Fonctions Stalk

Introduction

STalk est un langage permettant l'écriture d'expressions intervenant dans plusieurs fonctionnalités de Workey : règles de comportement des opérations automatiques, règles de calcul des champs, sujet des documents, calcul des colonnes des vues créées par les utilisateurs, etc.

L'objectif de STalk est de fournir un langage suffisamment simple et bien intégré à Workey, évitant l'écriture de code Java pour spécifier les expressions souvent très simples intervenant dans les règles de gestion des processus .

STalk possède beaucoup de caractéristiques d'un langage de programmation complet, mais ne peut être considéré comme un langage dit «universel », car il ne permet pas de déclarer des structures répétitives («tant que», par exemple).

Document de référence

Dans toutes les situations où il peut être utilisé, STalk a pour contexte un document Workey, qui constitue la source, et également la cible, de ses informations . La présence de ce document est implicite et n'a donc pas besoin d'être spécifiée dans le langage lui-même.

Structure du langage

Types

Définition

STalk est un langage typé : les objets intervenant dans une expression STalk, ainsi que le résultat de l'expression, possèdent un type reconnu par Workey. La cohérence des types utilisés est vérifiée lors de l'analyse des expressions. STalk n'oblige cependant pas à déclarer explicitement le type des objets qu'il utilise, car ils sont déduits, dans tous les cas, des types déclarés lors de la conception des formulaires.

Objets de STalk

Une expression STalk peut faire intervenir un certain nombre d'objets, qui sont décrits ci-dessous, et retourne un résultat.

Champs

L'intégration de STalk dans Workey se manifeste par la prise en compte des champs comme source de valeurs. On rappelle qu'un champ d'un document Workey est:

- typé (exemples : “Texte”, “Entier”, “Décimal”, “Date”, “Pièce-jointe”)
- mono ou multivalué. S’il est multivalué, le champ peut comporter plusieurs valeurs.

Une expression STalk peut faire intervenir les champs présents dans le document de référence, qui sont reconnus au moyen de leur nom interne (“Designer name”). La plus simple expression STalk consiste donc en un nom de champ. Par exemple:

Montant

est une expression STalk valide si le document de référence comporte un champ appelé “Montant”. Le type de l’expression est celui du type du champ “Montant”, et l’expression sera multivaluée si le champ “Montant” est déclaré multivalué.

Attention

Attention, le nom des champs utilisés doit respecter les caractères majuscules et minuscules.

Constantes

STalk reconnaît les constantes de type :

- numérique (ex : 12, 324.75)
- chaîne de caractères (exemple "paiement en espèces")
- date (ex : 24/11/2004)
- logique (true, false)
- la constante « valeur nulle » : **?**

Attention dans le cas d’un nombre le ‘?” est vérifié si la valeur est 0.

Opérateurs

Opérateurs arithmétiques

Une expression STalk peut faire intervenir les opérateurs arithmétiques usuels :

- Addition : **+**
- Soustraction : **-**
- Multiplication : *****
- Division : **/**

Exemples d’expressions :

Montant * 0.12

Le résultat est du type du champ Montant.

"Type de paiement : " + "paiement en espèces"

le résultat est une chaîne de caractères.

Montant - Rabais

Le résultat est d’un type compatible avec les champs Montant et Rabais.

Fonctions agrégatives

Ce sont les fonctions SUM, PRODUCT, MINIMA et MAXIMA.

Voir plus loin : « Extensions des opérateurs arithmétiques usuels » et « Fonctions agrégatives ».

Opérateurs relationnels

Une expression STalk peut faire intervenir les opérateurs relationnels usuels:

- Plus grand que: >
- Plus petit que: <
- Égal: = ou ==
- Différent: !=
- Plus grand ou égal : >=
- Plus petit ou égal: <=
- Appartient à: in (A in B retourne vrai si l’élément A est présent dans le conteneur multivalué B)

Exemples d’expressions:

Montant > 10000

TypeClient != "normal"

"A" in myField

Dans tous les cas ci-dessus, le résultat est du type Booléen (valeur **true** ou **false**)

Opérateurs logiques

Les opérateurs logiques reconnus sont:

- Ou exclusif : **or** ou **||**
- Et : **and** ou **&&**
- Négation logique : **!**

Exemple d’expressions:

(Montant > 10000) and (TypeClient != "normal")

Le résultat est du type Booléen (valeur **true** ou **false**)

Opérateurs ensemblistes

Pour les objets multi-valués, STalk reconnaît également les opérateurs ensemblistes suivants:

- Union: **union**
- Intersection: **inter**
- Différence : **minus**
- Appartient à: **in** (A in B retourne vrai si l'élément A est présent dans le conteneur multivalué B)

Exemples d'expressions :

- ListeClients union ListeFournisseurs

Retourne l'union des 2 ensembles de valeurs contenues dans les champs (obligatoirement multi-valués) *ListeClients* et *ListeFournisseurs*.

- "A" in myField

Retourne "vrai" si la valeur "A" est présente dans le champ multivalué myField

Parenthèses

Toute expression STalk peut comporter des parenthèses, sans restriction de niveau. Exemples d'expressions :

(Montant + (Prime * 2) - Rabais) / 100

Fonctions

STalk reconnaît un certain nombre de fonctions, décrites dans les tableaux ci-dessous. Les lignes vertes (de charAt à trim) sont les fonctions de manipulation de chaîne de caractères ; les lignes rouges (de maxima à sum) sont les fonctions d'aggrégation de données.

Fonction	Description	Paramètres	Type de retour
activeWriters()	Retourne la liste des identificateurs des acteurs ayant le droit d'écrire sur le document courant ceci prends aussi en considération la notion de prise en charge.		WKLong[]
actorsHistory()	Retourne dans un conteneur multivalué la liste des identifiants des acteurs étant intervenus sur le document courant. La liste est sans doublons d'acteur.		WKString[]
actorsRoles(*roles*)	Retourne dans un conteneur multivalué la liste des noms internes des acteurs appartenant à au moins un des rôles spécifiées en paramètre.	**WKString[]** roles : Noms des rôles	WKLong[]
actorsUnits(*units*)	Retourne dans un conteneur multivalué la liste des noms internes des acteurs appartenant à au moins une des unités spécifiées en paramètre.	**WKString[]** units : Nom des unités	WKString[]
actorUnits(*actor*)	Retourne la liste des unités auxquelles appartient un acteur.	**WKString** actor : Nom interne de l'acteur	WKString[]
addBusinessDay(*d*, *i*)	Ajoute à une date un nombre de jours ouvrables. Un jour ouvrable est un jour de semaine (du lundi au vendredi inclus), hors des jours fériés à date fixe (Noël, par exemple).Une heure ouvrable est une heure comprise entre 8h et 18h	**WKDate** d : Date initiale**WKInteger** i : Nombre de jours (peut être négatif)	WKDate
addBusinessHour(*d*, *i*)	Ajoute à une date un nombre d'heures ouvrables	**WKDate** d : Date initiale**WKInteger** i : Nombre d'heures (peut être négatif)	WKDate

Fonction	Description	Paramètres	Type de retour
addDay(*d*, *i*)	Ajoute à une date un nombre de jours	**WKDate** d : Date initiale**WKInteger** i : Nombre de jours (peut être négatif)	WKDate
addHour(*d*, *i*)	Ajoute à une date un nombre d'heures	**WKDate** d : Date initiale**WKInteger** i : Nombre d'heures (peut être négatif)	WKDate
addMinute(*d*, *i*)	Ajoute à une date un nombre de minutes	**WKDate** d : Date initiale**WKInteger** i : Nombre de minutes (peut être négatif)	WKDate
addMonth(*d*, *i*)	Ajoute à une date un nombre de mois	**WKDate** d : Date initiale**WKInteger** i : Nombre de mois (peut être négatif)	WKDate
addWeekDay(*d*, *i*)	Ajoute à une date un nombre de jours de semaine	**WKDate** d : Date initiale**WKInteger** i : Nombre de jours (peut être négatif)	WKDate
addYear(*d*, *i*)	Ajoute à une date un nombre d'années	**WKDate** d : Date initiale**WKInteger** i : Nombre d'années (peut être négatif)	WKDate
authorId()	Retour l'identificateur de l'auteur du document		WKString
childrenIds(*type*)	Retourne la liste des identificateurs des documents fils (premier niveau), appartenant au type de document spécifié (utiliser "*" pour tout type de document)	**WKString** type : Nom du type de document	WKInteger[]
concat(*m*, *s*)	Concatène les valeurs d'une expression multivaluée dans une seule chaîne de caractères, en les séparant par une chaîne choisie. Depuis la version 6.10.1, l'expression multivaluée peut être un champ de n'importe quel type, et plus seulement une expression de type Texte.	**WKExpression[]** m : Expression multivaluée**WKString** s : chaîne de séparation	WKString
count(*exp*)	Retourne le nombre de valeurs d'une expression.	**WKExpression** exp	WKInteger
countChildren()	Retourne le nombre de documents fils du document courant		WKInteger
countTypedChildren(*doctype*)	Retourne le nombre de documents fils du document courant d'un type donné. Si "*" est passé, tous les types de documents sont comptabilisés.	**WKString** doctype : Nom du type de document	WKInteger
createTime()	Retourne la date de création du document courant		WKDate
documentState(*ids*)	Retourne le nom interne de l'état actuel d'un ou plusieurs documents	**WKLong[]** ids : Identifiant des documents	WKString
documentType()	Retourne le nom interne du type du document de référence		WKString
documentTypeName()	Retourne le nom localisé du type de document auquel appartient le document courant		WKString
element(*name*, *i*)	Retourne le Nième élément d'un champ multivalué	**WKObject[]** name : Champ multivalué**WKInteger** i : Position de l'élément à retourner	WKObject
fatherId()	Retourne l'identifiant du document père		WKLong
fatherStates()	Retourne la liste des noms internes des états atteints par le document père du document courant		WKString[]
fatherStatesNames(*o*)	Retourne la liste des noms localisés des états atteints par le document père du document courant. (Attention, la valeur à passer en paramètre est inutilisée, mais obligatoire.)	**WKObject** o : *inutilisé*	WKString[]
first(*m*)	Retourne le premier élément du conteneur	**WKObject[]** m : Conteneur multivalué	WKObject
firstActor()	Retourne le nom interne du premier acteur étant intervenu sur le document courant		WKString
firstUnit()	Retourne le nom interne de la première unité étant intervenue sur le document courant (i.e.. l'unité à laquelle appartenait le premier acteur)		WKString
form()	Retourne le nom interne du formulaire utilisé pour éditer le document courant		WKString
formId()	Retourne l'identifiant interne du formulaire utilisé pour éditer le document courant		WKLong
generateUUID()	Renvoie un nouvel identifiant universel unique (UUID)		WKString
getCurrentDay()	Retourne le numéro du jour de la date d'aujourd'hui (1-31)		WKInteger
getCurrentMonth()	Retourne le numéro du mois de la date d'aujourd'hui (0-11)		WKInteger
getCurrentYear()	Retourne l'année de la date du jour		WKInteger
getDay(*fieldDate*)	Retourne le numéro du jour (1-31) de la ou les dates du champ passé en paramètrePar exemple, si le champ Dates_Commandes contient les valeurs [19/06/2020; 03/05/2019; 29/01/2019] la fonction getDay(Dates_Commande) renverra les valeurs [19; 3; 29].	**WKDate[]** fieldDate : Champ date dont on souhaite obtenir le jour	WKInteger[]
getMonth(*fieldDate*)	Retourne le numéro du mois (0-11) de la ou les dates du champ passé en paramètrePar exemple, si le champ Dates_Commandes contient les valeurs [19/06/2020; 03/05/2019; 29/01/2019] la fonction getMonth(Dates_Commande) renverra les valeurs [5; 5; 0].	**WKDate[]** fieldDate : Champ date dont on souhaite obtenir le jour	WKInteger[]
getYear(*fieldDate*)	Retourne les années de la ou les dates du champ passé en paramètrePar exemple, si le champ Dates_Commandes contient les valeurs [19/06/2020; 03/05/2019; 29/01/2019] la fonction getYear(Dates_Commande) renverra les valeurs [2020; 2019; 2019].	**WKDate[]** fieldDate : Champ date dont on souhaite obtenir le jour	WKInteger[]

Fonction	Description	Paramètres	Type de retour
historyOf(*champ*)	Retourne dans une chaîne de caractères un aperçu de l'historique d'un champ. La valeur de ce champ n'est jamais persistée en base.	**WKString** champ : Nom interne d'un champ historisé	WKString
htmlHistoryOf(*champ*)	Retourne dans une chaîne de caractères un aperçu de l'historique d'un champ au format HTML. La valeur de ce champ n'est jamais persistée en base.	**WKString** champ : Nom interne d'un champ historisé	WKString
id()	Retourne l'identifiant du document courant		WKLong
in(*a*)	Retourne vrai si un des éléments de a appartient à la liste bPar exemple, si le champ Mode_Deplacement contient les valeurs ["Voiture";"Velo";"Bus";"Metro"], alors la fonction "Velo" in Mode_Deplacement renverra la true.	**WKObject[]** a : Valeur à chercher (peut être ou non multivaluée)	WKBoolean
last(*m*)	Retourne le dernier élément du conteneur	**WKObject[]** m : Conteneur multivalué	WKObject
lastActor()	Retourne le nom interne du dernier acteur étant intervenu sur le document courant		WKString
lastUnit()	Retourne le nom interne de la dernière unité étant intervenue sur le document courant (i.e. l'unité à laquelle appartenait le dernier acteur)		WKString
ldapId(*ids*)	Retourne la liste des identifiants de l'annuaire qui correspondent à liste liste d'identifiants d'acteurs dans le workflow.Si par exemple le paramètre "ids" contient "101; 456", alors "ldapId (ids)" retournera un conteneur multivalué : "dn=foo;dn=bar", si l'acteur identifié par "101" possède la valeur de l'attribut-clé "dn=foo" dans l'annuaire, et si l'acteur identifié par "456" possède la valeur de l'attribut-clé "dn=bar". Autre exemple, ldapId (writers()) retourne les identifiants des ayant-droit d'écriture sur le document courant.	**WKObject[]** ids	WKString[]
local(*field*)	Retourne le nom de l'alias d'un domaine.Si un domaine de valeur contient les entrées "1*Oui / 2*Non / 3*Sans réponse", et qu'un champ "Champ_avec_Domaine" utilisant ce domaine contient comme valeur "2;3", le résultat de la fonction local("Champ_avec_Domaine") sera "Sans réponse;Non".	**Nom interne d'un champ ayant un domaine défini avec une liste de valeurs** field	WKString
localizedState()	Retourne le nom local de l'état courant (dans la langue utilisée pour ouvrir le document). Fonction obsolète, la fonction "stateName" la remplace.		WKString
localTextToNumber(*expr*, *format*)	Retourne un nombre décimal évalué depuis une chaîne de caractère, en utilisant un format identique à celui utilisé par la fonction text(expr, format)Par exemple, sur un serveur français, la fonction localTextToNumber("1 234 567,891", "#,#00.0#") renverra la valeur "1234567.89". Attention, en français, les espaces séparateurs de millier doivent être des espace insécables (AltGr+0160)!	**WKString** expr : Chaîne de caractères**WKString** format : Format de nombre valide	WKNumber
max(*a*, *b*)	Calcule le maximum de 2 valeurs	**WKNumber** a**WKNumber** b	WKNumber
message(*id*)	Retourne le texte localisé d'un message	**WKObject** id : Soit l'identifiant du message, soit son nom interne	WKObject
min(*a*, *b*)	Calcule le minimum de 2 valeurs	**WKNumber** a**WKNumber** b	WKNumber
nextOperationsNames()	Retourne la liste des noms internes des prochaines opérations pouvant être effectuées sur le document courant		WKString[]
now()	Retourne la date et heure actuelle		WKDate
operationActors(*operation*)	Retourne la liste des acteurs ayant réalisé l'opération désignée	**WKString** operation : Nom interne de l'opération	WKString[]
operationCountTimes(*operation*)	Retourne le nombre de fois qu'une opération a été effectuée sur le document courant	**WKString** operation : Nom interne de l'opération	WKInteger
operationFirstTime(*operation*)	Retourne la date à laquelle une opération a été effectuée pour la première fois sur le document courant	**WKString** operation : Nom interne de l'opération	WKDate
operationLastTime(*operation*)	Retourne la date à laquelle une opération a été effectuée pour la dernière fois sur le document courant	**WKString** operation : Nom interne de l'opération	WKDate
prettyDate(*date*)	Retourne la date formatée en fonction de la langue de l'utilisateur.Par exemple, si le champ Dates_Commandes contient les valeurs [19/06/2020; 03/05/2019; 29/01/2019] et que l'utilisateur est en français, la fonction prettyDate(Dates_Commande) renverra les valeurs ["19 juin 2020"; "3 mai 2019" ; "29 janv. 2019"]. Également, les dates autour de la date courante sont notées différement ("Avant-hier", "hier", "Ajourd'hui", "Demain", "Après-demain").	**WKDate[]** date : Champ date à formatter	WKString
prettyDateTime(*date*)	Retourne la date et l'heure formatée en fonction de la langue de l'utilisateur.Par exemple, si l'utilisateur est en français, la fonction prettyDateTime(now()) renverra par exemple Aujourd'hui 17:20.	**WKDate[]** date : Champ date à formatter	WKString
processInstanceId()	Retourne l'identifiant du processus auquel appartient le document courant		WKLong
processType()	Retourne le nom interne du processus auquel appartient le document courant		WKString

Fonction	Description	Paramètres	Type de retour
processTypeName(*o*)	Retourne le nom localisé du processus auquel appartient le document courant. (Attention, la valeur à passer en paramètre est inutilisée, mais obligatoire.)	**WKObject** o : *inutilisé*	WKString
processVersion()	Cette fonction retourne le numéro de version du processus auquel appartient le document courant.		WKInteger
property(*property*)	Retourne la valeur d'un propriété définie au niveau du serveur WorkeyPar exemple, property("com.clog.workey.archive.directory") retournera le répertoire défini pour l'archivage des documents, par exemple "/Applications/workey/archives"	**WKString** property : Nom de la propriété système	WKString
readers()	Retourne la liste des identifiants des acteurs autorisés à lire le document courant (sans tenir compte de leurs rôles)		WKLong[]
roleActorsIds(*name*)	Retourne une liste d'identificateurs d'acteurs du rôle désigné. Dans un conteneur monovalué, la première occurrence de la liste est retournée. Dans un conteneur multivalué, l'intégralité des occurrences de la liste est retournée.	**WKString** name : Nom interne du rôle	WKLong[]
round(*n*)	Arrondi le nombre passé en paramètre	**WKNumber** n	WKNumber
siblings(*documentType*)	Retourne la liste des identifiants des documents de la fratrie du document, appartenant au type de document spécifié. N.B. Le document courant n'est pas inclus dans la liste.	**WKString** documentType : Utiliser "*" pour tout sélectionner tous les types de document	WKLong[]
state()	Retourne le nom interne de l'état courant du document		WKString
stateActors(*name*)	Retourne la liste des acteurs ayant placé le document dans l'état désigné	**WKString** name : Nom interne de l'état	WKString[]
stateCountTimes(*name*)	Retourne le nombre de fois qu'un état a été atteint par le document courant	**WKString** name : Nom interne de l'état	WKInteger
stateFirstTime(*name*)	Retourne la date à laquelle un état a été atteint pour la première fois par le document courant	**WKString** name : Nom interne de l'état	WKDate
stateLastTime(*name*)	Retourne la date à laquelle un état a été atteint pour la dernière fois par le document courant	**WKString** name : Nom interne de l'état	WKDate
stateName()	Retourne le nom local de l'état courant (dans la langue utilisée pour ouvrir le document).		WKString
states()	Retourne la liste des noms internes des états atteints par le document courant		WKString[]
statesNames()	Retourne la liste des noms localisés des états atteints par le document courant		WKString[]
storedValue(*name*)	Cette fonction retourne la valeur enregistrée du champ dont le nom figure en paramètre. Le résultat a donc le même type de données que le champ concerné. Si aucune valeur n'est encore enregistrée, la fonction retourne la valeur inconnue.Exemple: v:= storedValue(""Description""); retourne la valeur enregistrée dans la base de données pour le champ nommé ""Description"".	**WKString** name : Nom interne du champ	WKObject
string(*mv*, *sep*)	Retourne la chaîne des valeurs de l'expression, séparées par un symbole. **Attention :** depuis la version 6.10.1, cette fonction est obsolète. Privilégiez la méthode concat(mv, c) à la place.	**WKString[]** mv : Expression multivaluée**WKString** sep : séparateur	WKString
subject()	Retourne la valeur du "sujet" du document courant		WKString
synchroStatus()	Obtient le nombre de synchronisations dont le document courant attend la réalisation		WKInteger
text(*expr*, *format*)	Retourne un texte formaté	**WKObject** expr : Expression à formatter**WKString** format : "shortLdapName" pour obtenir le nom court correspondant à un DN , "e_mail" pour obtenir l'adresse e-mail, ou format de date valide (cf. annexe)	WKString
textToDate(*expr*, *format*)	Convertit une chaîne de caractères en une date, suivant un format de date donné.	**WKString** expr : Expression à formatter**WKString** format : syntaxe de formatage (ex. "dd/MM/yyyy")	WKDate
textToInteger(*str*)	Cette fonction retourne un nombre entier évalué depuis un conteneur de type "chaîne de caractères".Par exemple, textToInteger ("42") retourne la valeur 42. Si la chaîne de caractères ne peut être interprétée comme un nombre entier, la fonction retourne l'entier nul.	**WKString** str	WKInteger
textToNumber(*str*)	Cette fonction retourne un nombre décimal évalué depuis un conteneur de type "chaîne de caractères".Par exemple, textToInteger ("3.14") retourne la valeur 3.14. Si la chaîne de caractères ne peut être interprétée comme un nombre entier, la fonction retourne l'entier nul.	**WKString** str	WKFloat
timeFormat(*duree*, *format*)	Retourne un texte formaté pour afficher une durée.	**WKLong** duree : Entier exprimé en millisecondes**voir XXX pour la syntaxe** format	WKString
uniqueValues(*name*)	Renvoie les valeurs uniques d'un champ multivaluéPar exemple, sur un champ "Entiers" contenant les valeurs "1;2;2;3;3;4", la fonction uniqueValues(Entiers) renverra les valeurs "1;2;3;4".	**WKObject[]** name : Champ multivalué	WKObject[]
unitActorsIds(*name*)	Cette fonction retourne la liste des identifiants des acteurs appartenant à l'unité est passé en paramètre.Exemple: l:= unitActorsIds("Ventes") union unitActorsIds ("Production");	**Wkstring** name : Nom interne de l'unité	WKLong[]

Fonction	Description	Paramètres	Type de retour
unitLocalName(*id*)	Retourne le nom localisé d'unités organisationnelles, le paramètre peut être soit le nom interne, soit l'identifant des unités.	**WKString[]** id : Identifiant ou nom interne de l'unité	WKString[]
unitsHierarchy(*unit*, *from*, *to*)	Retourne la liste des unités se situant dans la hiérarchie d'une unité "unit", du niveau "from" au niveau "to". Les paramètres "from" et "to" peuvent être négatifs ou nuls.Par exemple, unitsHierarchy ("dn=U", -1, -1) retourne les unités filles de l'unité U.	**WKString** unit**WKInteger** from**WKInteger** to	WKString[]
unitsOf(*id*)	Cette fonction retourne la liste des unités d'un acteur identifié par son identifiant ou son nom interne. Un paramètre multivalué est accepté, la fonction retournant alors l'union (avec doublons éventuels) de toutes les unités d'appartenance.Exemple : unitsOf(323) retourne "Direction". unitsOf("gontran") retourne ["Direction Commerciale", "Service Etranger"],	**WKString** id : Identifiant ou nom interne de l'acteur	WKString[]
updateTime()	Retourne la date de la dernière mise à jour du document courant		WKDate
userDisplayName()	Retourne le nom usuel de l'utilisateur courant		WKString
userId()	Retourne l'identifiant de l'utilisateur courant		WKLong
userLanguage()	Retourne le language utilisé par l'utilisateur courant. Si aucun n'a été précisé, la propriété système com.clog.workey.engine.defaultLanguage est utilisée.		WKString
userName()	Retourne le nom interne de l'utilisateur courant		WKString
userRoles()	Retourne la liste des noms internes des rôles joués par l'utilisateur courant.		WKString[]
userRolesNames(*o*)	Retourne la liste des localisés des rôles joués par l'utilisateur courant. (Attention, la valeur à passer en paramètre est inutilisée, mais obligatoire.)	**WKObject** o : *inutilisé*	WKString[]
userUnits()	Retourne la liste des noms interne des unités auxquelles appartient l'utilisateur courant. L'ordre des éléments retenus pour le choix courant selon cette formule dépend de l'ordre de création de ces éléments. C'est l'ordre que vous pouvez observer dans votre base qui déterminera le comportement observé.		WKString[]
writers()	Retourne la liste des identifiants des acteurs ayant le droit d'écrire sur le document courant		WKLong[]
charAt(*c*, *i*)	Retourne la valeur numérique du caractère occupant la position i dans la chaîne de caractères c. Si i est négatif ou supérieur à la longueur de c, la valeur -1 est retournée.	**WKString** c : Chaîne de caractères**WKInteger** i : Nombre entier représentant la position	WKString
endsWith(*c*, *s*)	Retourne vrai si la chaîne c se termine par la chaîne sPar exemple, endsWith("voilou", "lou") retourne vrai.	**WKString** c : Chaîne de caractères**WKString** s : Chaîne recherchée	Boolean
indexOf(*c*, *s*)	Retourne la position de la première apparition de la chaîne recherchée s dans la chaîne c (retourne -1 si elle n'est pas présente)	**WKString** c : Chaîne de caractères**WKString** s : Chaîne recherchée	WKInteger
indexOfFrom(*c*, *s*, *i*)	Retourne la position de la première apparition de la chaîne recherchée s dans la chaîne c à partir de la position i (retourne -1 si elle n'est pas présente)	**WKString** c : Chaîne de caractères**WKString** s : Chaîne recherchée**WKInteger** i : Index du début de la recherche	WKInteger
lastIndexOf(*c*, *s*)	Retourne la position de la dernière apparition de la chaîne recherchée s dans la chaîne c (retourne -1 si elle n'est pas présente)	**WKString** c : Chaîne de caractères**WKString** s : Chaîne recherchée	WKInteger
lastIndexOfFrom(*c*, *s*, *i*)	Retourne la position de la dernière apparition de la chaîne recherchée s dans la chaîne c à partir de la position i (retourne -1 si elle n'est pas présente)	**WKString** c : Chaîne de caractères**WKString** s : Chaîne recherchée**WKInteger** i : Index du début de la recherche	WKInteger
length(*c*)	Retourne la longueur de la chaîne c	**WKString** c : Chaîne de caractères	WKInteger
startsWith(*c*, *s*)	Retourne vrai si la chaîne c commence par la chaîne s	**WKString** c : Chaîne de caractères**WKString** s : Chaîne recherchée	Boolean
subString(*c*, *i*)	Retourne la sous-chaîne de c commençant à la position i	**WKString** c : Chaîne de caractères**WKInteger** i : Position du début de la sous-chaîne	WKString
subStringTo(*c*, *i*, *k*)	Retourne la sous-chaîne de c commençant à la position i et se terminant à la position k-1	**WKString** c : Chaîne de caractères**WKInteger** i : Position du début de la sous-chaîne**WKInteger** k : Position du fin de la sous-chaîne	WKString
toLower(*c*)	Retourne la chaîne c convertie en minuscules	**WKString** c : Chaîne de caractères	WKString
toUpper(*c*)	Retourne la chaîne c convertie en majuscules	**WKString** c : Chaîne de caractères	WKString
trim(*c*)	Retourne la chaîne c avec les espaces retirés au début et à la fin	**WKString** c : Chaîne de caractères	WKString
maxima(*aggreg*)	Renvoie le nombre le plus élevé ou la date la plus récente de la liste d'éléments passés en paramètre	**WKObject[]** aggreg : Peut être un champ WKDate ou WKNumber	WKObject
minima(*aggreg*)	Renvoie le nombre le plus faible ou la date la plus ancienne de la liste d'éléments passés en paramètre	**WKObject[]** aggreg : Peut être un champ WKDate ou WKNumber	WKObject
product(*aggreg*)	Renvoie le produit des éléments passés en paramètre	**WKNumber[]** aggreg	WKObject
sum(*aggreg*)	Renvoie la somme des éléments passés en paramètre	**WKNumber[]** aggreg	WKObject

Variables

L’utilisation de variables est permise dans STalk. Une variable se déclare au moyen d’une assignation de valeur, en utilisant l’opérateur d’assignement : =, suivi d’une expression STalk. Par exemple:

```
MyShare := Montant / 2
```

L’expression ci-dessus :

- déclare une variable nommée MyShare
- détermine le type de cette variable comme étant celui de l’expression à droite de l’opérateur d’assignement
- calcule l’expression et renseigne la variable à cette valeur

Le type de la variable est donc déterminé, mais n’a pas besoin d’être explicité. Naturellement, une variable peut intervenir dans des phrases subséquentes : Exemple :

```
(MyShare > 100) and (MyShare < 200)
```

Il faut éviter de nommer une variable du même nom qu’un champ du document de référence, ou du même nom qu’une fonction connue de STalk.

Phrases

Une expression STalk peut comporter plusieurs phrases, séparées par des points-virgule. Par exemple :

```
MyShare := Montant / 2;
GoodThing := (MyShare > 100) and (MyShare < 200);
```

Le résultat de l’expression est alors le résultat de la dernière phrase.

Structures conditionnelles

Une condition s’exprime, classiquement, par une structure de type if (condition) expression1 else expression2, où “condition” désigne une expression STalk de type booléen, et “expression1”, “expression2” deux expressions STalk quelconques, la clause “else” étant optionelle.

Exemples:

```
if (MyShare < 1000)   Remain := Montant - MyShare;
if (Remain > 0)   « Il m'en reste » ; else « il ne reste rien » ;

if (Type == "1") TT := 160; else if (Type == "2") TT := 30; else TT := 60;
```

Exemples d’utilisation

- Concaténation :Text := “Le document de type: ” + documentType() + ” est dans l’état: ” + state()
- Valeur initiale d’un champ :now();

- Valeur calculée d’un champ :Montant – Rabais
- Sujet d’un document :”Affaire suivie par : ” + userName()
- Règle de contrôle pour une opération automatique :Montant > 10000
- Contenu d’une colonne dans une vue :if (Solde > 0) “green.gif” else “red.gif”

Extensions des opérateurs arithmetiques usuels

Les principaux opérateurs arithmétiques (+, -, *, /) prennent en compte les conteneurs multi-valués

Cas de deux conteneurs multi-valués

Pour autant que les deux conteneurs A et B aient le même nombre de valeurs, l’expression A op B retourne un conteneur de même cardinalité dans lequel la i-ème valeur est égale à la i-ème valeur de A op i-ème valeur de B, avec op = +, -, * ou /.

Par exemple, si le champ Prix comporte 3 valeurs (10 , 12, 20) et le champ Qty 3 valeurs (1, 3, 2), l’expression Prix * Qty retourne les valeurs (10, 36, 40).

Cas d’un conteneur multi-valué et un conteneur mono-valué

Si A est multi-valué et B mono-valué , l’expression A op B retourne un conteneur de même cardinalité que A dans lequel la i-ème valeur est égale à la i-ème valeur de A op valeur de B, avec op = +, -, * ou /. Par exemple, si le champ Prix comporte 3 valeurs (10 , 12, 20) et le champ Rabais la valeur 0.1, l’expression Prix * Rabais retourne les valeurs (1, 1.2, 2).

Fonctions agrégatives

Ces nouvelles fonctions calculent un agrégat sur un conteneur multivalué .

sum

Si A est multi-valué, sum(A) retourne la somme des valeurs de A Par exemple, si le champ Qty comporte 3 valeurs (1, 3, 2), sum(Qty) retourne 6. Si le champ Prix comporte 3 valeurs (10 , 12, 20), sum(Prix*Qty) retourne 86 .

product

Si A est multi-valué, product(A) retourne le produit des valeurs de A Par exemple, si le champ Qty comporte 3 valeurs (1, 3, 2), product(Qty) retourne 6.

minima

Si A est multi-valué, minima(A) retourne le minimum des valeurs de A Par exemple, si le champ Qty comporte 3 valeurs (1, 3, 2), minima(Qty) retourne 1.

maxima

Si A est multi-valué, maxima(A) retourne le maximum des valeurs de A Par exemple, si le champ Qty comporte 3 valeurs (1, 3, 2), maxima(Qty) retourne 3.

Fonctions particulières

La fonction « text »

Cette fonction a pour but de produire une chaîne de caractères en mettant en forme une valeur de type quelconque. La syntaxe est :

text (valeur, format)

Paramètres

- **valeur** : conteneur de type quelconque, typiquement un nom de champ du document
- **format** : chaîne de caractères représentant le format à appliquer à la valeur (Voir annexe).

La fonction text (container, syntaxe) est maintenant applicable sur un container multivalué : elle est appliquée à chaque valeur de ce container.

Par exemple, si ‘my_names’ est une variable multivaluée contenant des identificateurs d’acteurs : « dn=foo ; dn=bar »,

alors text (my_names, “shortLdapName”) retourne une variable multivaluée contenant:

“foo;bar”

Exemple:

text (Date_remise, "EEE, d MMM yyyy")

Si le champ Date_remise est du type « date » et contient la date du 24 novembre 1947, et pour une localisation française, la fonction text retourne : “Lun., 24.11.1947”

Implémentations actuelles

- **Type « Date »**

Pour une valeur de type « Date », les formats autorisés sont ceux de la classe Java « Format », fournis en annexe . La « localisation » s’effectue automatiquement en fonction de la langue de l’utilisateur.

- **Type « Text »**

Pour une valeur de type « Text », le seul format actuellement implémenté est « shortLdapName » . Si la valeur contient un nom complet Ldap (dn), ce format

le convertit en un nom court. Exemple : text (“cn=Tutti,ou=Frutti,ou=AnRutti”, “shortLdapName”) retourne “Tutti”.

- **Autres types**

Non encore implémentés, mais le premier candidat est le formatage « monétaire » pour les champs numériques.

Conseil

En raison de la présence de cette fonction « text », la fonction « userName() » retourne maintenant le dn complet de l’acteur courant, et non plus le « short name ». Il faut donc appliquer la fonction « text » pour obtenir le même résultat :text (userName (), “shortLdapName”)

Formatage des durées

Une durée est un nombre entier, exprimé en millisecondes. La fonction «timeFormat » permet de formater une durée en une chaîne de caractères : timeFormat (entier, format) retourne le nombre entier représentant la durée formaté par la chaîne de caractères format.

Paramètres

- **entier**
- **format** peut contenir les caractères réservés suivants :

character	**durationelement**
y	years
M	months
d	days
H	hours
m	minutes
s	seconds
S	milliseconds
‘text’	arbitrary text content

L’occurrence de ces caractères dans format est remplacée par la valeur de la composante correspondante dans la chaîne formatée construite. Tout autre caractère n’est pas interprété et figurera tel quel dans la chaîne résultante.

Par exemple, une heure correspond à 3600 secondes, donc 3600000 milli-secondes.

```
timeFormat (3600000, "Hh M' Ss") ;
```

affichera : “1h 0′ 0s” .

```
timeFormat ( ( Date2 - Date1 ) , "d J, H:m:s");
```

affichera : 3 J, 23:59:59

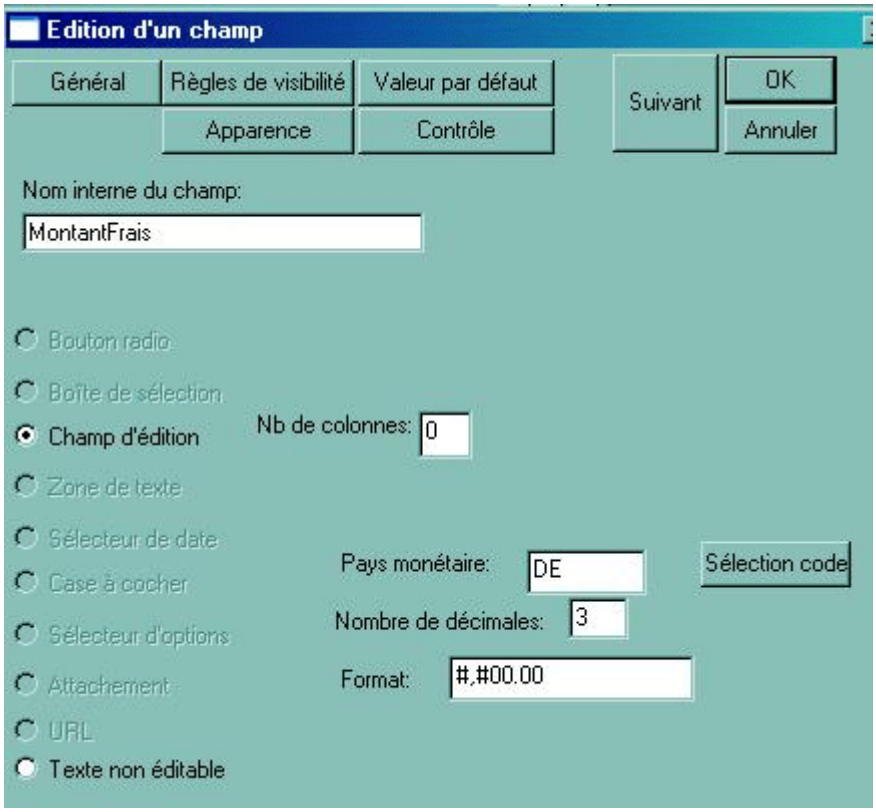
Les caractères réservés peuvent être répétés consécutivement pour préciser le nombre minimal de chiffres affichés pour une zone : le nombre de caractères est le nombre minimal de chiffres, des zéros étant affichés en préfixe pour compléter. Par exemple, timeFormat (3600001, “HH h MM’ SS s LLL”) affichera : 01 h 00` 00 s 001. Pour utiliser le caractère ” dans le format, il faut le faire précéder du caractère \ . Par exemple, timeFormat (3600000, “HHh MM’ SS\””) affichera : 01h 00′ 00″ .

Formats de présentation des données

Ce paragraphe présente les possibilités de présentation (et de saisie) des données dans les champs de formulaires. Workey pouvant opérer dans un contexte international, cette présentation pourra être personnalisée pour chaque pays.

Boîte de dialogue

La saisie des informations relatives au format s’effectue dans l’onglet « Apparence » de la boîte de dialogue de chaque champ :



La présentation effective des champs permettant le formatage dépend du type de champ (date, entier, décimal ou autre).

Champs de type « date »

Les champs de type Date peuvent être représentés en utilisant un format. Ce format est très complet, il est celui de la classe Java DateFormat et est présenté en annexe . Il doit être renseigné dans le champ « Format » de la boîte de dialogue.

Par défaut, le réglage d'usine est « dd/MM/yyyy », qui correspond à la syntaxe francophone et permet d'entrer sans problème « 14/7/1789 ». (qui sera corrigé à l'affichage en « 14/07/1789 »).

Cas particulier pour les dates

L'exemple donné est la saisie d'une chaîne "33/33/33" pour un format "dd/MM/yyyy".

En fait, il y a 2 mécanismes mis en cause dans cet exemple.

1. La syntaxe "yyyy" provoque la saisie littérale de l'année. Ainsi "33" est interprété comme l'an 33 après JC. L'utilisateur doit saisir "2006" et non pas "06" s'il veut indiquer l'année 2006. Si on veut une complémentation automatique de la date, il faut utiliser une syntaxe "yy" , comme "dd/MM/yy". Dans ce cas, "06" est interprété comme 2006.
2. Le mode par défaut pour la saisie des dates est le mode dit "astucieux". Dans ce mode, Workey va activer des règles astucieuses pour interpréter la saisie.

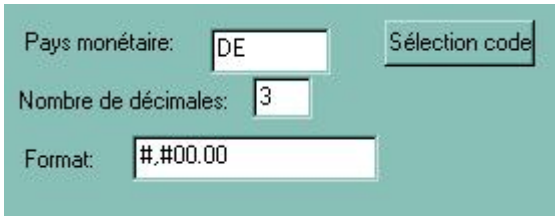
Comme il sait bien qu'une année n'a que 12 mois, il va interpréter le nombre "33", spécifié pour le mois, comme "24 + 9 mois", c'est à dire 2 ans et 9 mois. De même, il va interpréter le nombre "33" pour un jour comme étant "33" le nombre de jours du mois courant" plus un mois. Par exemple, "29/02/2006" sera ré-interprété comme "01/03/2006".

En application des 2 principes énoncés, la saisie de "33/33/33" pour un format "dd/MM/yyyy" donnera: – le dernier "33" interprété comme "33 après JC" – le "33" du milieu interprété comme "9" et l'année convertie en "35" – le premier "33" comme "30", "3" jours étant affectés au mois suivant, qui devient "10" La date saisie est donc le 3 octobre de l'an de grâce 35 après JC.

Il est suggéré d'adopter le mode "non astucieux" qui refusera la saisie de la date de l'exemple Donc soit utiliser une syntaxe "yyyy" et dans ce cas il est nécessaire de faire un contrôle postérieur sur la date. Workey pourrait proposer par défaut un contrôle du type "l'année ne peut être inférieure à 1900". Un contrôle STalk sur le champ est également possible. Soit opter pour une syntaxe "yy".

Champs de type « nombre »

Il s'agit des champs de type « Entier » et « Numérique » (comportant une partie décimale). Pour ce type de champs, le format pourra être spécifié de plusieurs manières, non exclusives, présentées ci-après par ordre de complexité. L'ensemble des éléments pour le formatage est accessible dans cette partie de la boîte de dialogue :



La spécification du pays

L'élément essentiel pour la « localisation » du format est la donnée du pays de référence.

La spécification d'un pays n'est pas connue de Workey par la préférence exprimée par l'utilisateur sur son poste de travail, contrairement à la langue. Le pays doit être spécifié pour chaque champ utilisant un format localisé. Cela permet de mêler dans un même formulaire différents formats localisés.

Une extension prévue est de pouvoir spécifier le pays par défaut au niveau du processus, ce qui éviterait de devoir le spécifier pour chaque champ.

La codification des pays

Chaque pays possède un code associé selon la norme ISO-3166. Cette liste est fournie en annexe III. Workey Designer permet de sélectionner le code dans un menu déroulant (bouton Sélection code). Dans l'exemple ci-dessus, le code « DE » correspond à l'Allemagne.

Format monétaire

C'est le format qui sera sélectionné par Workey si un code de pays a été sélectionné et en l'absence d'un format spécifiquement donné.

La manière de représenter les nombres dans une monnaie sera adaptée aux règles de ce pays. Outre le choix de symboles monétaires différents, la manière de séparer les parties décimales, de regrouper les milliers et millions, de placer le symbole monétaire en préfixe ou en suffixe, varient d'un pays à l'autre. Par exemple, le nombre 1525.75 sera représenté dans un format monétaire « 1 525,75 EUR » en France (noter l'espace entre le 1 et le 5), et « Sfr. 1525.75 » en Suisse.

Par exemple, le choix du code « FR », représentant la France, affiche le nombre 3 sous la forme « 3,00 EUR ».

Choix du nombre de décimales

Ce choix n'est évidemment possible que pour les champs de type « Numérique ». Il revient à adopter les conventions du pays spécifié pour l'affichage des nombres (choix des séparateurs) et à fixer la nombre de décimales affichées à la valeur choisie

Choix d'une structure

La manière la plus complète de formater un nombre est de spécifier une

structure complète. Par exemple, la structure “#,#00.0#” spécifie un minimum de deux chiffres décimaux, un chiffre décimal au moins et au maximum deux. Elle affichera le nombre 1234.56 sous la forme « 1,234.56 ». L’annexe III fournit une documentation complète de la syntaxe de ces structures.

Précisions

La localisation joue le premier rôle dans les formats: c’est elle qui détermine les symboles utilisés dans le format. par exemple, si la localisation est “fr”, le symbole utilisé pour séparer la parie décimale est la virgule, celui pour grouper les milliers est l’espace.

- cette localisation est renseignée automatiquement par Workey qui ne fait que lire le paramètre de langue dans la configuration du poste de l’utilisateur;
- cette localisation peut être changée pour un champ particulier en choisissant un pays “monétaire” dans l’onglet “apparence”;
- les symboles utilisés pour définir un format particulier ne sont pas pris littéralement, mais sont interprétés selon cette localisation; par exemple, dans le format “#,##0.##”, la virgule représente la séparation des milliers (donc un espace si la localisation est “fr”), et le point la séparation décimale (donc la virgule si la localisation est “fr”);
- pour obtenir un format affichant des pour-cents, “#.00%”, par exemple (attention, le nombre est multiplié par 100 avant l’affichage)
- pour compliquer le tout, il y a un bug apparemment dans la méthode “parse” du package java : la méthode arrête son analyse au premier espace rencontré, ce qui est gênant pour une chaîne comme “-12 345,67” dans une localisation “fr” (la méthode renvoie -12 dans ce cas!).