



Documentation de BotServer

Multimédia SOLUTIONS (c) <https://www.lug.com>

Version v3.0, February 11, 2016

Table des matières

BotServer	1
Interface	1
Connexion	1
Page d'accueil	1
Création et gestion des robots	3
Création d'un robot de tâches	3
Les différents éléments d'un robot	3
Création et gestion des tâches	7
Création d'un robot	7
Les différents éléments de l'édition des tâches	8
Les tâches	9
Langage de macro	11
Bases du macro langage	12
Syntaxe des fonctions	12
Syntaxe des variables	12
Macro spéciale	13
Affectation de variable	13
Opérations conditionnelles	13
Opérateurs logiques	14
Boucle dans une macro	14
Instruction d'itération de texte	15
Les fonctions	16
Fonctions de base (environnement @)	16
Fonctions sur les chaînes	16
Fonctions de comparaison	21
Fonctions de trace	22
Fonctions de conversion	22
Fonctions mathématiques	24
Manipulations de dates	26
Fonctions Web	28
Fonctions diverses	29
Fonctions de fichiers (environnement @File)	30

BotServer

Le logiciel BotServer permet de créer des robots de tâches. Il permet de configurer un certain nombre de tâches exécutées par l'ordinateur et qui seront lancées en fonction de règles pré-définies dans le robot.

Interface

Connexion

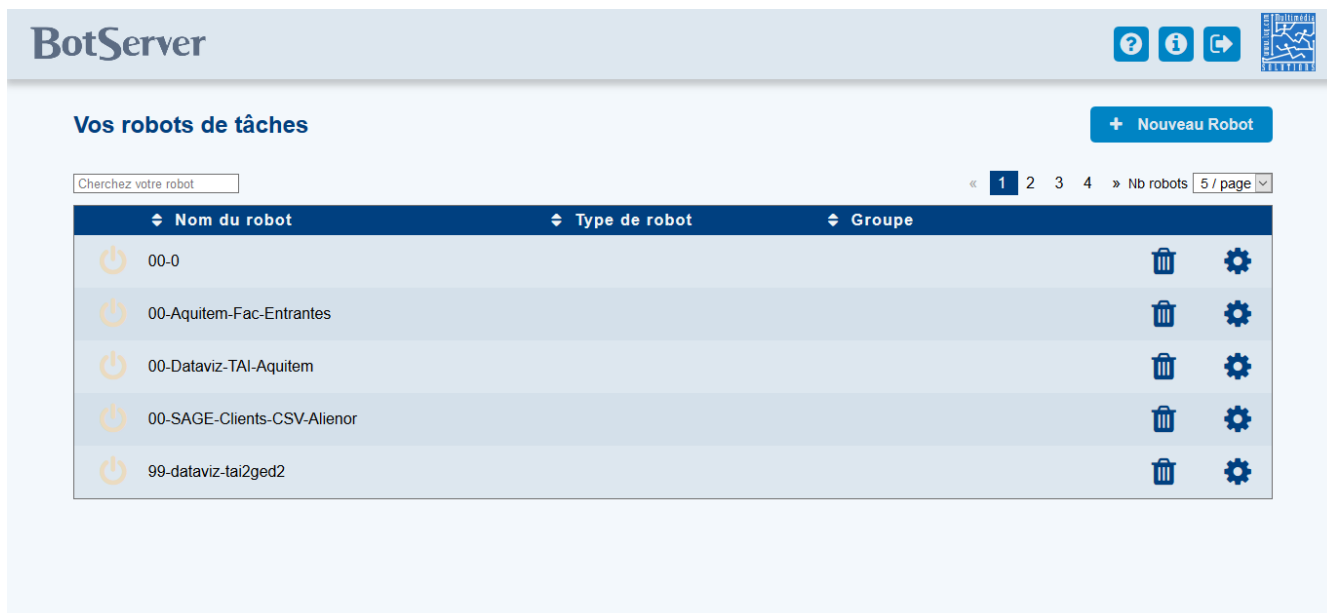
L'accès au logiciel BotServer nécessite de se connecter sur l'adresse [suivante](#). Localhost est le nom du serveur, sauf si vous êtes sur le serveur sur lequel est installé BotServer. Le login par défaut est : "administrator" avec pour mot de passe "administrator" :



Capture d'écran 1. Connexion à BotServer

Page d'accueil

Une fois quelques robots ajoutés, vous obtiendrez la page d'accueil ci-dessous :



Capture d'écran 2. Liste des robots

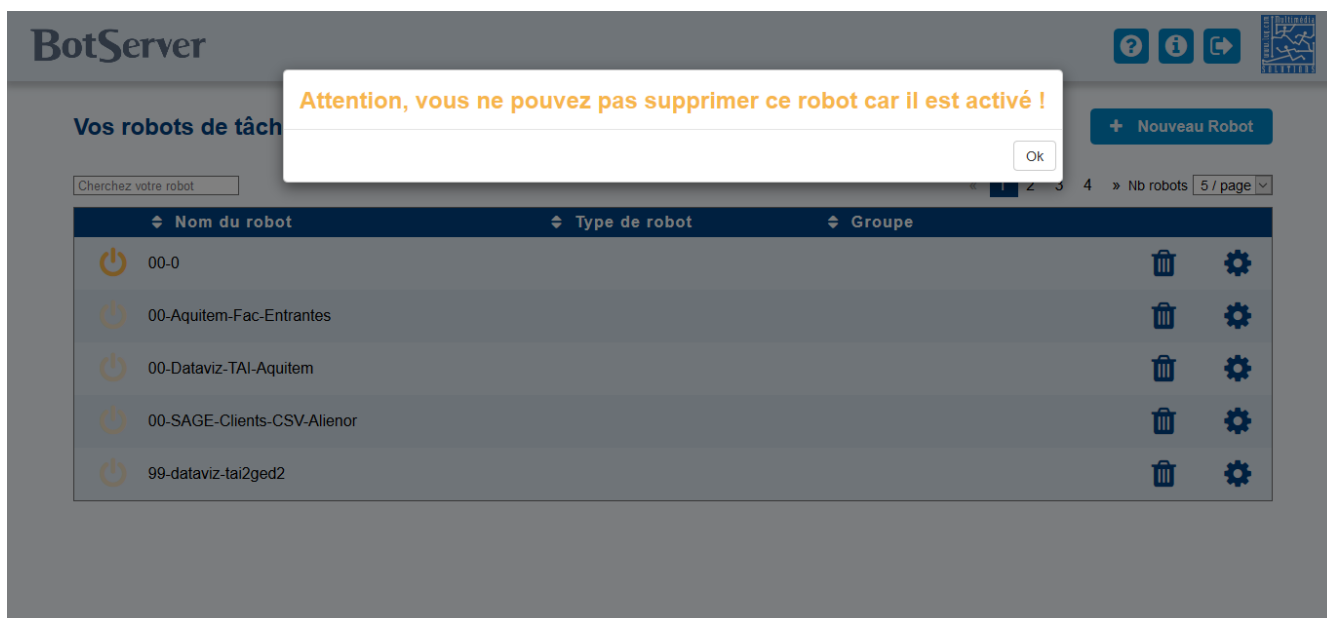
Les dénominations des boutons sont basées sur leurs infobulles (passer la souris dessus) :

- Bouton "A propos" : Il sert à informer sur la société Multimédia Solutions et les droits d'utilisation de Botserver.
- Bouton "Documentation" : Il sert à accéder à la documentation que vous consultez actuellement.
- Bouton "Se déconnecter" : Il sert à se déconnecter de votre compte utilisateur et revenir sur la page de connexion (voir [ici](#) pour se connecter).
- Lien "Lien vers le site de Multimédia Solutions".
- Bouton "Nommer un nouveau robot" : Il sert à créer un nouveau robot en donnant un nom.
- Bouton activer/désactiver un robot : Pour que le robot puisse exécuter ses tâches sur les fichiers concernés, le voyant doit être allumé. Si ce n'est pas le cas, il suffit de cliquer dessus. La solution est la même lorsque vous souhaitez le désactiver.
- Edition d'un robot : Vous pouvez accéder à ses tâches en cliquant sur le robot ou bien accéder aux paramètres du robot en cliquant sur l'engrenage.
- Bouton "Supprimer le robot sélectionné" : Après confirmation, il sert à supprimer un robot avec toutes ses données.

TIP *Nouveauté*
Vous pouvez maintenant gérer la présentation de vos robots (champ de recherche, tri, pagination).

WARNING *Attention!*

- Il faut d'abord désactiver le robot avant de pouvoir le supprimer.
- Le nom du robot doit contenir uniquement des lettres, chiffres et les caractères "\"-_.\""



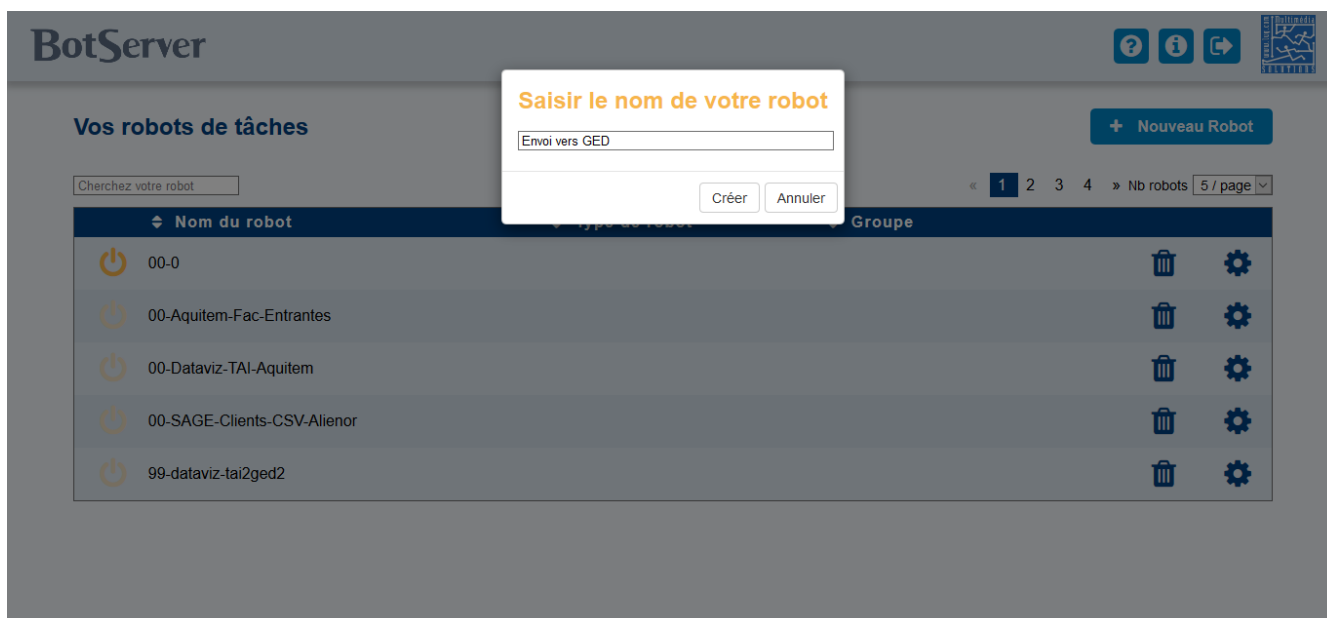
Capture d'écran 3. Tentative de suppression d'un robot activé

Création et gestion des robots

Création d'un robot de tâches

Sur la page d'accueil cliquez sur le bouton "Nouveau Robot".

Saisir le nom de votre robot (*exemple : "Envoi vers GED"*) puis cliquer sur "Créer".



Capture d'écran 4. Création et nommage d'un nouveau robot

Les différents éléments d'un robot

Paramètres

Sauvegarder

Les tâches

Edition du robot

Nom

Envoi vers GED

Type de robot

Surveillance d'un fichier

Période (en ms)

1000

Délai (en ms)

0

Retard (en ms)

0

Dossier de dépôt

\$@Global.workspaceDir\$

Condition

Saisir une condition

Description

Ma description du robot

Nom du groupe de robots

Mon groupe de robots

Nom du fichier log

log

Type de robot : Surveillance d'un fichier

Capture d'écran 5. Edition des paramètres d'un robot

- Nom: Nom du robot.
- Type de robot :
 - "Surveillance d'un fichier" : Le robot attend qu'un fichier existe pour lancer les tâches assignées au préalable.
 - "Surveillance d'un dossier" : Le robot attend qu'un dossier contienne des fichiers pour lancer les tâches assignées au préalable. Choix de défaut.
 - "Surveillance d'un dossier et ses sous-dossiers" : Le robot attend qu'un dossier contienne des fichiers dans ses sous-dossiers ou lui-même pour lancer les tâches assignées au préalable.
 - "Surveillance d'un dossier FTP" : Le robot attend qu'un dossier distant (pas sur le même PC que l'utilisateur) contienne des fichiers pour lancer les tâches assignées au préalable.
- Période (ms) : La fréquence à laquelle le robot vérifie la présence de fichiers dans le ou les dossiers toutes les x millisecondes. Une période de 1000 à 5000ms (5s) est conseillé.
- Délai (ms) : Pour éviter de solliciter le serveur dès son lancement ou un empilement de robots lancés simultanément, ce champ donne un délai d'activation au robot lors du lancement de BotServer.
- Retard (ms) : Pour éviter un conflit entre un utilisateur et un robot cherchant à modifier un même fichier, on impose un retard de date de mise à jour du fichier du côté du robot.
- Dossier de dépôt : Dossier de référence où les fichiers sont supposés être déposés avant d'être traités par les tâches assignées.
- Condition : En plus de la présence de fichiers selon le dossier de dépôt et le type de robot, une condition peut être ajoutée au robot pour lancer ses tâches. En cliquant sur "Saisir une condition", une condition est ensuite complétée (*Par exemple : attendre qu'un dossier de destination soit vide avant de lui envoyer des fichiers*) et si la condition est vraie, alors les tâches sont lancées.
- Description : La description du robot. Il n'y a pas de limites de caractères.
- Nom du groupe de robots : Groupe servant d'étiquette au robot pour catégoriser plusieurs

robots liés à une activité similaire.

- Log : Nom du fichier log événementiel lorsque le robot est démarré.

The screenshot shows the BotServer configuration page for a robot. The interface is divided into several sections:

- Nom:** A text field containing "Envoi vers GED".
- Type de robot:** A dropdown menu set to "Surveillance d'un dossier".
- Période (en ms):** A text field containing "1000".
- Délai (en ms):** A text field containing "0".
- Retard (en ms):** A text field containing "0".
- Dossier de dépôt:** A text field containing "\$@Global.workspaceDir\$".
- Condition:** A link labeled "Saisir une condition".
- Description:** A large text area containing "Ma description du robot".
- Nom du groupe de robots:** A text field containing "Mon groupe de robots".
- Nom du fichier log:** A text field containing "log".
- Type de robot : Surveillance d'un dossier:** A section header.
- Nombre max de fichiers à traiter:** A text field.
- Filtre:** A dropdown menu set to "Format PDF".

Capture d'écran 6. Edition des paramètres d'un robot en surveillant un dossier

- Valable pour la surveillance d'un dossier, d'un dossier et ses sous-dossiers et d'un dossier FTP :
 - Nombre max de fichiers à traiter : Nombre maximum de fichiers consécutifs à traiter. Lorsque ce nombre est atteint, le robot attend la prochaine scrutation. Cela permet d'effectuer des attentes liées à la période. Ce paramètre est utile car certains logiciels plantent lorsqu'on leur envoie trop de fichiers d'affilé.
 - Filtre : Seul les fichiers correspondant au filtre choisi seront traités par les tâches du robot. Le mode "EXPERT" permet d'avoir un filtre personnalisé.

TIP

Nouveauté

Le filtre dispose de modes prédéfinis désormais.

The screenshot shows the BotServer web interface. At the top, there's a header with the 'BotServer' logo and navigation icons. The main content area is titled 'Type de robot' and shows a dropdown menu set to 'Surveillance d'un dossier FTP'. Below this, there are three input fields: 'Période (en ms)' with the value '1000', 'Délai (en ms)' with '0', and 'Retard (en ms)' with '0'. A 'Dossier de dépôt' field contains '\$@Global.workspaceDir\$'. A 'Condition' link is present. To the right, there's a large empty box, a 'Nom du groupe de robots' field with 'Mon groupe de robots', and a 'Nom du fichier log' field with 'log'. A section titled 'Type de robot : Surveillance d'un dossier FTP' contains several fields: 'Serveur', 'N° port', 'Délai fichier non modifié (en ms)', 'Nom d'utilisateur', 'Mot de passe', 'Nombre max de fichiers à traiter', 'Dossier distant', and a 'Filtre' dropdown set to 'Format PNG'.

Capture d'écran 7. Edition des paramètres d'un robot en surveillant un dossier FTP

- Valable pour la surveillance d'un dossier FTP :
 - Serveur : Nom de domaine du serveur à accéder.
 - N° port : Numéro de port à préciser pour échanger avec le serveur. Si le champ est vide, il s'agit du port 21.
 - Nom d'utilisateur : Nom d'utilisateur pour se connecter sur le serveur.
 - Mot de passe : Mot de passe pour se connecter sur le serveur.
 - Dossier distant : Dossier se trouvant dans le serveur avec pour racine du chemin absolu la racine du serveur.

NOTE

Information

- Plusieurs robots peuvent utiliser le même fichier de log.
- En mode EXPERT, le filtre doit être noté sous la forme d'une expression régulière de type Java : Plus d'informations [ici](#).

BotServer

Nom
Envoi vers GED

Type de robot
Surveillance d'un dossier

Période (en ms) 1000 **Délai (en ms)** 0 **Retard (en ms)** 0

Dossier de dépôt
\$@Global.workspaceDir\$

Condition
Saisir une condition

Description
Ma description du robot

Nom du groupe de robots
Mon groupe de robots

Nom du fichier log
log

Type de robot : Surveillance d'un dossier

Nombre max de fichiers à traiter

Filtre Filtre personnalisé
EXPERT

Capture d'écran 8. Edition des paramètres d'un robot en surveillant un dossier avec le mode Expert

Création et gestion des tâches

Création d'un robot

Cliquer sur le bouton "+" ("Créer une tâche" en infobulle) puis choisir le type de tâche affiché dans la boîte de dialogue "Choisissez une tâche".

BotServer

Liste des tâches

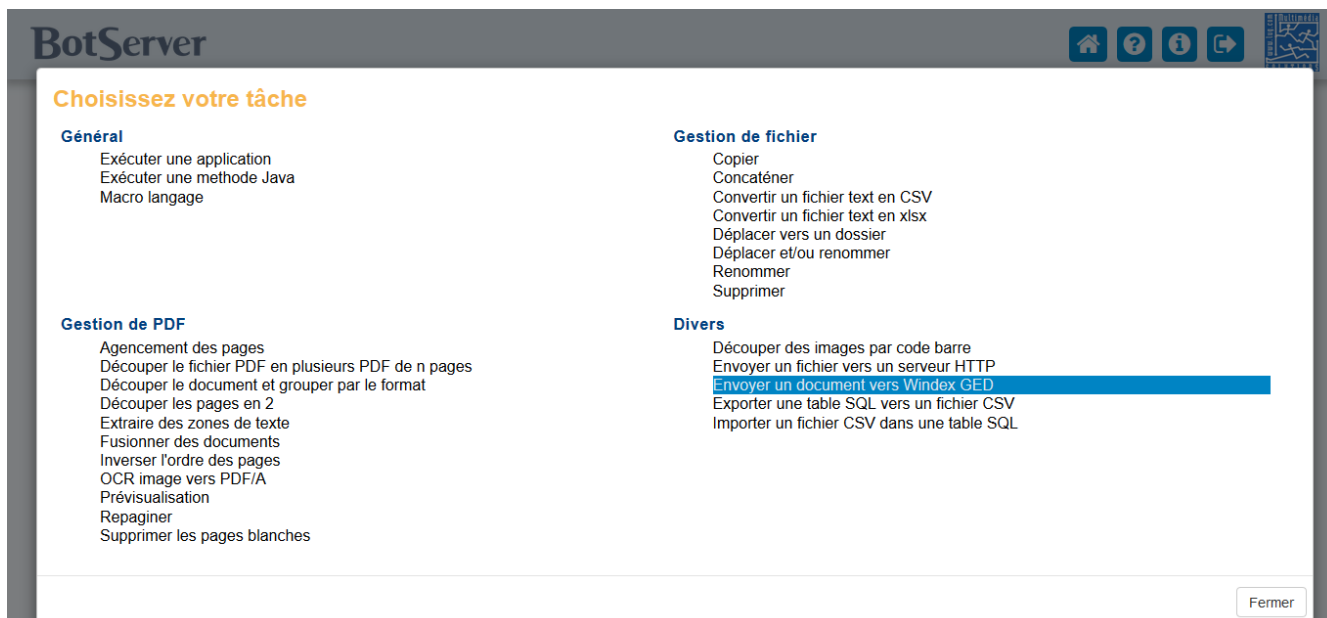
Sauvegarder Paramètres du robot

Tâches

Créer une tâche

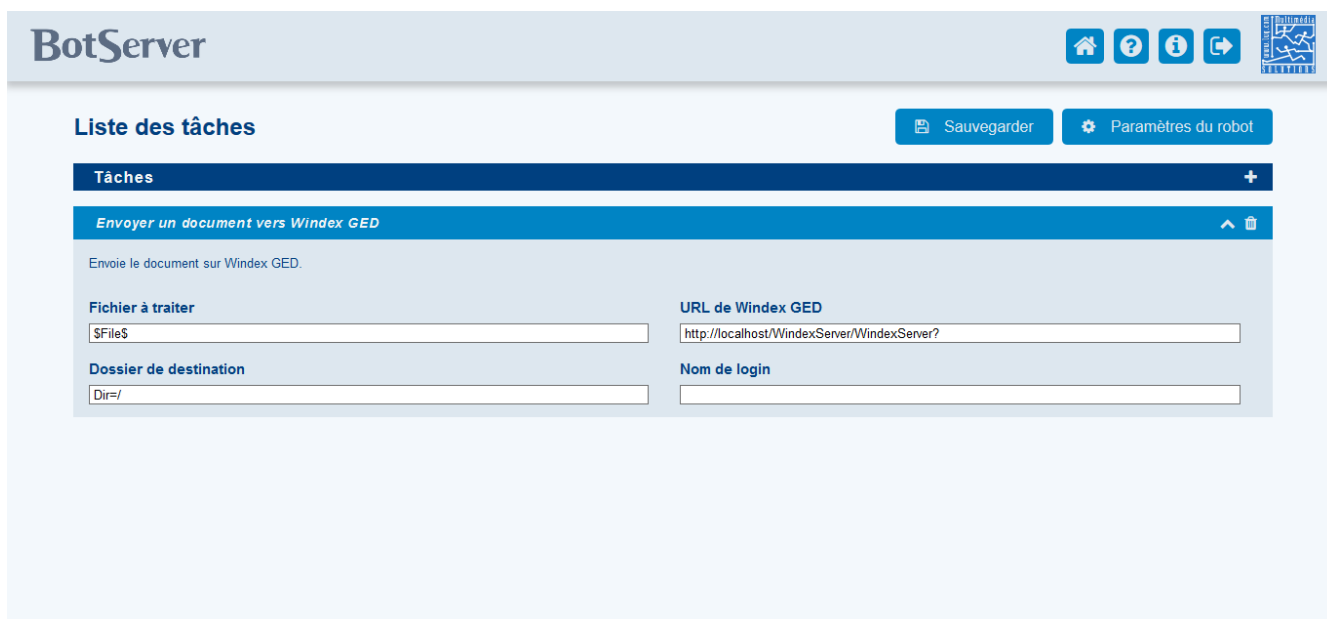
Capture d'écran 9. Ajout d'une tâche : étape 1

Cliquer sur une tâche (ici nous avons pris la tâche suivante "Envoyer un document vers windex GED").



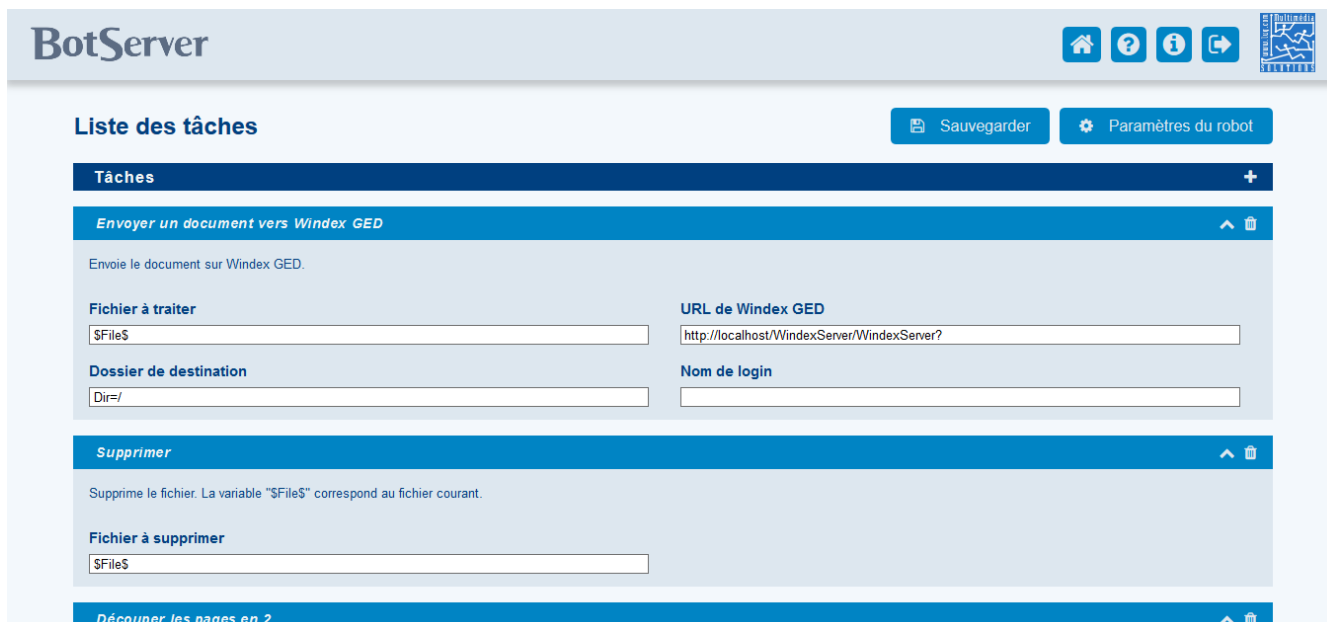
Capture d'écran 10. Ajout d'une tâche : étape 2

Pour chaque tâche, il y a une liste de paramètres à renseigner. Dans chaque champ de paramétrage, vous pouvez utiliser un langage de macro (voir chapitre [Langage de Macro](#)) en mettant les instructions entre deux "\$".



Capture d'écran 11. Ajout d'une tâche : étape 3

Les différents éléments de l'édition des tâches



Capture d'écran 12. Edition des tâches d'un robot

- Bouton "Maison" (Liste des robots en infobulle) : Permet de revenir à la page d'accueil où tous les robots sont répertoriés (*pour en savoir plus, cliquer [ici](#)*).
- Bouton "Sauvegarder" : Permet de sauvegarder toutes les modifications liées au robot.
- Les Tâches/Paramètres du robot : Selon où vous vous situez **dans l'édition d'un robot**, vous pourrez toujours accéder à vos données et vos modifications de données. Pensez à sauvegarder avant de revenir sur la [liste de robots](#).
- Bouton de suppression ([trash]) : Permet de supprimer une tâche. **L'action ne possède pas, contrairement à la suppression d'un robot, une demande de confirmation !**

NOTE	<p><i>Information</i></p> <ul style="list-style-type: none"> • Chaque robot peut accéder au chemin complet du fichier en cours au travers de la macro \$File\$. • Les tâches sont exécutées dans l'ordre d'apparition. Elle peuvent être déplacées par glisser/déplacer en cliquant sur le titre de la tâche.
	<p><i>Nouveauté</i></p> <ul style="list-style-type: none"> • Pour une meilleure lisibilité, il est désormais possible de cacher et déployer une ou plusieurs tâches avec le bouton [chevron down] (à côté de [trash]) . Elles se cacheront à nouveau par défaut si vous quittez l'édition du robot.

Les tâches

Macro

Cette tâche lance une macro (voir [Langage de Macro](#)).

Exécuter une application

Lance automatiquement une application tierce (*logiciel métier par exemple*).

Exécuter une méthode Java

Permet un développement spécifique sur une méthode sous Java.

Copier un fichier

Copie le fichier à traiter vers un dossier de destination.

Déplacer un fichier vers un dossier

Déplace un fichier à traiter vers un dossier de destination.

Déplacer et/ou renommer un fichier

Déplace ou renomme un fichier à traiter vers un dossier de destination.

Renommer un fichier

Renomme un fichier à traiter.

Créer un dossier et sous dossier

Créer tous les dossiers correspondant au chemin indiqué.

Supprimer un fichier

Supprime un fichier à traiter.

Concaténer deux fichiers en un seul

Contrairement à la fusion des fichiers PDF (voir [ici](#)

Convertir un fichier texte en CSV

Convertir un fichier texte en xlsx

Découper le fichier PDF en plusieurs PDF de n pages

Découper en 2 les pages du fichier PDF

Le fichier PDF doit être constitué uniquement d'images. Un nouveau fichier PDF sera généré et comprendra le double de page, ou chaque page est la moitié d'une page du document initial.

Fusionner des fichiers PDF

Cette tâche permet de concaténer ou assembler plusieurs fichiers PDF dans un seul document PDF.

Agencement des pages d'un PDF

Permet de ré-ordonner les pages dans le document PDF.

Découper le PDF suivant pages ses séparations (code barre)

Découpe en plusieurs PDF un document comportant des pages de séparation ayant un code barre (E39) répondant à un critère de la règle regexp.

Prévisualisation du PDF

Cette tâche permet de créer un fichier image correspondant à une vignette de prévisualisation du fichier PDF.

Supprimer les pages blanches d'un PDF

Suppression de pages blanches générées par la numérisation recto/verso, par exemple.

Inverser l'ordre des pages du PDF

Dans le fichier de sortie, la première page devient la dernière, et inversement ...

Repaginer le PDF

Permet de refaire la pagination d'un fichier PDF.

Découper le document PDF

Découpe automatique des fichiers en fonction de règles pré-définies.

OCR image vers PDF/A

Reconnaissance automatique de caractères. Cette tâche convertit une image (jpg, png, gif) ou les images d'un fichier image tiff en un fichier PDF/A. Le texte reconnu par l'OCR est incrusté en arrière plan.

Extrait des zones de texte du PDF

Découpage d'images par code barre

Exemple d'utilisation : le code barre peut servir de séparateur lors de la numérisation d'un lot de documents.

Découper un document image/PDF ayant des page de séparation comportant un séparateur de type code barre

Découpe en plusieurs fichier un document comportant des pages de séparation ayant un code barre (E39) répondant à un critère.

Envoyer un document sur Windex GED

Envoie le fichier dans Windex GED.

Langage de macro

Dans les champs de saisie des tâches vous pouvez appeler les fonctions du langage de macro en entourant votre macro par deux "\$". Ainsi \$File\$ sera remplacé par le chemin complet du fichier.

Langage de macro

Dans les champs de saisie des tâches, vous pouvez appeler les fonctions du langage de macro en entourant votre macro par deux "\$". Ainsi \$File\$ sera remplacé par le chemin complet du fichier

Le macro langage est sensible à la case.

WARNING

Liste des variables mis à disposition pour chaque tâche :

- File : Chemin complet du fichier (dossier + nom). *Exemple* : "C:/scan-data/scaner-1/456456.tif",
- FileName : Nom du fichier. *Exemple* : "456456.tif",
- FilePath : Chemin du dossier du fichier *Exemple* : "C:/scan-data/scaner-1",
- ReturnValue : Contient la valeur retourné par la dernière tâche.

Bases du macro langage

Les expressions du langage de macro sont toujours constituées soit d'un nom ou d'un nom suivi de parenthèses. Il ne supporte pas les expressions de type "a+b" et il ne supporte pas d'opérateur d'affectation (sauf au travers de la macro @Set()) Le macro langage supporte les chaînes de caractères en les entourant de guillemets ou simple quotes.

Exemple :

```
"ma chaîne"  
'ma chaîne'
```

Syntaxe des fonctions

Le nom d'une fonction est précédé d'un arobase. Une fonction exécute un traitement.

Exemple :

```
@ma_fonction()
```

Une fonction peut accepter des paramètres. Dans ce cas, ils sont indiqués après le nom de la fonction, entre parenthèses, séparé par une virgule.

Exemple :

```
@ma_fonction( param1 , param2 )  
@ma_fonction( param1 )
```

Si un paramètre est une constante, il est toujours encadré par des quotes simples, que sa valeur soit numérique ou non. Si une chaîne de caractères contient une quote, il faut la faire précéder d'un anti-slash. @ma_fonction('texte\n bonjour','2','3')

Syntaxe des variables

Une variable représente une valeur numérique ou alphanumérique. Elle est composée de lettres de

chiffres et du caractère "_". Elle commence soit par une lettre soit par "_". Les variables sont sensibles à la casse.

Exemple :

```
ma_variable
```

Macro spéciale

Affectation de variable

@Set(var1 , valeur1 , var2 , valeur2 , [... , varN , valeurN , varN+1 , valeurN+1] , valeur_de_retour)

Affectation à la variable retournée par var1 (resp: var2).

La valeur retournée par valeur1 (resp: valeur2)

Puis retourne la valeur donnée par valeur_de_retour.

@Get(varname)

Retourne la valeur de l'expression **varname**.

Exemple :

```
@Get( @concat( "ma" , "variable" ) )  
@Call( env , macro , param1 [... , paramN] )
```

Dans l'environnement env, on évalue la macro retournée par macro en lui donnant les arguments param1 à paramN.

@EvalMacro(macro)

Exécute la macro passée en arguments.

@EvalMacro(file , charset)

Charge puis évalue la macro se trouvant dans le fichier file de jeux de caractères charset.

Les valeurs communes de charset sont "ISO-8859-1" (Linux), "ISO-8859-1" (Windows), "UTF-8" (universel).

@exit()

Sort de la macro.

Opérations conditionnelles

@true

Correspond à la valeur booléenne vraie.

@false

Correspond à la valeur booléenne faux.

@If(condition1 , expression1 [... , conditionN , expressionN] , default)

Si **condition1** est vrai, **expression1** est évaluée.

Sinon, si **condition2** est vrai, **expression2** est évaluée et ainsi de suite.

Si aucune condition n'est vraie, **default** est évalué.

@IsEmpty(expression1)

Retourne vrai si **expression1** est une chaîne vide ou une valeur Null.

@IsNull(expression1)

Retourne vrai si **expression1** est une chaîne vide ou une valeur Null.

@IsNotEmpty(expression1)

Retourne la valeur faux si **expression1** est une chaîne vide ou une valeur Null.

@IfEmpty(expression1 , defaultValue)

Retourne la valeur **defaultValue** si **expression1** est une chaîne vide ou une valeur Null.

Sinon retourne **expression1**.

Opérateurs logiques

@Not(expression)

Expression doit être de type booléen.

Si **expression** est vrai, retourne faux.

Si **expression** est faux, retourne vrai.

@And(expression1 , expression2)

expression1 et **expression2** doivent être de type booléen.

Effectue un "et / &" logique entre les deux expressions.

Si **expression1** et **expression2** sont vrais, alors vrai est retourné.

Dans tous les autres cas, faux est retourné.

@Or(expression1 , expression2)

expression1 et **expression2** doivent être de type booléen.

Effectue un "ou / ||" logique entre les deux expressions.

Si **expression1** et **expression2** sont faux, alors faux est retourné.

Dans tous les autres cas, vrai est retourné.

Boucle dans une macro

@Do(arg1 [... , argN])

Exécute les macros **arg1** à **argN**.

Puis retourne la valeur de **argN**.

@While(condition , arg1[... , argN])

Tant que la condition **condition** retourne vrai

Exécute les macros **arg1** à **argN**

Puis retourne la valeur de **argN**.

@DoWhile(arg1 [... , argN])

Exécute les macros **arg1** à **argN**, puis tant que la condition condition est vraie réitère l'exécution de **arg1** à **argN**. Puis retourne la dernière valeur évaluée.

@ForEach(varname , array , eval1 [... , evalN])

Affecte à la valeur retournée par varname chaque élément de array, puis évalue les expressions eval1 à evalN

Instruction d'itération de texte

Les macros suivantes permettent d'itérer le texte compris dans leur **espace** **\$@End\$**.

\$@Begin(environnement , debut , max)\$

Les variables **debut** et **max** sont optionnelles. Cette fonction s'utilise dans un environnement contenant des informations sous forme de liste. Cette liste est parcourue à partir de l'enregistrement indiqué par debut (numérotation commençant à 0) jusqu'à l'enregistrement debut + max. ou à concurrence du dernier enregistrement.

Syntaxe à respecter :

```
$@Begin( environnement , debut , max )$  
<!-- texte à générer -->  
$@End$
```

Le code compris entre **\$@Begin([...])\$** et **\$@End\$** est répété pour chaque enregistrement dans la page HTML retournée.

À noter également que environnement est considéré comme l'environnement par défaut : il n'est donc pas nécessaire de le préciser lorsqu'on utilise des fonctions ou variables de cet environnement, sauf pour lever une ambiguïté.

Cet exemple :

```
$@Begin( @mon_environnement , 5 , 10 )$  
Il pleut,<br>  
$@End$
```

Génère le texte suivant :

```
Il pleut,<br>  
Il pleut,<br>  
Il pleut,<br>  
Il pleut,<br>  
Il pleut,<br>  
Il pleut,<br>
```

\$@BeginWhile(environnement , condition)\$ texte \$@End\$

Tant que la condition condition est vraie, itère le texte text.

`$$@BeginCache(varName)$ texte $$@End$`

Affecte à la variable retournée par varName le texte texte.

`$$@BeginIf(env , condition1)$ texte1 $$@ElseIf(condition2)$ texte2 $$@Else$ texte3 $$@End$`

Affiche le texte text1 si la condition condition1 est vraie, sinon affiche le texte texte2 si la condition condition2 est vraie, sinon affiche le texte texte3. Les clauses @ElseIf et @Else sont optionnelles.

Cet exemple :

```
$$@BeginIf( environnement , condition1 )$  
<!-- texte à afficher -->  
$$@ElseIf( condition2 )$  
<!-- texte à afficher -->  
$$@Else$  
<!-- texte à afficher par défaut -->  
$$@End$
```

Information

- Bien entendu, vous pouvez mettre autant de @ElseIf que nécessaire et un seul \$\$@Else\$.
- **environnement** est considéré comme l'environnement par défaut : il n'est donc pas nécessaire de le préciser lorsqu'on utilise des fonctions ou variables de cet environnement **env**, sauf pour lever une ambiguïté.
- Vous pouvez omettre l'environnement **env** en écrivant **`$$@Begin(condition1)$`**.

NOTE

`$$@BeginIfNot(environnement , condition)$`

Idem à \$\$@BeginIf()\$ mais lorsque la condition est fausse.

`$$@BeginElseIf()$`

Est un équivalent de \$\$@ElseIf()\$

`$$@BeginElseIfNot()$`

Est un équivalent de \$\$@ElseIfNot()\$

Les fonctions

Fonctions de base (environnement @)

Fonctions sur les chaînes

Ce qui est entre crochets est optionnel.

`@Length(chaîne)`

Cette fonction retourne alors la longueur de la chaîne **chaîne**.

@Left(chaîne , nbCaracteres)

nbCaracteres doit être une expression de type numérique. Retourne les nbCaracteres caractère de la chaîne **chaîne** à partir de sa gauche.

@Right(chaîne , nbCaracteres)

nbCaracteres doit être une expression de type numérique. Retourne les nbCaracteres caractère de la chaîne **chaîne** à partir de sa droite.

@Mid(chaîne , debut [, fin])

Début et fin doivent être des expressions de type numérique. Cette fonction retourne la portion de la chaîne **chaîne** depuis le caractère indiqué par debut (la numérotation commence à 0) jusqu'au caractère indiqué par fin (exclus). Si fin est omis, la portion de texte commence à debut jusqu'à la fin de la chaîne.

@Trim(chaîne)

Retourne la chaîne **chaîne** après lui avoir supprimé les espaces de droite et gauche.

@Replace(chaîne1 , chaîne2 , chaîne3)

Remplace la chaîne **chaîne2** par la chaîne **chaîne3** dans la chaîne **chaîne1**.

@ToLowercase(expression)

Retourne la chaîne expression convertit en minuscules.

@ToUppercase(expression)

Retourne la chaîne expression convertit en majuscules.

@Capitalize(expression)

Retourne la chaîne expression capitalisée : 1 lettre en majuscule, le reste des lettres est inchangé.

@Contain(val , arg1 [... , argK])

Retourne la première valeur de **arg1** à **argK** qui contient la valeur **val**.

@GenerateRandomString(taille)

Retourne une chaîne tirée au hasard de taille **taille**. Si la **taille** est omise, retourne une chaîne de taille 8.

@Concat(expression1 [, expressionN ...])

Concatène les différentes chaînes **expression1** à **expressionN**.

@Concat2(séparateur , expression1 [... , expressionN])

Concatène les différentes chaînes **expression1** à **expressionN** en les séparant par **séparateur**. Les chaînes vides ou nulles sont ignorées.

@Transpose(expression1 , expression2 [, expression3])

Retourne une chaîne après avoir remplacé dans la chaîne **expression1**, chaque caractère de la chaîne **expression2** par le caractère correspondant de la chaîne **expression3**. S'il n'y a pas de correspondance, le caractère est supprimé. Si **expression3** n'est pas renseigné, tous les caractères de **expression2** trouvés dans **expression1** sont supprimés. S'il y a plus de caractères dans

expression3 que dans **expression2**, les caractères de **expression3** situés au-delà de la longueur de **expression2** sont ignorés. la fonction est sensible à la casse.

Table 1. Exemples d'utilisation de @Transpose

Expression	Résultat
@Transpose("abcdef", "b", "Z")	aZcdef
@Transpose("abcdef", "bdf", "ZZZ")	aZcZeZ
@Transpose("abcdef", "bdf")	ace
@Transpose("abcdef", "bdf", "ZZ")	aZcZe

@Split(val, regexp [, nb])

Retourne le tableau résultant du découpage de la chaîne val en utilisant l'expression régulière regexp comme séparateur.

Affiche le texte text1 si la condition condition1 est vraie, sinon affiche le texte texte2 si la condition condition2 est vraie, sinon affiche le texte texte3. Les clauses @ElseIf et @Else sont optionnelles.

Exemple 1 :

```
$@Split( "toto,titi;tutu" , "[,;]" )$
```

Génère le résultat suivant :

```
["toto","titi","tutu"]
```

L'exemple 2 utilise le nombre nb qui indique la taille maximale du tableau à retourner.

Exemple 2 :

```
$@Split( "toto,titi;tutu" , "[,;]" , '1' )$
```

Génère le résultat suivant :

```
["toto"],["titi","tutu"]
```

En combinaison avec @Implode() on peut éliminer certaines parties d'une chaîne.

Exemple 3 :

```
@Implode(  
  @Split("Info:mon info;ceci est le résumé qui utilise aussi le ';' dans son texte !",  
    ";", 2) ,  
  " " , -1)
```

Génère le résultat suivant :

```
"ceci est le résumé qui utilise aussi le ';' dans son texte !"
```

@ReplaceKey(expression)

Remplace les variables se trouvant entre-deux "%" dans la chaîne expression. Les variables sont prises dans l'environnement actuel. S'il y a deux "%" consécutif ("%") alors il est remplacé par un seul "%". La variable peut commencer par une directive de conversion :

- (html) : convertit en HTML la valeur de la variable
- (html:br) : convertit en HTML la valeur de la variable, en remplaçant les retours chariot par "
"
- (url) : convertit au format URL une variable (urlEncode). Par défaut, utilise le jeu de caractères UTF-8
- (url:utf8) : convertit au format URL une variable (urlEncode). En utilisant le jeu de caractère se trouvant après les deux points si présents. sinon utf8
- (#url) : décode la variable "urlEncode" en chaîne. C'est le contraire d'URL
- (js) : convertit en chaîne JavaScript la valeur de la variable
- (jshtml) : convertit en chaîne JavaScript et HTML la valeur de la variable
- (sql) : convertit en chaîne SQL la valeur de la variable
- (sqllike) : convertit en chaîne SQL et like la valeur de la variable

Exemple avec **va**leur valant "L'espadaon" :

```
@ReplaceKey("ma %(html)valeur%)
```

Génère le résultat suivant :

```
"ma L&quote;espadaon"
```

@Filter(val , regexp)

Si val est un tableau, retourne un tableau contenant chaque valeur du tableau dont les chaînes contiennent la forme exprimée par l'expression régulière regexp .

Sinon, si val contient la forme exprimée par l'expression régulière regexp, elle est retournée.

@Substring(chaine1 [, regexp] [, numGroup] [, devaultValue])

Extrait de la chaîne **chaine1**, la chaîne donnée par le numéro de groupe numGroup de l'expression régulière regexp.

Si aucune sous-chaîne n'est trouvée alors la chaîne devaultValue est retournée (par défaut la chaîne vide)

@SelectInList(expression , liste , défaut [, séparateur])

La chaîne **expression** est retournée si elle existe dans la liste **liste**.

Sinon c'est défaut qui est renvoyé. La variable liste est une chaîne représentant une liste de

chaînes, séparées par un caractère (par défaut, la virgule ou le point-virgule). D'autres séparateurs peuvent être indiqués via le paramètre séparateur.

@ReplaceRegex(chaîne1 , regexp , chaîneDeRemplacement)

Remplace dans **chaîne** les sous-chaînes répondant à l'expression régulière **regexp** par la valeur de **chaîneDeRemplacement**

@IndexOf(val , expression)

Retourne l'index de la chaîne **val** dans la chaîne **expression**
Sinon retourne faux.

@LastIndexOf(val , expression)

Retourne le dernier index de la chaîne **val** dans la chaîne **expression**
Sinon retourne faux.

@Implode(tableau , 'separator' [, nbElement])

Retourne la concaténation du tableau de valeur **tableau** en utilisant comme séparateur la chaîne 'separator' . Si **nbElement** est supérieure à zéro cette fonction ne concatène que les **nbElement** première valeur du tableau 'tableau'. Si **nbElement** est négatif, la concaténation ne s'effectue que sur les **nbElement** dernière valeur du tableau **tableau**.

@RepeatString(chaîne , nb)

Retourne une chaîne contenant **nb** fois la chaîne **chaîne** ou une chaîne vide.

@FilterExtract(val , regexp [, iExtract])

Si **val** est un tableau, il retourne un tableau avec les éléments du tableau original qui correspond à l'expression régulière **regexp**.

Sinon il convertit **val** en une chaîne, puis retourne la valeur correspondant à l'expression régulière **regexp**.

Si **iExtract** vaut zéro (défaut), il retourne toute la chaîne.

Sinon il retourne la partie correspondant au groupe **iExtract** de l'expression régulière.

@ElementAt(data , iPosition , défaut)

Retourne l'élément se trouvant à la position **iPosition** de l'objet **data**.

Si 'data' est un tableau retourne l'élément **iPosition**. +
Si 'data' n'est pas un tableau, il est convertit en chaîne puis retourne le caractère se trouvant à la position '**iPosition**'. +
Si la position **iPosition** est hors contexte, retourne défaut si existe sinon faux.

@SelectValueInMap(key , map [, défaut] [, separator] [, affectation] [, casseSensitif])

Retourne la valeur dans la carte **map** correspondant à chaîne **key**.

La chaîne **map** doit avoir un format particulier de la forme "clé1=valeur;clé2=valeur2". Le paramètre **defaut** est la valeur à retourner si la clé n'est pas trouvée. Le paramètre **separator** est le symbole de séparation de couple (clé,valeur). Par défaut c'est le ";". Le paramètre **affectation** est le symbole séparant la clé de sa valeur. Par défaut c'est le "=" Le paramètre **casseSensitif** indique si la recherche de clé est sensible à la casse

Exemples :

```
@SelectValueInMap( chaîneArechercher , listeElement , "val de default" , separator=";"
, affectation="=" , isCaseSensitif)
@SelectValueInMap( "toto" , "A=1;toto=2" , "default" , "," , "=" , isCaseSensitif )
@SelectValueInMap( "toto" , "A=1;toto=2" )
```

Fonctions de comparaison

@Compare(val1 , val2 [, caseInsensitive])

Compare les deux chaînes **val1** et **val2**. Retourne :

- 0, si **val1** et **val2** sont égaux.
- -1, si **val1** est plus petit que **val2**.
- +1, si **val1** est plus grand que **val2**. La comparaison est sensible à la casse si **caseInsensitive** est vrai.

Par défaut la comparaison est sensible à la casse.

@Equal(nombre1 , nombre2)

Retourne vrai si le nombre **nombre1** est égal au nombre **nombre2**.

Sinon faux. Cette fonction convertit ces arguments en nombre avant comparaison.

@Contain(val , expression1 [... , expressionN])

Retourne la chaîne de **expression** à **expressionN** qui contient **val**.

Sinon faux.

@Cmp(val1 , val2)

Retourne vrai si les deux chaînes **val1** et **val2** sont identiques ou nulles.

@IsInString(val , expression)

Retourne vrai si la valeur de **val** est dans la chaîne **expression**.

Sinon faux.

@IsEmpty(expression)

Retourne vrai si la valeur de **expression** est vide, ou nulle.

Sinon faux.

@HasEmpty(expression)

Idem que **@IsEmpty**.

@IsNotEmpty(expression)

Retourne vrai si la valeur de **expression** est vide.

Sinon faux.

@IsTrue(expression)

Retourne vrai si la valeur de **expression** est vraie.

Sinon faux.

@IsFalse(expression)

Retourne vrai si la valeur de **expression** est faux.
Sinon faux.

@StartWith(val , begin)

Retourne vrai si la valeur de **val** commence par **begin**.
Sinon faux.

@EndWith(val , end)

Retourne vrai si la valeur de **val** commence par **end**.
Sinon faux.

@Match(val , regexp)

Retourne vrai, si **val** valide l'expression régulière **regexp**.
Sinon retourne faux.

@IsInList(expression , liste [, séparateurs])

Retourne vrai si la chaîne de **expression** est présente dans la liste **liste**.
liste étant une suite de chaînes de caractères, séparées par le caractère **séparateurs** (par défaut la virgule ou le point-virgule, si non présent).

Exemple :

```
@IsInList( "doc" , "doc;HTML;htm;txt" )
```

Génère le résultat suivant :

```
true
```

Fonctions de trace

@Print(expression1, [... , expressionN])

Affiche sur la sortie standard les valeurs **expression1** à **expressionN**.

@Println(expression1, [... , expressionN])

Affiche sur la sortie standard les valeurs **expression1** à **expressionN**, suivies d'une fin de ligne.

@Debug(expression1, [, expressionN ...])

Affiche sur la sortie standard les valeurs **expression1** à **expressionN**, suivies d'une fin de ligne.

Fonctions de conversion

@ToString(expression)

Convertit la valeur de **expression** en une chaîne.

@ToFloat(expression)

Convertit la valeur de **expression** en un nombre de type "float".

@ToDouble(expression)

Convertit la valeur de **expression** en un nombre de type "double".

@ToInteger(expression)

Convertit la valeur de **expression** en un nombre entier de type "integer" (32bit).

@ToLong(expression)

Convertit la valeur de **expression** en un nombre de type "long" (64 bit).

@ToDate(expression)

Convertit la valeur de **expression** en une date.

Si **expression** est de type nombre, il est considéré comme étant le nombre de milliseconde depuis le 1/1/1970 (epoc).

Si **expression** est de type chaîne, c'est une date de format "yyyy/MM/dd" ou "yyyy-MM-dd" ou "dd/MM/yyyy" de type "dd-MM-yyyy".

@ToEnvironnement(expression) OU @ToEnv(expression)

Convertit la valeur de **expression** en un environnement. Cela permet par la suite de l'itérer dans **\$@Begin()**. Suivant le type de **expression** :

- Cas 1 : si un seul argument qui n'est pas un tableau il est convertit en SimpleEnvironnement, ce qui permet d'accéder à toutes ces méthodes publiques.
- Cas 2 : si le 2ème paramètre contient "field", "getter" ou "map", le nom des champs sur les variables publiques, méthode getter ou sur les noms "F0" à "fxx".
- Cas 3 : si le paramètre 0 contient "header", alors un paramètre sur 2 qui suit contient le nom de la colonne et 1 sur 2 contient le tableau de la colonne. Sinon le nom de chaque colonne est F0 à Fk.

Exemples :

```
<!-- Cas 1 -->
$@Begin( @ToEnvironnement( "," ) )$ $F0$ $@End$
$@Begin( @ToEnvironnement( "," , @true ) )$ $F0$ $@End$
$@Begin( @ToEnvironnement( @Split( "v1,v2,v3" , "," ) ) )$ $F0$ $@End$
<!-- Cas 3 -->
$@Begin( @ToEnvironnement( "Header" , "Champ1" , @Split( "v1,v2,v3" , "," ) , "Champ2"
, @Split( "w1,w2,w3" , "," ) ) )$ $Champ1$-$Champ2$ $@End$
```

@to(type , expression)

Convertit la valeur de **expression** dans le type indiqué par type.

Les différentes valeurs de **type** sont : date, float, double, integer, long, environnement, string.

@Format(chaîne , format [, default] [, local] [, nbChar] [, align] [, charEmpty])

Retourne la chaîne **chaîne** formatée avec le format **format** suivant la local **local** (ex FR, FR/FR).

La chaîne obtenue doit faire **nbChar** caractères si présent.

Elle est alignée à droite si **align** vaut "right", à gauche si **align** vaut "left".

charEmpty est le caractère servant à compléter la chaîne de sortie.

Pour le format de format voir dans les dates @FormatDate et les nombres @FormatNumber

Fonctions mathématiques

@Sum(val1 [, valN ...])

Retourne la somme des valeurs de **val1** à valN Si une des valeurs est une chaîne, elle est convertit en nombre avant de faire la somme. Si la conversion est impossible, elle est ignorée.

@Plus(val1 [, valN ...])

Retourne la somme des valeurs de **val1** à valN

@Add(val1 [, valN ...])

Retourne la somme des valeurs de **val1** à valN

@Minus(val1 [, valN ...])

Retourne la soustraction de la valeur **val1** moins valN

@Mult(nombre1 , nombre2)

Retourne le résultat de la multiplication du nombre **nombre1** par le nombre **nombre2**

@Divide(nombre1 , nombre2)

Retourne le résultat de la division du nombre **nombre1** par le nombre **nombre2**

@Div(nombre1 , nombre2)

Retourne le reste de la division de **nombre1** par **nombre2**.

@Modulo(number , modulo)

Retourne-le modulo modulo du nombre number.

@Dec(expression)

Retourne la valeur de **expression** décrétementée de 1.

@Inc(expression)

Retourne la valeur de **expression** incrémentée de 1.

@neg(val1)

Retourne le négatif de **val1**.

@Superior(val1 , val2)

Retourne vrai si le nombre **val1** est supérieur au nombre **val2**

@SuperiorEqual(val1 , val2)

Retourne vrai si le nombre **val1** est supérieur ou égal au nombre **val2**

@Superior0(val1 , val2)

Retourne vrai si le nombre **val1** est supérieur ou égal au nombre **val2**

@Inferior(val1 , val2)

Retourne vrai si le nombre **val1** est inférieur au nombre **val2**

@InferiorEqual(val1 , val2)

Retourne vrai si le nombre **val1** est inférieur ou égal au nombre **val2**

@Inferior0(val1 , val2)

Retourne vrai si le nombre **val1** est inférieur ou égal au nombre **val2**

@max(val1 , val2)

Retourne le plus grand des nombres **val1** à **val2**

@min(val1 , val2)

Retourne le plus petit des nombres **val1** à **val2**

@abs(val)

Retourne la valeur absolue du nombre **val**

@Ceil(val)

Retourne le nombre entier directement supérieur ou égal au nombre **val**

@Floor(val)

Retourne le nombre entier directement inférieur ou égal au nombre **val**

@Round(val)

Retourne le nombre entier directement supérieur ou égal du nombre **val**

@formatNumber(number , format)

Convertit le nombre **number** en une chaîne correspondant au format **format**.

Table 2. Exemples d'utilisation de @formatNumber

Expression	Format de sortie	Résultat
123.10	##	123.1
123.10	#.00	123.10
123456789	#,###,###	123.456.789
0.12	#%	12%

Il est possible de prendre en compte les caractéristiques locales à un pays et/ou une langue en renseignant les paramètres **localisationEntrée** et **localisationSortie**. Leurs valeurs peuvent influencer le séparateur décimal ou de milliers par exemple. Les deux paramètres ont la même syntaxe : "langue/PAYS"

- **localisationEntrée** : indique les règles du pays et/ou de la langue dans lesquelles la valeur numérique est exprimée.
- **localisationSortie** : indique les règles du pays et/ou de la langue dans lesquelles la valeur numérique sera formatée. Nous n'indiquerons pas ici la liste de toutes les langues et pays (plusieurs centaines de lignes). Sachez que pour le format français, le code est fr/FR et que pour le format anglais, le code est en/EN.

Table 3. Exemples d'utilisation de @formatNumber en fonction du modèle Anglais/Français

Expression	Format de sortie	Localisation Entrée	Localisation Sortie	Résultat
123.10	#. #	en/EN	fr/FR	123,1
123.10	#. #	en/EN	en/EN	123.1
123456789	#,###,###	en/EN	fr/FR	123 456 789
0.12	#,###,###	en/EN	en/EN	123,456,789

Manipulations de dates

@Date

Retourne la date actuelle. La date peut être formatée par la fonction @FormatDate.

@Date(temps_en_ms)

Retourne la date **temps_en_ms** exprimée en millisecondes depuis le 1er janvier 1970.

@Date(année , mois , jour)

Retourne la date correspondant à jour/mois/année.

@Date(année , mois , jour , heure , minute , seconde)

Retourne la date correspondant à jour/mois/année heure:minute:seconde.

@Dateadd(date , année , mois , jour , heure , minute , seconde, milliseconde)

Retourne la date résultat de l'addition à la date **date** et le temps donné par les valeurs positives ou négatives **année, mois, jour, heure, minute, seconde, milliseconde**.

@DateAge(date [, default])

Retourne le nombre d'années entre la date **date** et maintenant. Retourne default ou faux si **date** n'est pas une date.

@DateField(type , date)

La valeur **date** doit être une date. Suivant la valeur de type parmi les chaînes suivantes :

- "YEAR" : retourne l'année
- "MONTH" : retourne le numéro de mois : 1 à 12
- "DATE" : retourne le jour de la date 1 à 31
- "HOUR" : retourne l'heure basée sur 24
- "MINUTE" : retourne les minutes
- "SECOND" : retourne les secondes
- "MILLISECOND" : retourne les millisecondes
- "WEEK_OF_MONTH" : retourne la semaine du mois
- "DAY_OF_YEAR" : retourne le jour de l'année
- "WEEK_OF_YEAR" : retourne la semaine de l'année

- "DAY_OF_WEEK" : retourne le jour de la semaine basé sur dimanche pour 1er jour
- "DAY_OF_WEEK_IN_MONTH" : retourne 1 pour 1er mardi du mois, 2 pour 2e ... Sinon retourne FAUX.

@FormatDate(date , format)

Convertit la date **date** en une chaîne correspondant au format **format**

@FormatDate(date , formatIn , formatOut)

Convertit la chaîne **date** en une date (en utilisant le format **formatIn**) puis la convertit en une chaîne correspondant au format **format**. Si **date** est déjà une date, elle est utilisée tel quel sans passer par le formatage de **formatIn**.

Table 4. Exemples d'utilisation de @formatDate

Symbole	Signification	Type	Exemple
G	Ere	Texte	JC
y	Année	Nombre	2019
M	Mois dans l'année	Texte et nombre	Juillet et 07
d	Jour dans le mois	Nombre	10
h	Heure (de 1 à 12)	Nombre	12
H	Heure (de 0 à 23)	Nombre	0
k	Heure (de 1 à 24)	Nombre	24
K	Heure (de 0 à 11)	Nombre	0
m	Minute	Nombre	30
s	Seconde	Nombre	55
S	Milliseconde	Nombre	978
E	Jour de la semaine	Texte	mardi
D	Jour dans l'année	Nombre	189
F	Jour de la semaine dans le mois	Nombre	2 (2ème mercredi de juillet)
w	Semaine dans l'année	Nombre	27
W	Semaine dans le mois	Nombre	2
a	am/pm (afficher le format heure)	Texte	PM
z	Fuseau horaire	Texte	Pacific Standard Time
'	Caractère d'échappement	Séparateur	
"	Quote simple	Littéral	'

Le nombre de fois où le symbole est répété détermine le format final :

- Pour les symboles de type "Texte"
 - Si le symbole est indiqué au plus 3 fois, c'est la forme abrégée qui est affichée si elle existe.

Exemple : "E" affichera "mar."" pour mardi

- Si le symbole est indiqué plus de 3 fois, c'est la forme complète qui est affichée. Exemple : "EEEE" affichera "mardi"
- Pour les symboles de type "Nombre" Indique le nombre minimum de chiffres. Par exemple pour le dixième jour du mois :
 - "d" affichera "10"
 - "dd" affichera "01" Cependant, pour les années, si on indique yy, seuls les deux derniers chiffres de l'année seront affichés. Par exemple pour l'année 2002 :
 - "yy" affichera "02"
 - "yyyy" affichera "2002"
- Pour les symboles de type "Texte et Nombre"
 - Si le symbole est répété 3 fois et plus, c'est la règle des symboles de type "Texte" qui s'applique
 - Si le symbole est répété moins de 3 fois, c'est la règle des symboles de type "Nombre" qui s'applique Par exemple pour le mois de février, "MM" affichera "02"; "MMM" affichera "fév." ** "MMMM" affichera "février"

Fonctions Web

@UrlEncode(expression)

Convertit la chaîne **expression** en une chaîne de type URL pour utilisation dans les paramètres d'une URI.

@UrlDecode(expression)

Décode un paramètre d'URI en une chaîne. Contraire de @UrlEncode.

@HtmlEncode(expression)

Convertit la chaîne **expression** en une chaîne HTML.

@HtmlDecode(expression)

Convertit la chaîne HTML **expression** en une chaîne. Contraire de @HtmlEncode.

@JSHtmlEncode(expression)

Convertit la chaîne **expression** en une chaîne de type JavaScript dans du HTML.

@JSEncode(expression)

Convertit la chaîne **expression** en une chaîne JavaScript.

@ReadUrlToString(url , charset)

Retourne la chaîne correspondant au fichier texte se trouvant sur l'URL url. Le flux de l'URL est lu avec le jeu de caractères charset.

@URLTo(expression , type)

Extrait de l'URL **expression** une ou toute la chaîne représentant l'URL suivant le type type. Les différentes valeurs de type (exemple d'URL :

"http://monserver:8087/WindexGed/name?Param1=val&Param2=val2#signe")

- file : retourne la partie fichier de l'URL. Exemple :
"/WindexGed/name?Param1=val&Param2=val2"
- path : retourne la partie fichier de l'URL. Exemple : "/WindexGed/name"
- query : retourne la partie fichier de l'URL. Exemple : "Param1=val&Param2=val2"
- rel : retourne la partie fichier de l'URL. Exemple : "signe"
- port : retourne la partie port de l'URL. Exemple : "8087"
- host : retourne la partie serveur de l'URL. Exemple : "monserver"
- protocol : retourne la partie fichier de l'URL. Exemple : "http"
- la chaîne représentant l'URL
"http://monserver:8087/WindexGed/name?Param1=val&Param2=val2#signe"

@FileToURL(surl)

Retourne l'objet de type URL que représente la chaîne surl.

@HtmlEncode(expression [, séparateur])

Encode **expression**. En HTML, certains caractères sont traduits par une séquence d'un ou plusieurs caractères afin qu'ils n'entrent pas en conflit avec la syntaxe même du langage.

Ainsi, si vous voulez afficher le caractère ">", vous devrez écrire en HTML
ou bien

La fonction **@HtmlEncode** effectue cette conversion sur le contenu de l'expression qu'elle retourne. Elle est très utile lorsque vous affichez le contenu d'une variable dans la page HTML et que vous ne contrôlez pas forcément le contenu de cette variable.

Si **expression** contient le caractère de saut de ligne ("n"), il peut être remplacé par la valeur du séparateur **séparateur**. Ainsi, dans un document HTML, le saut à la ligne n'étant pas interprété, on peut par exemple le remplacer par la balise.

@HtmlDecode(expression)

Décode la chaîne **expression**.

Cette fonction est l'opposée de **@HtmlEncode**.

Fonctions diverses

@AddInList([list [, exp1 [... , expK]]])

Si aucun élément ne retourne une liste (Java ArrayList) vide.

Si **list** est une liste (au sens Collection de Java), alors les éléments de **exp1** à **expK** y seront ajoutés.

Si **list** est nul ou n'est pas une liste (au sens Collection de Java), alors une liste est créée, puis les éléments **list**, **exp1** à **expK** y sont ajoutés.

@Runtime(param)

Si **param** vaut :

- "FreeMemory" : retourne la quantité de mémoire libre,

- "TotalMemory" : retourne la quantité de mémoire allouée,
- "MaxMemory" : retourne la quantité maximale de mémoire allouable.

Retourne la propriété de nom **param**.

@SetValueInEnv(env , key1 , value1 , [... , keyK , valueK] [, returnValue])

Affecte dans l'environnement **env** la clé **key1** (resp **keyK**) à la valeur **value1** (resp **valueK**), puis retourne le dernier paramètre ou **returnValue** (si nombre pair de paramètres, impaire de couple clé,valeur).

@StringToRegexp(expression)

Convertit la chaîne **expression** en une chaîne de type expression régulière pour une utilisation dans une expression régulière.

@SQLEncode(expression)

Convertit la chaîne **expression** en une chaîne de type SQL pour utilisation dans une requête SQL.

@SQLLikeEncode(expression)

Convertit la chaîne **expression** en une chaîne de type SQL-Like pour utilisation dans un "Like" d'une requête SQL. @ToIterate(nbIteration=0 [,fieldName='nIterate'], [envName='@Iterator']) Génère un environnement d'itération d'entier débutant à zéro jusqu'à nbIteration. Le paramètre fieldName est le nom de champ d'affectation du numéro de l'itération ("nIterate" par défaut). Le paramètre envName est le nom de l'environnement généré (par défaut "@Iterator").

Fonctions de fichiers (environnement @File)

Ci-dessous le terme "fichier" désigne aussi bien un dossier qu'un document fichier. Toutes les fonctions site sont accessibles par @File.@Nomfonction() ou directement par @Nomfonction() s'il n'y a pas d'ambiguïté avec un autre environnement.

@FileListRoot()

Liste les unités root du système.

@File2String(file , charset [, offset] [, nbChar])

Retourne le contenu du fichier file de jeu de caractères charset. Débute la lecture du fichier à l'offset offset (zéro par défaut) pour nbChar (-1 par défaut ou tous) caractères.

@FileNbFile(dossier)

Retourne le nombre de fichiers du dossier dossier.

@FileCanonicalPath(file)

Retourne le chemin canonique du fichier file.

@FileAbsolutePath(file)

Retourne le chemin absolu du fichier file.

@Filepath(file)

Retourne le chemin dossier du fichier file.

@FileLastPath(folder , nbFolder)

Retourne les nbFolder dernier dossier du dossier folder. Si nbFolder<0, retourne le dossier (finissant par un /) complément au nb dernier dossier.

@Filesize(file) @Filesizekb(file) @Filesizemb(file) @Filesizegb(file)

Retourne la taille du fichier file en octet (resp : Kb, Mb, Gb)

@FileLastmodified(file)

Retourne la date de dernière modification du fichier file.

@FileExists(file)

Retourne vrai si le fichier file existe.

@FileDirname(file)

Retourne le chemin du dossier du fichier file.

@Filename(file) , FileBasename(file)

Retourne le nom du fichier file (avec extension).

@FileExtention(file) FileExtension(file)

Retourne l'extension du fichier file.

@FileRootname(file)

Retourne le radical du fichier file (dans "/root/toto.doc" c'est "toto"), sinon chaîne vide.

@FileWithouttext @Filewithouttextention()

Retourne le chemin complet du fichier sans son extension.

@FileIsfile(file)

Retourne vrai si le fichier est un document et nom un dossier.

@FileIsDirectory(file)

Retourne vrai si le fichier est un dossier.

@FileConcat(path1 [... , pathK])()

Retourne la concaténation des chemins path1 à pathK en les séparent par un "/".

@FileJava(file)

Convertit le chemin fichier système en un chemin fichier Posfix (Java).

@FileSystem(file)

Convertit le chemin fichier en un chemin fichier système.

@FileJavapath(path)

Ajoute un '/' a la fin de path, s'il n'existe pas. Si path vaut nul, on retourne "/".

@FileTmpPath()

Retourne un dossier temporaire.

@FileGetTmpFile(onlyName)

Retourne un nom de fichier unique temporaire et le crée à vide. Si onlyName est vrai, retourne uniquement le nom du fichier et pas son chemin complet.

@FileGetTmpDir(onlyName)

Retourne un nom de dossier temporaire unique et le crée. Si onlyName est vrai, retourne uniquement le nom du dossier et pas son chemin complet.

@Filedir(dossier , regexp , mustMatch, fileType)

Liste les fichiers et dossiers du dossier dossier. Le paramètre regexp est une expression régulière permettant de filtrer les fichiers et dossiers retournés. Le paramètre mustMatch indique que chaque nom de fichier doit correspondre à l'expression régulière. Le paramètre fileType indique le type de fichier/dossier voulu :

- 0 : pour tous type de fichier (par défaut)
- 1 : pour que les fichiers
- 2 : pour que les dossiers L'environnement retourné met à disposition pour chaque ligne retournée le nom du fichier au travers de la variable FileName et le nom du dossier au travers de DirName.