minuscules la première occurrence de `Address`. |

# Ltrim

Renvoie une chaîne sans aucun caractère d'espace blanc à gauche.

Les caractères d'espace incluent l'espace ASCII, la tabulation horizontale, le changement de ligne, la tabulation verticale, le changement de page, le retour chariot et les caractères d'espace Unicode (catégorie Unicode Zs).

### Syntaxe

```
Ltrim(s)
```

### Paramètres

| Paramètre | Description |
|---|---|
| `s` | Chaîne à dégrossir. |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Ltrim` :

| Expression | Renvoie |
|---|---|
| `Ltrim("  ABCD")` | `"ABCD"` |
| `Ltrim(Rtrim("  Tony Blue  "))` | `"Tony Blue"`<br>Voir aussi « Rtrim » à la page 91. |
| `Ltrim(Address[0])` | Supprime tous les espaces situés à gauche de la première occurrence de `Address`. |

# Parse

Analyse les données fournies selon le format d'image donné.

Une analyse réussie des données génère l'un des résultats suivants :

- Format d'image de date : Chaîne de date ISO au format YYYY-MM-DD.
- Formats d'image d'heure : Chaîne d'heure ISO au format HH:MM:SS.
- Format d'image de date-heure : Chaîne de date-heure ISO au format YYYY-MM-DDTHH:MM:SS.
- Format d'image numérique : Un nombre.
- Images de texte : Du texte.

### Syntaxe

```
Parse(s1, s2 )
```

### Paramètres

| Paramètre | Description |
|---|---|
| `s1` | Chaîne de format d'image de date ou d'heure valide.<br>Pour plus de détails sur les formats de date et d'heure, voir « Structuration des dates et des heures » à la page 43. |
| `s2` | Données de la chaîne à analyser. |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Parse` :

| Expression | Renvoie |
|---|---|
| `Parse("MMM D, YYYY", "Sep 1, 2002")` | `2002-09-01` |
| `Parse("$9,999,999.99", "$1,234,567.89")` | `1234567.89`aux États-Unis. |

## Replace

Remplace toutes les occurrences d'une chaîne par une autre, dans la chaîne indiquée.

### Syntaxe

`Replace(`*`s1`*`, `*`s2`*` [, `*`s3`*` ])`

### Paramètres

| Paramètre | Description |
|---|---|
| `s1` | Chaîne source. |
| `s2` | Chaîne à remplacer. |
| `s3` (Facultatif) | Chaîne à utiliser pour le remplacement.<br>Si vous omettez la valeur `s3`, ou si `s3` est nul, la fonction utilise une chaîne vide. |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Replace` :

| Expression | Renvoie |
|---|---|
| `Replace("Tony Blue", "Tony", "Chris")` | `Chris Blue` |
| `Replace("ABCDEFGH", "D")` | `ABCEFGH` |
| `Replace("ABCDEFGH", "d")` | `ABCDEFGH` |
| `Replace(Comments[0], "recieve", "receive")` | Corrige les utilisations mal orthographiées du mot `receive` dans la première occurrence de `Comments`. |

# Right

Extrait un nombre spécifique de caractères dans une chaîne donnée, en commençant par le dernier caractère à droite.

### Syntaxe

```
Right(s, n )
```

### Paramètres

| Paramètre | Description |
|---|---|
| `s` | Chaîne à extraire. |
| `n` | Nombre de caractères à extraire.<br>Si `n` est supérieur à la taille de la chaîne, la fonction renvoie toute la chaîne.<br>Si `n` est 0 ou moins, la fonction renvoie une chaîne vide. |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Right` :

| Expression | Renvoie |
|---|---|
| `Right("ABCDEFGH", 3)` | `FGH` |
| `Right("Tony Blue", 5)` | `" Blue"` |
| `Right(Telephone[0], 7)` | Les sept derniers caractères de la première occurrence de `Telephone`. |
| `Right(Rtrim(CreditCard_Num), 4)` | Les quatre derniers caractères de `CreditCard_Num`.<br>Voir aussi « Rtrim » à la page 91. |

# Rtrim

Renvoie une chaîne sans aucun caractère d'espace blanc à droite.

Les caractères d'espace incluent l'espace ASCII, la tabulation horizontale, le changement de ligne, la tabulation verticale, le changement de page, le retour chariot et les caractères d'espace Unicode (catégorie Unicode Zs).

### Syntaxe

```
Rtrim(s )
```

### Paramètres

| Paramètre | Description |
|---|---|
| `s` | Chaîne à dégrossir. |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Rtrim` :

| Expression | Renvoie |
|---|---|
| `Rtrim("ABCD ")` | `"ABCD"` |
| `Rtrim("Tony Blue ")` | `"Tony Blue"` |
| `Rtrim(Address[0])` | Supprime tous les espaces situés à droite de la première occurrence de `Address`. |

# Space

Renvoie une chaîne composée d'un nombre donné d'espaces blancs.

### Syntaxe

`Space(`*n* `)`

### Paramètres

| Paramètre | Description |
|---|---|
| `n` | Nombre d'espaces. |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Space` :

| Expression | Renvoie |
|---|---|
| `Space(5)` | `"  "` |
| `Space(Max(Amount[*]))` | Chaîne vide contenant autant de caractères que la valeur de l'occurrence la plus élevée de `Amount`.<br>Voir aussi « Max » à la page 38. |
| `Concat("Tony", Space(1), "Blue")` | `Tony Blue` |

# Str

Convertit un nombre en une chaîne de caractères. FormCalc formate le résultat selon la largeur indiquée et l'arrondit au nombre de décimales défini.

### Syntaxe

`Str(n1 [, n2 [, n3 ]])`

### Paramètres

| Paramètre | Description |
|---|---|
| `n1` | Nombre à convertir. |
| `n2` (Facultatif) | Largeur maximale de la chaîne. Si vous omettez la valeur `n2`, la fonction utilise la valeur `10` comme largeur par défaut.<br>Si la chaîne obtenue est plus longue que `n2`, la fonction renvoie une chaîne d'astérisques (*) de la largeur indiquée par `n2`. |
| `n3` (Facultatif) | Nombre de chiffres après le point décimal. Si vous omettez la valeur `n3`, la fonction utilise 0 par défaut. |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Str` :

| Expression | Renvoie |
|---|---|
| `Str(2.456)` | `"          2"` |
| `Str(4.532, 6, 4)` | `4.5320` |
| `Str(234.458, 4)` | `" 234"` |
| `Str(31.2345, 4, 2)` | `****` |
| `Str(Max(Amount[*]), 6, 2)` | Convertit l'occurrence la plus élevée de `Amount` en une chaîne de six caractères comprenant deux décimales.<br>Voir aussi « Max » à la page 38. |

# Stuff

Insère une chaîne dans une autre chaîne.

### Syntaxe

`Stuff(s1, n1, n2 [, s2 ])`

### Paramètres

| Paramètre | Description |
|---|---|
| `s1` | Chaîne source. |
| `n1` | Position, dans `s1`, où la nouvelle chaîne `s2` doit être insérée.<br>Si `n1` est inférieur à un, la fonction utilise la position du premier caractère. Si `n1` est supérieur à la longueur de `s1`, la fonction utilise la position du dernier caractère. |
| `n2` | Nombre de caractères à supprimer de la chaîne `s1`, en commençant à la position du caractère `n1`.<br>Si `n2` est inférieur ou égal à 0, la fonction ne supprime aucun caractère. |
| `s2` (Facultatif) | Chaîne à insérer dans `s1`.<br>Si vous omettez la valeur `s2`, la fonction utilise une chaîne vide. |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Stuff` :

| Expression | Renvoie |
|---|---|
| `Stuff("TonyBlue", 5, 0, " ")` | `Tony Blue` |
| `Stuff("ABCDEFGH", 4, 2)` | `ABCFGH` |
| `Stuff(Address[0], Len(Address[0]), 0, "Street")` | Le mot `Street` est ajouté à la fin de la première occurrence de `Address`.<br>Voir aussi « Len » à la page 87. |
| `Stuff("members-list@myweb.com", 0, 0, "cc:"` | `cc:members-list@myweb.com` |

# Substr

Extrait une partie d'une chaîne donnée.

### Syntaxe

`Substr(`*`s1`*`, `*`n1`*`, `*`n2`*` )`

### Paramètres

| Paramètre | Description |
|---|---|
| `s1` | Chaîne source. |
| `n1` | Position, dans la chaîne `s1`, marquant le début de l'extraction.<br>Si `n1` est inférieur à un, la fonction utilise la position du premier caractère. Si `n1` est supérieur à la longueur de `s1`, la fonction utilise la position du dernier caractère. |
| `n2` | Nombre de caractères à extraire.<br>Si `n2` est inférieur ou égal à 0, la fonction renvoie une chaîne vide. Si `n1 + n2` est supérieur à la longueur de `s1`, la fonction renvoie la sous-chaîne à partir de `n1` et à la fin de `s1`. |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Substr` :

| Expression | Renvoie |
|---|---|
| `Substr("ABCDEFG", 3, 4)` | `CDEF` |
| `Substr(3214, 2, 1)` | `2` |
| `Substr(Last_Name[0], 1, 3)` | Les trois premiers caractères de la première occurrence de `Last_Name`. |
| `Substr("ABCDEFG", 5, 0)` | `""` |
| `Substr("21 Waterloo St.", 4, 5)` | `Water` |

# Upper

Convertit en majuscules toutes les minuscules d'une chaîne.

### Syntaxe

`Upper(s [, k ])`

### Paramètres

| Paramètre | Description |
|---|---|
| `s` | Chaîne à convertir. |
| `k` (Facultatif) | Chaîne représentant un paramètre régional valide. Si vous omettez la valeur `k`, la fonction utilise le paramètre régional ambiant.<br>Voir aussi « Paramètres régionaux » à la page 43.<br>**Remarque :** cette fonction convertit uniquement les caractères Unicode U+61 à U+7A (du jeu de caractères ASCII), ainsi que les caractères U+FF41 à U+FF5A (du jeu de caractères pleine largeur). |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Upper` :

| Expression | Renvoie |
|---|---|
| `Upper("abc")` | `ABC` |
| `Upper("21 Main St.")` | `21 MAIN ST.` |
| `Upper(15)` | `15` |
| `Upper(Address[0])` | Cet exemple convertit en majuscules la première occurrence de `Address`. |

# Uuid

Renvoie une chaîne UUID qui sert de méthode d'identification.

### Syntaxe

`Uuid([n ])`

### Paramètres

| Paramètre | Description |
|---|---|
| `n` | Numéro identifiant le format de la chaîne UUID. Numéros valides :<br>• 0 (valeur par défaut) : la chaîne UUID contient uniquement des valeurs hexadécimales.<br>• 1: la chaîne UUID contient des tirets séparant les séquences de valeurs hexadécimales à des positions fixes.<br>Si vous omettez la valeur `n`, la fonction utilise la valeur par défaut. |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Uuid` :

| Expression | Renvoie |
|---|---|
| `Uuid()` | Une valeur telle que `3c3400001037be8996c400a0c9c86dd5` |
| `Uuid(0)` | Une valeur telle que `3c3400001037be8996c400a0c9c86dd5` |
| `Uuid(1)` | Une valeur telle que `1a3ac000-3dde-f352-96c4-00a0c9c86dd5` |
| `Uuid(7)` | Une valeur telle que `1a3ac000-3dde-f352-96c4-00a0c9c86dd5` |

# WordNum

Renvoie le texte équivalent à un nombre donné.

### Syntaxe

`WordNum(`*`n1`* `[, `*`n2`* `[, `*`k`* `]])`

### Paramètres

| Paramètre | Description |
|---|---|
| `n1` | Nombre à convertir.<br>Si l'un des énoncés suivants est vrai, la fonction renvoie des astérisques (*) pour indiquer une erreur :<br>• `n1` n'est pas un nombre.<br>• La valeur entière de `n1` est négative.<br>• La valeur entière de `n1` est supérieure à 922 337 203 685 477 550. |
| `n2` (Facultatif) | Numéro identifiant l'option de formatage. Numéros valides :<br>• 0 (valeur par défaut) : le nombre est converti en texte représentant la partie entière du nombre.<br>• 1: le nombre est converti en texte représentant la valeur monétaire, sans décimales.<br>• 2: le nombre est converti en texte représentant la valeur monétaire, avec les décimales.<br>Si vous omettez la valeur `n2`, la fonction utilise la valeur par défaut (`0`). |
| `k` (Facultatif) | Chaîne représentant un paramètre régional valide. Si vous omettez la valeur `k`, la fonction utilise le paramètre régional ambiant.<br>Voir aussi « Paramètres régionaux » à la page 43.<br>**Remarque :** à compter de la présente version, vous ne pouvez plus indiquer un identificateur de paramètre régional autre que l'anglais pour cette fonction. |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `WordNum`.

| Expression | Renvoie |
|---|---|
| `WordNum(123.45)` | `One Hundred and Twenty-three Dollars` |
| `WordNum(123.45, 1)` | `One Hundred and Twenty-three Dollars` |
| `WordNum(1154.67, 2)` | `One Thousand One Hundred Fifty-four Dollars And Sixty-seven Cents` |
| `WordNum(43, 2)` | `Forty-three Dollars And Zero Cents` |
| `WordNum(Amount[0], 2)` | Cet exemple utilise la première occurrence de `Amount` comme nombre à convertir. |

# 10 Fonctions URL

Ces fonctions portent sur l'envoi et la réception d'informations (type de contenu et codage des données compris) à destination ou en provenance de n'importe quel emplacement URL accessible.

**Fonctions**



## Get

Télécharge le contenu de l'adresse URL indiquée.

**Remarque :** la fonction `Get` ne s'applique qu'aux formulaires certifiés. Adobe Acrobat® et Adobe Reader® ne sont pas capables d'identifier les formulaires certifiés tant que l'événement `initialize` n'a pas été lancé. Pour tirer parti de la fonction `Get` sur les formulaires certifiés avant de procéder au rendu du formulaire, utilisez l'événement `docReady`.

**Syntaxe**

`Get(`*`s`*`)`

**Paramètres**

| Paramètre | Description |
|---|---|
| `s` | Adresse URL à télécharger.<br>Si la fonction ne parvient pas à télécharger le contenu de l'adresse URL, elle renvoie une erreur. |

**Exemples**

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Get`.

| Expression | Renvoie |
|---|---|
| `Get("http://www.myweb.com/data/mydata.xml")` | Données XML extraites du fichier indiqué. |
| `Get("ftp://ftp.gnu.org/gnu/GPL")` | Contenu de la Licence Publique Générale GNU. |
| `Get("http://intranet?sql=SELECT+*+FROM+projects+FOR+XML+AUTO,+ELEMENTS")` | Résultats d'une requête SQL adressée au site Web indiqué. |

# Post

Place à l'adresse URL indiquée les données fournies.

**Remarque :** la fonction `Post` ne s'applique qu'aux formulaires certifiés. Acrobat et Adobe Reader ne sont pas capables d'identifier les formulaires certifiés tant que l'événement `initialize` n'a pas été lancé. Pour tirer parti de la fonction `Post` sur les formulaires certifiés avant de procéder au rendu du formulaire, utilisez l'événement `docReady`.

### Syntaxe

`Post(s1, s2 [, s3 [, s4 [, s5 ]]])`

### Paramètres

| Paramètre | Description |
|---|---|
| `s1` | Adresse URL à laquelle les données sont inscrites. |
| `s2` | Données à publier.<br>Si la fonction ne parvient pas à publier les données, elle renvoie une erreur. |
| `s3` (Facultatif) | Chaîne contenant le type de contenu des données à publier. Types de contenu valides :<br>• application/flux d'octets (valeur par défaut)<br>• texte/html<br>• texte/xml<br>• texte/en clair<br>• texte en plusieurs parties/données de formulaire<br>• application/formulaire x-www codé url<br>• tout autre type MIME valide<br>Si vous ne donnez pas de valeur à `s3`, la fonction affecte au type de contenu la valeur par défaut. L'application doit s'assurer que les données à publier utilisent le format adéquat en fonction du type de contenu indiqué. |
| `s4` (Facultatif) | Chaîne contenant le nom de la page de codes utilisée pour coder les données. Noms de pages de codes valides :<br>• UTF-8 (valeur par défaut)<br>• UTF-16<br>• ISO-8859-1<br>• Tout codage de caractères reconnu par l'IANA (Internet Assigned Numbers Authority)<br>Si vous ne donnez pas de valeur à `s4`, la fonction affecte à la page de codes la valeur par défaut. L'application doit s'assurer que le codage des données à publier correspond à la page de codes indiquée. |
| `s5` (Facultatif) | Chaîne contenant tout en-tête HTTP supplémentaire à inclure à la publication des données.<br>Si vous ne donnez pas de valeur à `s5`, la fonction n'ajoute pas d'en-tête HTTP supplémentaire à la publication.<br>Pour effectuer une publication sur un serveur SOAP, vous devez généralement utiliser un en-tête SOAPAction. |

### Exemples

Les expressions suivantes constituent des exemples d'utilisation de la fonction `Post` :

| Expression | Renvoie |
|---|---|
| `Post("http://tools_build/scripts/jfecho.cgi", "user=joe&passwd=xxxxx&date=27/08/2002", "application/x-www-form-urlencoded")` | Publie des données de connexion codées URL sur le serveur et renvoie l'accusé de réception de ce serveur. |
| `Post("http://www.nanonull.com/TimeService/TimeService.asmx/getLocalTime", "<?xml version='1.0' encoding='UTF-8'?><soap:Envelope><soap:Body><getLocalTime/></soap:Body></soap:Envelope>", "text/xml", "utf-8", "http://www.Nanonull.com/TimeService/getLocalTime")` | Publie une requête SOAP pour l'heure locale sur un serveur et attend une réponse XML. |

# Put

Télécharge les données fournies vers l'adresse URL indiquée.

**Remarque :** la fonction `Put` ne s'applique qu'aux formulaires certifiés. Acrobat et Adobe Reader ne sont pas capables d'identifier les formulaires certifiés tant que l'événement `initialize` n'a pas été lancé. Pour tirer parti de la fonction `Put` sur les formulaires certifiés avant de procéder au rendu du formulaire, utilisez l'événement `docReady`.

### Syntaxe

`Put(`*`s1`*`, `*`s2`*` [, `*`s3`*` ])`

### Paramètres

| Paramètre | Description |
|---|---|
| `s1` | Adresse URL de destination du téléchargement. |
| `s2` | Données à télécharger.<br>Si la fonction ne parvient pas à télécharger les données, elle renvoie une erreur. |
| `s3` (Facultatif) | Chaîne contenant le nom de la page de codes utilisée pour coder les données. Noms de pages de codes valides :<br>• UTF-8 (valeur par défaut)<br>• UTF-16<br>• ISO8859-1<br>• Tout codage de caractères reconnu par l'IANA (Internet Assigned Numbers Authority)<br>Si vous ne donnez pas de valeur à `s3`, la fonction affecte à la page de codes la valeur par défaut. L'application doit s'assurer que le codage des données à télécharger correspond à la page de codes indiquée. |

### Exemples

Les expressions suivantes constituent un exemple d'utilisation de la fonction `Put` :

| Expression | Renvoie |
|---|---|
| `Put("ftp://www.example.com/pub/fubu.xml", "<?xml version='1.0' encoding='UTF-8'?><msg>hello world!</msg>")` | Rien si le serveur FTP a autorisé l'utilisateur à télécharger certaines données XML vers le fichier `pub/fubu.xml` sur le serveur. Sinon, cette fonction renvoie une erreur. |

# Index