StarDispatch

INTRODUCTION

StarDispatch est le module de regroupement, de distribution, de copie ou de tri des données de la solution de gestion des documents StarJet.

Ce module peut :

- Lire plusieurs états différents en entrée (XML ou ASCII) (XML ou ASCII).
- Découper les états d'entrée selon des critères prédéfinis.
- Trier les états d'entrée selon des critères prédéfinis (**fonction de tri**).
- Dupliquer un état d'entrée (**fonction de copie**).
- Regrouper différents états en entrée dans le même état de sortie (**fonction de regroupement**).
- Générer plusieurs états en sortie (**fonction de distribution**).
- Découper un état de sortie en plusieurs documents indépendants.
- Regrouper différents documents indépendants en un même lieu.

Illustrons par quelques exemples les utilisations envisageables de ce module :

1. **Premier exemple :**

En fin d'année votre société édite, comme pour chaque fin de mois, les bulletins de salaire du mois de décembre. Elle édite également, pour chacun des salariés, un bilan social reprenant les différents avantages, cotisations et rémunérations perçus par les salariés. Ces deux états contiennent, à des emplacements différents sur la page de données, le numéro matricule du salarié concerné.

StarDispatch vous permet de lire les deux états de structure différente et de générer un seul état contenant les pages du bulletin de salaire d'un salarié, suivies par les pages du bilan social de ce même salarié.

L'intérêt de cette application réside évidemment dans l'expédition d'un pli unique à ce salarié, réduisant les frais d'affranchissement et le nombre d'enveloppes requises.

1. **Second exemple :**

Votre entreprise dispose de plusieurs centres opérationnels situés sur le territoire français. Le site central génère les factures de l'ensemble des sites.

StarDispatch peut lire l'ensemble de votre état de facturation et expédier sur des imprimantes (ou des fichiers) les factures concernant vos sites de Marseille, de Lyon, de Paris ou de Strasbourg.

Si certains de vos clients préfèrent recevoir leur facture par message électronique, alors que d'autres préfèrent recevoir des factures sur papier, StarDispatch peut découper l'état pour appeler StarEmail ou StarPage selon une donnée indicative contenue dans les pages de données.

Vous pouvez le constater : StarDispatch est un automate extrêmement puissant vous permettant d'envisager vos éditions de manière plus souple : le bon Document, sous le bon Format, au bon Moment, au bon Endroit.

Partie intégrante de l'ensemble StarJet, StarDispatch communique naturellement avec l'ensemble des autres modules de la famille : StarPage pour produire les documents sur papier, StarFind pour les archiver, StarEmail pour générer l'image correspondant au document et l'expédier par messagerie électronique, StarFax pour l'envoyer par télécopie.

StarDispatch est conçu, comme tous les autres modules de la famille des logiciels StarJet, pour être utile, efficace et performant.

CONCEPTS

FONCTIONNALITES DE STARDISPATCH

Star Dispatch est un module permettant de gérer la distribution des documents et faisant partie intégrante de la solution de gestion des documents StarJet dont APPIC est l'éditeur.

Il permet :

- la copie des documents c'est à dire leur obtention en 'x' exemplaires.

Le service comptable d'une entreprise ainsi que le client ont besoin de copies d'une facture. StarDispatch peut produire chaque copie, formatée spécifiquement aux besoins du destinataire, ainsi que l'envoyer.

- le regroupement des documents c'est à dire la combinaison de plusieurs fichiers de données en un document.

StarDispatch permet, par exemple, le regroupement au sein d'un même document de la facture et de la commande d'un même article.

- la distribution des documents c'est à dire l'envoi de documents à différents destinataires.

Par exemple, pour une entreprise disposant de plusieurs centres opérationnels et centralisant l'édition des factures, StarDispatch permet un envoi aisé des factures aux différents destinataires selon le mode choisi (e-mail, courrier postal, télécopie…).

- le tri des documents (réservé aux fichiers en entrée de type non XML) c'est-à- dire, l'ordonnancement de documents d'après différents critères.

Les fichiers provenant de la production ne sont pas toujours présentés dans

l'ordre qu'il convient pour pouvoir être immédiatement traités. StarDispatch permet de réorganiser la présentation des pages sans en changer le contenu en définissant et appliquant un ou plusieurs critères de tri.

LES DONNEES EN ENTREE

Les données à traiter sont fournies à StarDispatch de trois manières différentes :

1. Elles sont préparées et placées dans un fichier ASCII ;
2. Elles sont préparées et placées dans un fichier XML ;
3. Elles proviennent de l'entrée standard (ex : clavier).

LA PAGE

StarDispatch utilise une entité logique nommée **Page** pour contenir les données brutes et préserver leur intégrité.

L'information utile est identifiée au sein de l'entité page par des attributs définis par l'utilisateur que l'on nomme **Tags** ou, par les données elle mêmes.

Dans le cas d'un flot de type ASCII, les informations brutes sont découpées suivant 3 critères sur la **Page** :

1. Le nombre de lignes
2. Le code d'éjection
3. La chaîne de rupture (un ou plusieurs mots-clefs)

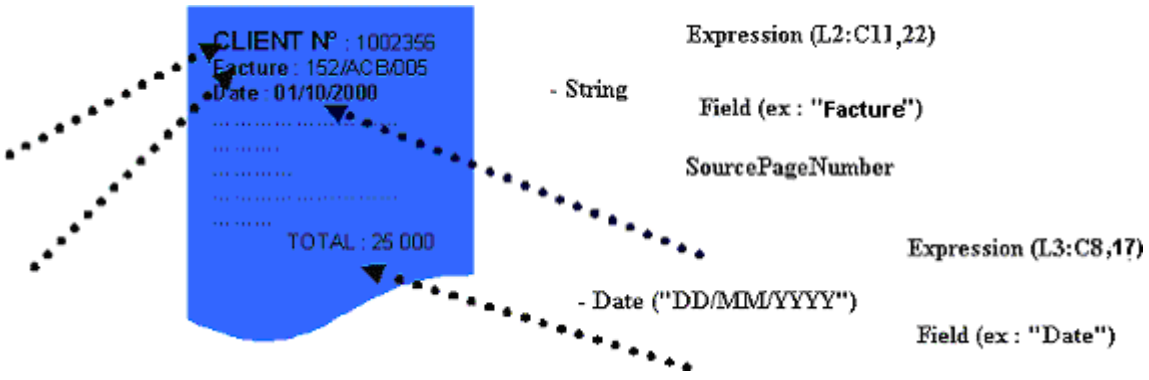
Dans le cas d'un flot de type XML, les informations sont découpées suivant un niveau de découpage.

StarDispatch lit au fur et à mesure les données de la source. Chaque fois qu'un groupe de données est détecté, StarDispatch crée une entité logique **Page** pour y ranger ces données.

Cette entité logique est créée lors du découpage des données en entrée, et elle sera détruite après sa sortie du système.

LE TAG

L'information contenue dans une **Page** et traitée par StarDispatch, peut être identifiée par des **Tags** spécifiques à l'utilisateur.



Les **Tags** peuvent être identifiés dans le cas d'une page ASCII par :

- Leur localisation sur la page (ligne et colonne),
- Leur localisation relative à un mot-clé,
- Leur localisation plus leur format (ex : format de la date).

Les Tags peuvent être identifiés dans le cas d'une page XML par :·

- leur position dans la page XML définie par une expression XPath.

Les **Tags** peuvent être identifiés dans le cas d'une page XML par :

- leur position dans la page XML définie par une expression XPath.

LE FLOT

StarDispatch dispose d'une entité logique nommée **Flot** pour effectuer la liaison entre les différentes unités de traitement, d'une part, ainsi que pour transporter les entités pages d'un point à l'autre dans le système.

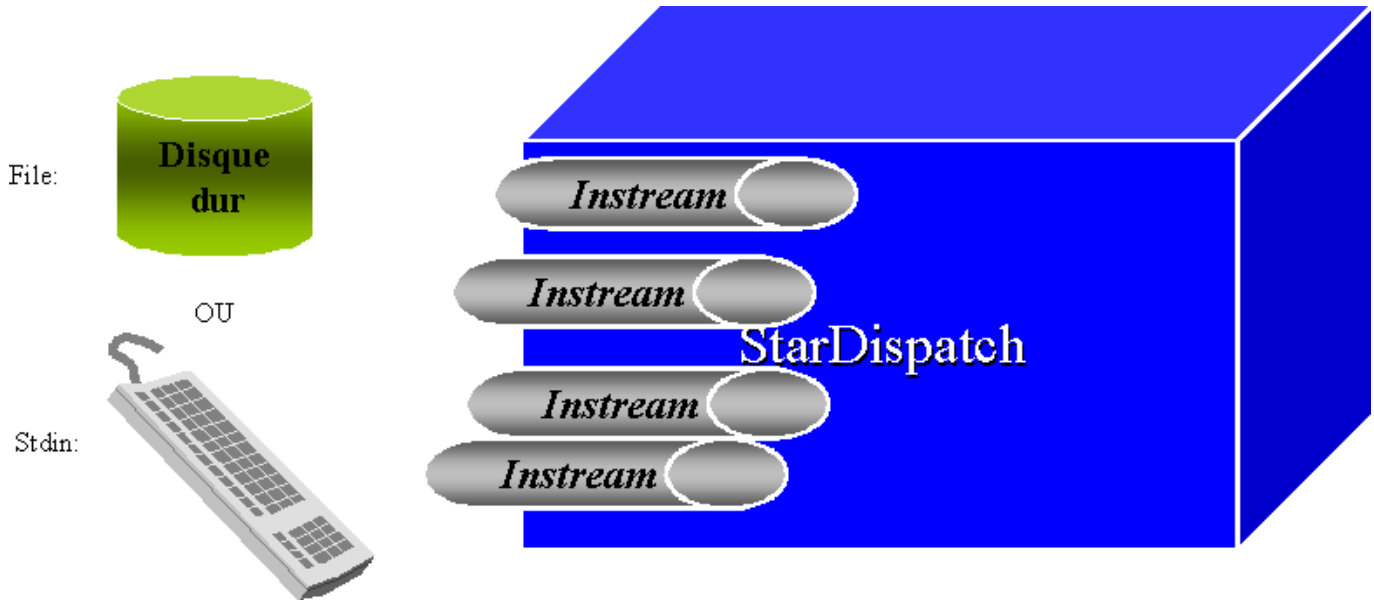
Il existe trois types de Flots :

Les flots externes :

1. Les Flots entrants : Ces flots permettent de transporter les données provenant des fichiers ASCII , des fichiers XML ou, de l'entrée standard (généralement le clavier), vers les unités de traitement de StarDispatch.
2. Les Flots sortants : Ces flots permettent de rendre les données traitées aux destinataires. Ils relient les unités de traitement de StarDispatch aux supports de sauvegarde, aux fichiers temporaires destinés à d'autres applications, ou à la sortie standard (généralement l'écran).
3. Les Flots internes : Ces flots permettent de relier les différentes unités de traitement entre elles. Ils permettent aussi de transporter les entités pages d'une unité de traitement à l'autre.

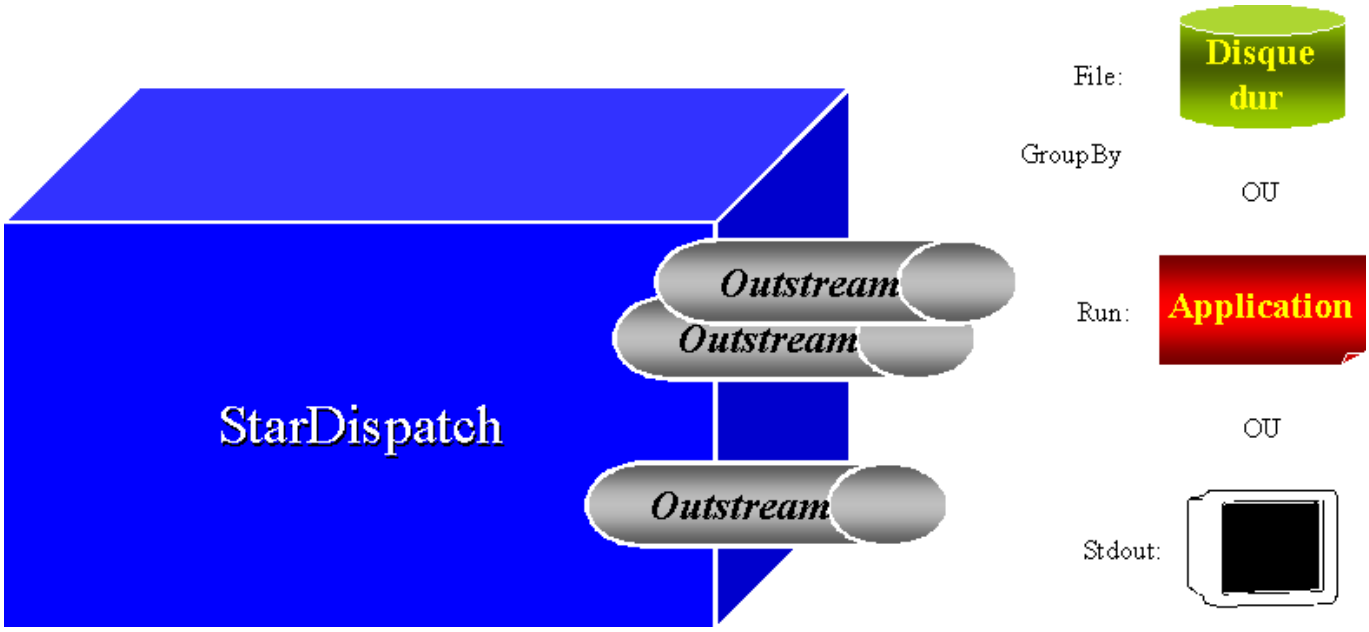
LE FLOT EXTERNE ENTRANT

Un flot externe entrant est un flot permettant le transport des données provenant de fichiers ASCII, de fichiers XML, de fichiers XML ou, d'entrée standard (clavier).



LE FLOT EXTERNE SORTANT

Un flot externe sortant est un flot permettant le transport des données traitées, vers les supports de sauvegarde, les fichiers temporaires destinés à d'autres applications, ou, la sortie standard.



LE FLOT INTERNE

Un flot interne est un flot permettant le transport des entités pages d'une unité de traitement à l'autre.

Boîte 1

Interst

Boîte 2

StarDispatch

LA BOÎTE

C’est l’unité de traitement d’un projet StarDispatch.

StarDispatch dispose de 4 types de boîte sachant que chacune est destinée à effectuer une opération élémentaire.

FEE ou FI

Une boîte de distribution (1 -> n) permet de

réaliser un aiguillage des pages vers des destinations différentes d’après des critères définis.

FEE ou FI

Une boîte de regroupement(n -> 1) permet de réaliser un regroupement de pages correspondant à des critères définis.

FES ou FI

FES ou FI

Boîte de

Regroupement (BM)

Boîte de distribution (BD)

FEE ou FI

Boîte de copie (BC)

Une boîte de copie (1 -> n) permet de réaliser

des duplications de pages .

FES ou FI

FEE ou FI

Boîte de tri (BT)

Une boîte de tri (1 -> 1) permet de réorganiser les pages dans un autre ordre .

FES ou FI

Légende

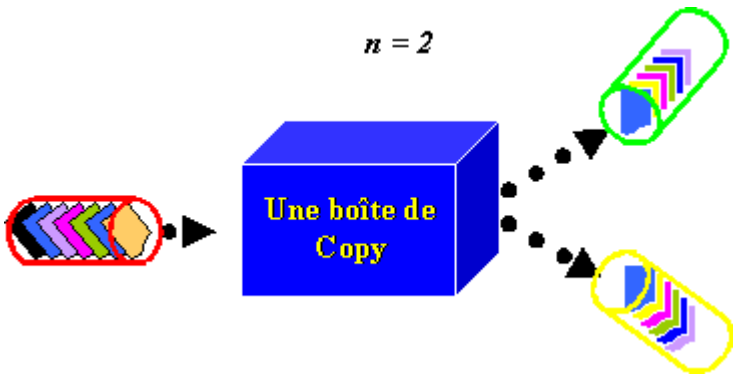
FEE : **F**lot **E**xterne **E**ntrant **FI** : **F**lot **I**nterne

FES : **F**lot **E**xterne **S**ortant Une page peut traverser différentes unités de traitement pendant sa durée de vie.

La boîte de tri ne fonctionne pas avec les fichiers XML.

LA BOÎTE DE COPIE

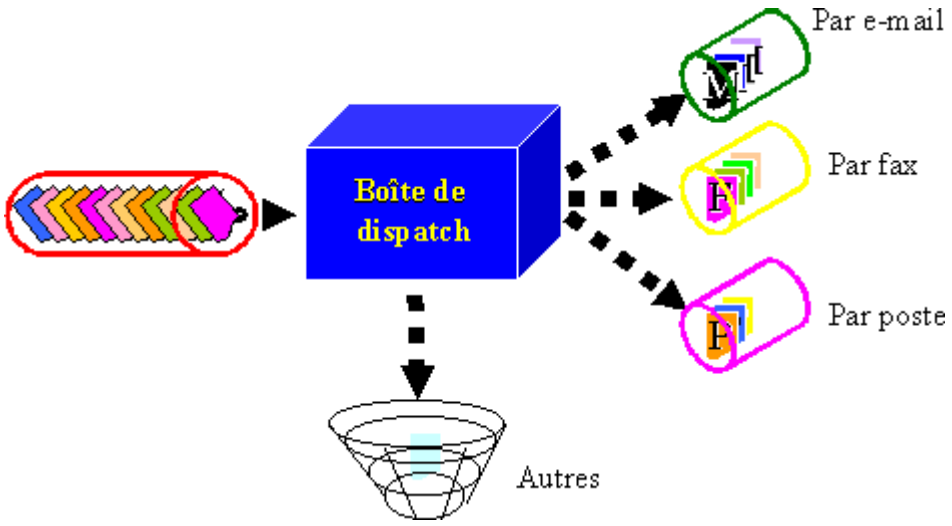
Une boîte de copie est une boîte permettant de copier les pages en n exemplaires.



LA BOÎTE DE DISTRIBUTION

Une boîte de distribution est une boîte permettant la distribution des pages.

Par exemple, ci-dessous, les documents seront envoyés par e-mail, par télécopie ou par courrier postal selon le choix du destinataire.

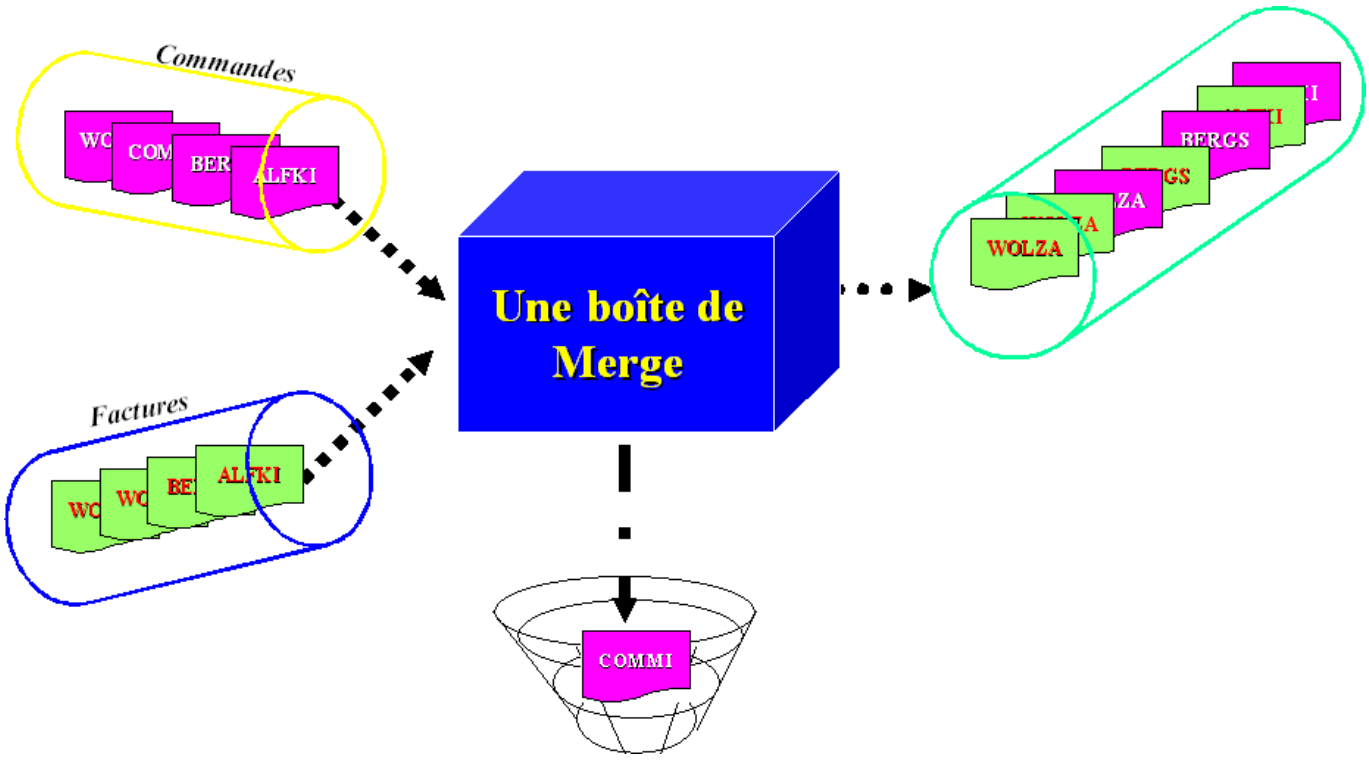


LA BOÎTE DE REGROUPEMENT

Une boîte de regroupement est une boîte permettant le regroupement de pages différentes vers une même destination. Ce regroupement est effectué par rapport à une information existante que l'on appelle "MergeKey".

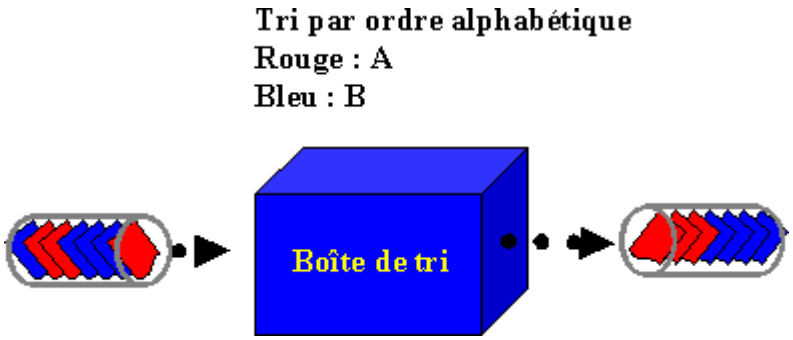
Dans l'exemple ci-dessous, l'utilisateur regroupe commandes et factures selon le tag

Code Client. L'information est de type ASCII et l'ordre de tri ascendant.



LA BOÎTE DE TRI

La boîte de tri est une boîte permettant de trier les pages à partir d'un ou plusieurs critères.



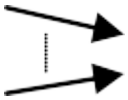
La boîte de tri ne fonctionne pas avec les fichiers XML.

LE PROJET

Un **projet** est l'intégration des différentes notions vues précédemment, conduisant à un ensemble de traitements de duplication, de regroupement et de distribution.

Chaque projet est constitué de données en entrée, de boîtes, de flots et enfin de différentes destinations. Un projet est plus ou moins complexe suivant le nombre de ses composants et la connexion entre ces composants.

Le schéma suivant illustre ce que peut être un projet StarDispatch :



FI

FEE

FES

LE FICHIER DE COMMANDES

Le **fichier de commandes** représente le projet, qui peut être décrit comme l'ensemble des instructions définissant quels fichiers utiliser, quelles sont les opérations à mener à bien, dans quel ordre, et ou distribuer les résultats.

CONTRAINTES ET PRE-REQUIS

StarDispatch fonctionne sous environnement Windows et Unix.

Système d'Exploitation

Système d'exploitation	**Conseillé**	**Obsolète**

Windows	Windows 2000 Pro/Server, Windows XP Pro Windows Server 2003	Windows 98 Windows ME Windows NT 4.0 (fonctionnalités pas toutes disponibles)
Unix	Linux version 32 bits[1] HP-UX, AIX SOLARIS	Linux version 16 bits

Processeur minimum

Système d'exploitation	Processeur minimum requis
Windows & Unix	1. 300 MHz Intel Pentium/Celeron family 2. AMD K6/Athlon/Duron family 3. tout processeur compatible

RAM

Système d'exploitation	Quantité de RAM minimale requise[2]
Windows 98 Windows ME Windows NT 4.0 Unix	64 Mo
Windows 2000 Pro/Server, Windows XP Pro Windows Server 2003	128Mo

Espace disque

Système d'exploitation	Espace disque minimal requis pour l'Application	Espace disque minimal requis pour le Traitement
Windows	3,5 Mo	3 fois la taille du fichier à traiter
Unix	700 Ko	3 fois la taille du fichier à traiter

Résolution

La résolution conseillée est **1024*768**

- Pour une liste exhaustive des distributions Linux compatibles, merci de contacter le support à

l'adresse électronique suivante : support@appic.com.

- Plus la quantité de RAM sera élevée, plus les performances seront importantes.

LES NOTIONS DE BASE

LES VARIABLES

StarDispatch peut utiliser trois familles de variables :

1. Les variables utilisateur sont les variables définies par l'utilisateur dans le cadre d'un projet StarDispatch.

Il en existe trois sortes : **Variable Tag Variable Condition Variable Contexte**

1. **Les variables StarDispatch**
2. **Les variables système**.

LA VARIABLE TAG

Elle est définie par l'utilisateur pour permettre l'accès aux données sur les entités pages (cf § Tag).

Elle peut être de type **String** ou **Date** :

- Une variable Tag de type **String** est une chaîne de caractères (cf § String).
- Une variable Tag de type **Date** correspond à une date formatée sous forme d'une chaîne de caractères (cf § Date).

La valeur d'une variable est référencée par une **Expression** ou un **Field** :

- Une **Expression** correspond à l'adresse de la valeur d'une variable sur une page de données ASCII ou, à un chemin d'accès permettant d'extraire des données dans une page de données XML ASCII ou, à un chemin d'accès permettant d'extraire des données dans une page de données XML (cf § Expression).
- Un **Field** est un mot clé permettant d'accéder à la valeur d'une variable sur une page de données (cf § Field)).

 En conclusion, 4 types de variables tags sont disponibles :

1. Variable String définie par une Expression,
2. Variable String définie par un Field,
3. Variable Date définie par une Expression,
4. Variable Date définie par un Field.

La valeur d'une variable est calculée définitivement, lors de la création de la page qui la contient.

Quatre variables tags **FILEPATH**, **FILENAME**, **FILEEXTENSION** et **DataLinesNumber** sont prédéfinies pour chaque fichier en entrée c'est à dire qu'il est nul besoin de les associer ni de les définir dans les flots entrants. Les 3 premières variables permettent la récupération du nom du fichier dans un tag et ceci, afin de l'utiliser dans la définition d'une sortie. Cette fonctionnalité est utile dans le cas où le nom du fichier de données est

significatif et porte des informations. La dernière variable, **DataLinesNumber** permet, quant à elle, de récupérer le nombre de lignes lues dans la page.

LA VARIABLE CONDITION

Il est possible de définir une variable pour nommer une expression booléenne trop longue, et, afin de faciliter son rappel (cf § condition).

* Sa valeur est calculée au moment de son utilisation.

LA VARIABLE CONTEXTE

La variable contexte permet de définir le contexte de travail de StarDispatch. L'utilisateur peut définir ces variables au besoin sur la ligne de commande et/ou dans le fichier de commandes.

Le nombre maximal de variables contexte pouvant être définies dans un projet est de 32.

StarDispatch évalue la valeur de cette variable dans l'ordre suivant :

1. Sur la ligne de commande de StarDispatch.
2. Dans le fichier de commandes de StarDispatch (cf § fichier de commandes).
3. Dans l'environnement du système d'exploitation.

LES VARIABLES STARDISPATCH

StarDispatch possède un certain nombre de variables propriétaires mises à la disposition de l'utilisateur pour réaliser des opérations spécifiques et dont certaines peuvent être initialisées (cf § liste des variables StarDispatch).

LES VARIABLES SYSTEME

Ce sont des variables reconnues et utilisées par le système d'exploitation. En absence d'une définition de la part de l'utilisateur concernant certaines variables, StarDispatch utilise les valeurs définies par le système d'exploitation.

LES OPERATEURS

Ils sont de 3 types :

1. Les opérateurs booléens (OpBool)
2. Les opérateurs numériques (OpNum)
3. Les opérateurs ASCII.

LES OPERATEURS BOOLEENS (OpBool)

Les opérateurs booléens reconnus et supportés dans une expression booléenne

sont les suivants :

- AND
- OR (inclusif)
- NOT

LES OPERATEURS NUMERIQUES (OpNum)

Les opérateurs booléens sur les valeurs numériques reconnus et supportés dans une expression booléenne sont les suivants :

- %= Egalité.
- %<, %<= Inférieur, inférieur ou égal.
- %>, %>= Supérieur, supérieur ou égal.
- %<> Différent de.

LES OPERATEURS ASCII

Les opérateurs ASCII reconnus et supportés dans une expression booléenne sont les suivants :

 = Egalité.

- <, <= Inférieur, inférieur ou égal.
- >, >= Supérieur, supérieur ou égal.
- <> Différent de.

L'EXPRESSION LIGNE COLONNE (ExpLC)

Cette expression, réservée aux flots de type ASCII,, réservée aux flots de type ASCII, permet de désigner une zone sur une page par un coin supérieur gauche et un coin inférieur droit. Chaque coin correspond à un couple ligne – colonne.

Tout au long de ce manuel, ce couple est indiqué sous la forme **Ll:Cc,c**. Le premier élément de la matrice est **L1:C1,1.** Par exemple, on peut référencer les 20 premières lignes par **L1,20**. Cependant, cette possibilité est rarement utilisée car, la plupart des opérations se déroulent sur une ligne.

Cette expression est nommée par la suite ExpLC.

L'EXPRESSION STRING

Cette expression, réservée aux flots de type ASCII, , réservée aux flots de type Apermet de définir le contenu d'une zone (ExpLC) sur une page de données ASCII. Cette expression est nommée par la suite ExpStr.

Fonctions

Des fonctions sont également disponibles pour permettre de tester les données contenues dans une zone. Ces fonctions portent sur le contenu alphanumérique des informations, sans les modifier sur la page d'entrée.

- **length (**ExpStr**) :** Cette fonction renvoie la longueur de ExpStr.
- **toupper (**ExpStr**) :** Cette fonction convertit les caractères de ExpStr en majuscules.
- **tolower (**ExpStr**) :** Cette fonction convertit les caractères de ExpStr en minuscules.
- **trim (**ExpStr**) :** Cette fonction supprime les blancs en début et en fin de ExpStr.
- **blank (**ExpStr**) :** Cette fonction renvoie une valeur vraie si ExpStr est à blanc.
- **left (**ExpStr, L**) :** Cette fonction permet d'extraire la partie gauche de l'ExpStr, sur la longueur L.
- Exemple :

Appel	**Résultat**
left ("Hello", 3)	"Hel"

- **right (**ExpStr, L**) :** Cette fonction permet d'extraire la partie droite de l'ExpStr, sur la longueur L.
- Exemple :

Appel	**Résultat**
right ("Hello", 3)	"llo"

- **format (**ExpStr, format [décimal, séparateur]**) :** Cette fonction formate ExpStr, selon **format**, spécifiant le caractère décimal et la présence ou non du séparateur de millier.
- Exemple :

ExpStr	**Format spécifié**	**Résultat**
56123,567	""	56123,567
56123,567	"# ###,##"	56 123,57
56123,567	"# ###,####"	56 123,567
56123,567	"#.###,0000"	56.123,5670
56123,567	"#,###.000"	56,123.567

- **alignright (**ExpStr, nombre, [remplissage]**) :** Cette fonction permet d'aligner à droite une ExpStr sur un espace défini par une valeur (nombre). Les blancs peuvent être remplacés par des caractères de remplissage spécifiés entre guillemets.
- Exemple :

Appel	**Résultat**
Alignright ("Hello", 10, '.')	" Hello"
Alignright ("Hello", 10)	" Hello"
Alignright ("Hello", 2, '.')	"Hello"

- **sum (**ExpLC**) :** Cette fonction effectue la somme des valeurs numériques contenues sur la colonne définie par une ExpLC.
- Exemple : Ligne 1 : *14* Ligne 2 : *15* Ligne 3 : *09* Ligne 4 : *12* Ligne 5 : *04*

Appel	**Résultat**
Sum(L1,5:C2,3)	54

NB : **sum** est la seule fonction qui accepte plusieurs lignes dans la définition de l'ExpLC. Toutes les autres fonctions citées plus haut traitent l'ExpStr sur une seule ligne.

Opérateur de concaténation (+)

- Exemple :

Appel	**Résultat**
"TO" + "to"	"TOto"
"TO" + toupper("to")	"TOTO"

Opérateurs numériques

Voici la liste des opérateurs numériques valables sur ExpStr :

{%+ , %- , %* , %/ , %mod , %div}

- Exemple :

Appel	**Résultat**
"14" **%+** "12"	26
"to" **%+** "1"	1
"to" **%+** "to"	0
"14" **%-** "10"	4
"7" **%*** "10"	70
"56" **%/** "8"	7
"27" **%mod** "3"	0
"27" **%div** "6"	4

Une Expression String permet aussi de manipuler le contenu d'une variable d'environnement (↔ variable globale au système).

Pour être utilisée dans une expression, une variable d'environnement doit être précédée d'un point d'exclamation !.

- Exemple :

toupper(!username)

L'EXPRESSION XPATH

L'expression XPath, réservée aux flots de type XML permet, à partir de

chemins d'accès permettant de sélectionner les nœuds[1], d'extraire les données d'un fichierXML.

A l'instar des Expressions String, les fonctions portant sur le contenu alphanumérique des informations sont disponibles pour permettre de tester les données contenues dans un fichier XML, ainsi que les opérateurs de concaténation et les opérateurs numériques (cf § L'expression String).

Cette expression est nommée par la suite ExpXPath.

[1] Un fichier XML peut être considéré comme une arborescence de nœuds. Il y en a différents types :

1- **Nœud document** ou **nœud racine** ( ) 2- **Nœud élément** ( )

3- **Nœud attribut** ( ) 4- **Nœud texte** ( )

```
<?xml version="1.0" encoding="iso-8859-1" ?>
- <bookstore>
  - <book>
      <title lang="eng">Around the world in eighty days</title>
      <author>Jules Vernes</author>
      <year>1962</year>
    </book>
  </bookstore>
```

L'EXPRESSION BOOLEENNE

Cette expression permet de définir les expressions booléennes de StarDispatch. Une expression booléenne peut être construite de la façon suivante :

- Syntaxe :

Une **ExpBool** est une (**ExpBool**) : Expression Booléenne simple ou une expression composée :

ExpBool OpBool ExpBool

ou **ExpLC OpNum** <num>

ou **ExpXPath OpNum** <num>

ou **ExpLC OpNum ExpLC**

ou **ExpXPath OpNum ExpXPath**

ou **ExpXPath OpNum ExpXPath**

ou Une_Variable_tag **OpNum** <num>

Un **OpBool** est une AND ou OR

ou NOT

Un **OpNum** est une %= ou %<

ou %<=

ou %>

ou %>=

ou %<>

Une **ExpLC** est une L<num>,<num>:C<num>,<num> ou

Une **ExpXPath**est une XPath(XpathExpStr)

ou XPath(XpathExpStr).Value

ou XPath(XpathExpStr).Count

ou XPath(XpathExpStr).Name

- Exemple :

Cet exemple représente la construction d’une expression booléenne :

(ExpBool)

ExpBoolOpBoolExpBool

(ExpBool)

AND

L2:C20,25

%<=

70 000

L1:C12,15

%=

L10:C72,75

La deuxième ligne de l’expression booléenne correspond à : (L2:C20,25 %<= 70 000) AND (L1:C12,15 %= L10:C72,75)

ExpLC OpNum <num>

ExpLC OpNum ExpLC

Dans cet exemple, le contenu de la zone L2:C20,25 est comparé avec la valeur 70 000 (vrai ou faux). Ensuite, le contenu de la zone L1:C12,15 est comparé avec le contenu de la zone L10:C72,75 (vrai ou faux). Enfin, une opération booléenne AND sera réalisée entre ces deux dernières comparaisons.

Par la suite, cette expression est nommée ExpBool.

LES OBJETS STARDISPATCH

Ce chapitre contient la description de deux grandes catégories d'objets StarDispatch :

1. les flots,
2. les boîtes.

Avant de décrire ces objets en détail, il est nécessaire d'introduire quelques définitions fondamentales.

La déclaration d'un flot consiste :

- à lui attribuer un identifiant logique,
- à définir les propriétés concernant le découpage et le groupement des pages,
- en la définition des variables tags.

La déclaration d'une boîte consiste :

- à lui attribuer un identifiant,
- à lui attribuer un nombre de sorties et/ou d'entrées
- à définir les sorties et/ou entrées

et, dans le cas d'une boîte de regroupement, à définir

- l'information de traitement (définie par une ExpStr)
- l'ordre de tri (ascendant ou descendant)
- le type d'information (ex : entier)

L'appel ou, autrement dit, l'utilisation d'une boîte consiste en l'association des flots déclarés à la boîte.

LES FLOTS

La déclaration d'un flot consiste :

- à lui attribuer un identifiant logique,
- à définir les propriétés concernant le découpage et le groupement des pages,
- en la définition des variables tags.

Les flots sont des entités logiques de StarDispatch permettant la connexion et le transport des données entre les composants d'un projet StarDispatch.

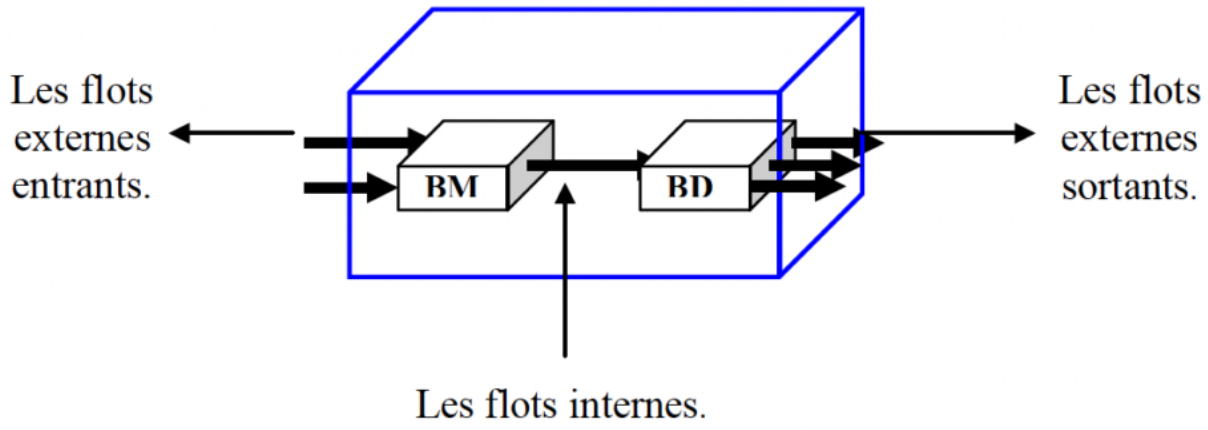
On peut distinguer deux grandes catégories de flots :

1. Les flots externes,
2. Les flots internes.

LES FLOTS EXTERNES

Les flots externes sont des flots situés aux extrémités du système de traitement. On peut distinguer deux types de flots externes :

1. Les flots entrants.
2. Les flots sortants.



Un flot externe entrant est associé à une boîte destination, de la même manière qu'un flot externe sortant est associé à une boîte d'origine. Par contre, un flot interne peut être le flot sortant de sa boîte d'origine et le flot entrant de sa boîte destination.

FLOTS EXTERNES ENTRANTS

Les flots externes entrants sont des flots se trouvant à l'entrée du système. Ces flots raccordent les différentes sources externes (les fichiers ASCII, les fichiers XML, les fichiers XML, clavier…) au système StarDispatch. Ces flots permettent de transporter les entités pages (cf § La page) provenant des sources externes vers les unités de traitement internes.

- Syntaxe de déclaration (Flot entrant de type ASCII) (Flot entrant de type ASCII)

instream $Input_1$[, $Input_2$,…, $Input_n$] [**maxlines** = ##]

[**ejectcode** = Car]

[**page** = Ll :Cc,c = ” $String_1$” ……]

[**usedtag** <$nom\_variable\_tag_1$> **is** [**String** | **Date** (“format“)] as [**Field** | **Expression**(1)(1) | **SourcePageNumber**]

[……]

endstream

[1]L'[1]L'expression est de type ExpStr.

instream .. endstream : mots réservés permettant d'encadrer la déclaration d'un flot entrant.

Input$_i$: nom attribué à un flot entrant par l'utilisateur.

maxlines (réservé au flot de type ASCII): le nombre maximum de lignes dans une entité page (cf § maxlines).

ejectcode (réservé au flot de type ASCII) : le code de saut de page (cf § ejectcode).

page (réservé au flot de type ASCII) : la condition de rupture des entités pages (cf § Page).

usedtag : l'initialisation des variables tags (cf § usedtag).

- Exemple

instream flot_in

maxlines = 40

ejectcode = 12

page = L1:C8,15 = "SOCIETE :"

usedtag *NumClient* **Is string as field** "Client :" **usedtag** *CodePostal* **Is string as expression** L3:C90,94 **usedtag** *BonlivraisonDate* **Is date** ("DDMMYYYY") **as field** "Date :"

usedtag *FactureDate* **is date** ("DDMMYYYY") **as expression**

L2:C70,77

endstream

Dans cet exemple, nous avons déclaré un flot entrant, flot_in. Il sera associé plus tard à un fichier contenant des données ASCII.

Les trois premières lignes décrivent des informations concernant le découpage des entités pages comme suit :

1ère ligne : indique que les données de la source seront découpées toutes les 40 lignes (maximum).

2ème ligne : indique que les pages de données sont séparées par un caractère de saut de page (12 = '\f').

3ème ligne : définit une condition de rupture (cf § Page) Chaque occurrence de la chaîne "SOCIETE :" positionnée en C8,16, déclenchera la création d'une nouvelle entité page de sorte que la chaîne "SOCIETE :" se trouve à la position L1:C8,16.

Les quatre lignes suivantes permettent d'initialiser les variables tags des entités pages du flot "flot_in". Cet exemple représente les quatre types de variables tags de StarDispatch.

La variable tag **NumClient** est de type String défini par un Field (cf § Field) Cette définition indique à StarDispatch qu'il doit chercher la valeur de cette variable après le mot "Client :" sur l'entité page.

La variable tag **CodePostal** est de type String défini par une Expression (cf § Expression). Cette définition indique à StarDispatch qu'il doit chercher la valeur de cette variable sur la troisième ligne, de la colonne 90 à la colonne 94 incluse.

La variable tag **BonlivraisonDate** est de type Date défini par un Field (cf § Date)

Cette définition indique à StarDispatch qu'il doit chercher la date après le mot "Date

:" sur l'entité page et qu'elle est formatée comme suit "DDMMYYYY".

Enfin, la variable tag **FactureDate** est de type Date défini par une ExpStr (cf § Date) Cette définition indique à StarDispatch qu'il doit chercher la date sur la deuxième ligne de la page, de la colonne 70 à la colonne 77 incluse, et qu'elle est formatée comme suit : "DDMMYYYY".

Il est possible de déclarer plusieurs flots entrants possédant les mêmes propriétés. Dans ce cas, une liste de noms sera associée à la déclaration :

InStream nom_1, nom_2,…,nom_x

…

EndStream

- Syntaxe de déclaration (Flot entrant de type XML)

instream $Input_1$[, $Input_2$,…, $Input_n$] [**SearchedNodeLevel = ##**]

[**usedtag** <$nom\_variable\_tag_1$> **is** [**String**] as[**Expression**(1)]

[……] endstream

1. L'(1)L'expression est de type ExpXPath.

instream .. endstream : mots réservés permettant d'encadrer la déclaration d'un flotentrant.

Inputi : nom attribué à un flot entrant par l'utilisateur. **searchedNodeLevel** : le niveau de découpage d'un fichier XML. **usedtag** : l'initialisation des

variables tags

- Exemple

instream NouveauFlotEntrant001

SearchedNodeLevel = 3

usedtag Num_Fact **is String As Expression XPath**

(“/*/*/*”).Name

usedtag Attribute **is String As Expression XPath**

(“/*/*/*”).Value

endstream

FLOTS EXTERNES SORTANTS

Les flots externes sortants sont des flots se trouvant à la sortie du système. Chaque flot de sortie est dédié à un objet externe et transporte donc les données concernant cet objet. Les objets externes peuvent être des fichiers, des sorties standards ou des appels d’applications.

- Syntaxe de déclaration (Flot sortant de type ASCII)

outstream Output1[, Output2,…, Outputn] [**maxlines =** <num> [**full**]] [**ejectcode =** Car]

[**GroupBy =** tag1, [tag2,…,tagn]] ou [**Maxpages =**

<num>]

[**PackBy =** <num>]

endstream

outstream .. endstream : Les mots réservés permettant d’encadrer la déclaration d’un flot sortant.

Output$_i$: le nom attribué à un flot sortant par l’utilisateur.

maxlines : le nombre maximum de lignes (cf § maxlines).

ejectcode : le code de saut de page (cf § ejectcode).

GroupBy : la déclaration de variables tags pour la création des documents (cf § GroupBy).

Maxpages : la déclaration d’un nombre défini de pages pour la création des documents (cf § Maxpages)

PackBy : le nombre de groupes de documents (cf § GroupBy).

- Exemple (Flot sortant de type ASCII)

outstream flot_out **maxlines =** 40 **full ejectcode =** 12

GroupBy = *Adrs_mail, No_Clt ou* **Maxpages =** *5*

PackBy = 50

endstream

Dans cet exemple nous avons déclaré un flot sortant “flot_out”.

1ère ligne : indique que les données dans les entités pages de StarDispatch seront écrites sur un maximum de 40 lignes. L’option **full** permet de compléter une page par des lignes vides dans le cas où le nombre de lignes de données d’une entité page de StarDispatch serait inférieur à 40.

2ème ligne : indique que les pages seront séparées par un caractère de saut de page (12

= ‘FF’).

GroupBy indique que les pages de ce flot vont être regroupées dans des documents. Chaque document contiendra des pages possédant les mêmes informations concernant l’adresse mail (définie par la variable tag **Adrs_mail**) et le numéro client (défini par la variable tag **No_Clt**). Chaque document possède un numéro accessible par la variable de contexte **DocNr**.

Maxpages indique que les pages de ce flot vont être regroupées dans des documents, sachant qu’un document sera créé toutes les 5 pages. Chaque document possède un numéro accessible par la variable de contexte **DocNr**.

PackBy indique que ces documents seront regroupés à leur tour par paquets de 50. Chaque paquet de documents possède un numéro de groupe. Ce numéro est accessible par la variable de contexte **GroupNr**.

Il est possible de déclarer plusieurs flots sortants possédant les mêmes propriétés.

Dans ce cas une liste de noms sera associée à la déclaration :

OutStream nom_1, nom_2,…,nom_x

…

EndStream

- Syntaxe de déclaration (Flot sortant de type XML)

outstream Output1[, Output2,…, Outputn] [**Maxlines =** -1]]

[**Ejectcode =** -1]

[**XMLLayout RootNode** “Nom de la balise globale”

Attributes=“Expression String”]

[**SubNode** “Nom de la balise” **Attributes=**“Expression String”]

When Condition

Breakvalue=Expression String

EndXMLLayout endstream

outstream .. endstream : Les mots réservés permettant d’encadrer la déclaration d’un flot sortant.

Output$_i$: le nom attribué à un flot sortant par l’utilisateur.

maxlines : l’instruction, réservée au mode ASCII, est désactivée en lui associant la valeur –1 (cf § maxlines).

ejectcode : le code de saut de page, réservé au mode ASCII, est désactivé en luiassociant la valeur –1 (cf § ejectcode).

XMLLayout .. EndXMLLayout : mots réservés permettant d’encadrer la définition d’une structure XML.

RootNode : le nom attribué à la balise globale

Attributes : l’attribut est une Expression String plus ou moins complexe.

SubNode : le nom attribué à la sous-balise.

When : Associe une condition.

Breakvalue : le critère de rupture.

Les attributs et noms ne doivent pas présenter d’espaces et doivent être entourées de quotes.

- Exemple (Flot sortant de type XML)

outstream flot_out

maxlines = -1

ejectcode = –1

XMLLayout RootNode “ENGLOBE_TOTALE” **Attributes=”” SubNode** “Englobe_Caratula” **Attributes=”” When** TRUE **BreakValue=**Attribut

EndXMLLayout endstream

FLOTS INTERNES

Les flots internes peuvent être définis comme tout flot de pages présent à l'intérieur du système. Ces flots permettent de relier les différentes unités de traitement de StarDispatch entre elles, et de garantir le transport des entités pages à l'intérieur du système.

- Syntaxe de déclaration

interstream Interflot1 [, Interflot2, …, Interflotn]

…

endstream

interstream .. endstream : Les mots réservés permettant d'encadrer la déclaration des flots internes.

Interflot$_i$: le nom attribué à un flot interne par l'utilisateur.

LES BOITES

Les boîtes sont les unités de traitement de StarDispatch. Une boîte reçoit des données en entrée par un flot. Elle les traite suivant sa fonction et ses instructions. Elle procède à l'aiguillage des données vers une nouvelle destination. Il existe quatre types de boîte :

1. Boîte de type distribution des données (**BD**)
2. Boîte de type regroupement des données (**BM**)
3. Boîte de type copie des données (**BC**)
4. Boîte de type tri des pages (**BT**)

Dans le cadre d'un projet, il est possible d'utiliser plusieurs boîtes de même type avec des instructions différentes. Par exemple, une **BD** avec 3 sorties et une autre **BD** avec 5 sorties et une autre **BD** avec 15 sorties. Dans ce cas, il faut déclarer toutes ces boîtes différentes.

De la même manière, il est possible d'utiliser la même boîte plusieurs fois. Par exemple, une boîte **BDR**égion**P**arisienne avec 7 sorties (75, 78, 91, 92, 93, 94, 95), peut être utilisée pour éclater notre fichier "clients", notre fichier "bons de livraisons" et notre fichier "bons de commandes". Dans ce cas, une seule déclaration suffit, et à chaque utilisation on précisera les entrées et les sorties correspondantes.

L'utilisation d'une boîte est précédée obligatoirement par sa déclaration.

LA BOÎTE DE DISTRIBUTION

La boîte de distribution permet de distribuer les entités pages vers une ou

plusieurs destinations. L'utilisateur détermine le nombre de sorties de la boîte et associe une expression booléenne à chacune des sorties.

Lorsqu'une entité page se présente, chaque sortie évalue son expression booléenne. Dans le cas où cette évaluation serait satisfaisante, l'entité page est récupérée par la sortie, sinon, elle est envoyée vers la sortie suivante.

- Syntaxe de déclaration (en mode ASCII)

dispatch BoxName **outputs =** $output_1$ [, $output_2$…] $output_1$ **when** $ExpBool_1$

[$output_2$ **when** $ExpBool_2$]

…

enddispatch

dispatch .. enddispatch : Les mots réservés permettant d'encadrer la déclaration d'une boîte de distribution.

BoxName : Le nom attribué par l'utilisateur. Ce nom permet de rappeler cette boîte.

outputs : La liste des sorties de cette boîte.

$Output_k$: Désigne la $K^{ième}$ sortie de la boîte.

When : Associe une condition à une sortie (cf § When).

- Exemple

condition Paris is L2:C91,92 %= 75

……..

dispatch dispatchRegion **outputs =** $Output_1$, $Output_2$, $Output_3$ $Output_1$ **when** L2:C91,92 %= 45

$Output_2$ **when** L2:C91,92 %> 45 AND L2:C91,92 %< 75

$Output_3$ **when** Paris

enddispatch

Cet exemple représente la déclaration d'une boîte de distribution avec trois sorties. Cette boîte est nommée "dispatchRegion" et chacune de ses sorties est associée à une condition ou une expression booléenne.

La sortie $Output_1$ est désignée pour contenir toutes les pages ayant le chiffre 45 sur la deuxième ligne, de la colonne 91 à la colonne 92 incluse.

La sortie $Output_2$ est désignée pour contenir toutes les pages ayant sur la

deuxième ligne, de la colonne 91 à la colonne 92 incluse, un chiffre supérieur à 45 et inférieur à 75.

La sortie $Output_3$ est désignée pour contenir toutes les pages vérifiant la condition

Paris définie précédemment (cf § condition).

Il existe une sortie implicite vers laquelle toutes les pages ne correspondant à aucun critère de traitement sont envoyées : la sortie de rejet. Le flot alimentant cette sortie est commun à toutes les boîtes.

☺ Il existe une astuce pour récupérer les données envoyées en rejet, afin, par exemple, de les réinjecter dans une autre boîte. Pour cela, on rajoute dans l'exemple une sortie $output_4$, comme ci-dessous :

condition *Paris* is L2:C91,92 %= 75

……..

dispatch dispatchRegion **outputs =** $Output_1$, $Output_2$, $Output_3$ $Output_1$ **when** L2:C91,92 %= 45

$Output_2$ **when** L2:C91,92 %> 45 AND L2:C91,92 %< 75

$Output_3$ **when** Paris $Output_4$ **when** true

enddispatch

La sortie rejet, implicite, ne contient plus aucune donnée.

Les boîtes de distribution sont appelées de la manière suivante :

- Syntaxe d'appel

BoxName (Input_Flot; $Output\_Flot_0$ [, $Output\_Flot_1$…])

- Exemple

dispatchRegion (Flot_in; Flot_45, Flot_reste, Flot_Paris)

    1. Chaque sortie est associée à une expression booléenne.
    2. L'ordre de déclaration de ces associations détermine leur priorité.
- Syntaxe de déclaration (en XML mode)

dispatch BoxName **outputs =** $output_1$ [, $output_2$…] $output_1$ **when** $XPathExp_1$

[$output_2$ **when** $XPathExp_2$]

…

enddispatch

LA BOÎTE DE REGROUPEMENT

La boîte de regroupement permet de regrouper les entités pages vers une destination. Ce regroupement est effectué par rapport à une information existante sur les entités pages, que l'on appelle par la suite "MergeKey".

L'utilisateur détermine le nombre d'entrées et définit les propriétés de la boîte de regroupement, en fournissant des informations suivantes : l'ordre de tri des entrées, le type de MergeKey, la synchronisation (pages regroupées si et seulement si la "MergeKey" concorde pour les différents flots) ou non des entrées, et enfin, la zone où se trouve MergeKey sur chaque entrée.

L'arbitre (cf § Select) explore les entrées de la boîte et désigne (en utilisant les propriétés ci-dessus) celles qui possèdent des données à expédier vers la destination. Après avoir envoyé ces données, l'arbitre reprend les mêmes opérations jusqu'à l'épuisement complet des données sur toutes les entrées.

- Syntaxe de déclaration (en mode ASCII)

merge BoxName **inputs =** $input_1$ [, $input_2$, …,$input_n$] **Select** [ascent | descent][ascii | number | date][synchronized]

$input_1$ **on** $ExpStr_1$

[$input_2$ **on** $ExpStr_2$]

.

.

endmerge

merge .. endmerge : Les mots réservés permettant d'encadrer la déclaration d'une boîte de regroupement.

BoxName : L'identifiant attribué à la boîte par l'utilisateur. Cet identifiant permet de faire appel à cette boîte.

inputs : La liste des entrées de cette boîte.

$input_k$: désigne la $k^{ième}$ entrée de la boîte

Select : Contient des propriétés de la boîte de regroupement (cf § Select).

On : associe une ExpStr à une entrée

- Exemple

merge mergeRegion **inputs =** $MyInput_1$, $MyInput_2$, $MyInput_3$

Select ascent number $MyInput_1$ **on** L2:C90,94 $MyInput_2$ **on** L4:C70,74 $MyInput_3$ **on**

L10:C60,64

endmerge

Cet exemple représente la déclaration d'une boîte de regroupement nommée "mergeRegion" ayant trois entrées. Cette boîte va regrouper les trois entrées en ordre ascendant selon une information de type numérique.

L'arbitre de cette boîte est défini comme suivant :

- Les entrées sont triées dans l'ordre ascendant.
- Le MergeKey est de type numérique.
- Le MergeKey se trouve dans la zone L2:C90,94 sur $MyInput_1$, dans la zone L4:C70,74 sur $MyInput_2$ et dans la zone L10:C60,64 sur $MyInput_3$.

Il existe une sortie implicite vers laquelle toutes les pages ne correspondant à aucun critère de traitement sont envoyées : la sortie de rejet.

Dans cet exemple, les flots sont implicitement non synchrones c'est à dire que toutes les pages sont regroupées même si elles ne sont pas valides sur toutes les entrées d'une boîte de regroupement.

Les boîtes de regroupement sont appelées de la manière suivante :

- Syntaxe d'appel

BoxName (Input_Flot$_0$ [,Intput_Flot$_1$…]; Output_Flot)

- Exemple

mergeRegion (Flot_r1, Flot_r2, Flot_r3; Flot_out)

1. L'ordre de déclaration des associations entrée-ExpStr détermine leur priorité.
2. L'ordre de tri est le même pour toutes les entrées d'une boîte de regroupement.

- Syntaxe de déclaration (en mode XML)

merge BoxName **inputs =** input$_1$ [, input$_2$, …,input$_n$] **Select** [ascent | descent][ascii | number | date][synchronized]

input$_1$ **on** XPathExp$_1$ [input$_2$ **on** XPathExp$_2$]

.

.

endmerge

LA BOÎTE DE COPIE

La boîte de copie permet de copier les données en entrée. Cette boîte ne

traite pas les données mais réalise des copies brutes. L'utilisateur indique une sortie par exemplaire souhaité.

- Syntaxe de déclaration

copy BoxName **outputs =** $output_1$ [**,** $output_2$ **…**, $output_n$] [**Recursive = TRUE**]

…

endcopy

copy .. endcopy : Les mots réservés permettant d'encadrer la déclaration d'une boîte de copie.

BoxName : L'identifiant attribué à la boîte par l'utilisateur. Cet identifiant permet de faire appel à cette boîte.

outputs : Liste contenant les sorties de la boîte.

Output$_k$: Désigne la $k^{ième}$ sortie de la boîte.

Recursive = TRUE : La fonction de lecture récursive est activée. Cette fonction est utile dans le cas ou, par exemple, le nombre de copies désiré est inconnu et lié au nombre de pages d'un autre fichier. La boîte de copie doit donc directement être lié au flot entrant.

- Exemple

copy copyRegion **outputs**=ident1, ident2, ident3

…

endcopy

Cet exemple représente la déclaration d'une boîte de copie avec trois sorties. Cette boîte est nommée "copyRegion" et permet de créer trois exemplaires des données en entrée.

Les boîtes de copie sont appelées de la manière suivante :

- Syntaxe d'appel

BoxName (Input_Flot; $Output\_Flot_0$ [, $Output\_Flot_1$**…**])

- Exemple

copyRegion (Flot_in**;** Flot_Out1, Flot_Out2, Flot_Out3)

LA BOÎTE DE TRI

La boîte de tri permet de trier les pages d'un fichier de type ASCII et non XML suivant certain(s) critère(s). Ce tri est réalisé à partir de la valeur d'une zone dans la page.

La boîte de tri accepte un fichier en entrée et un fichier en sortie.

L'utilisateur détermine le nombre de critères de tri et définit les propriétés de la boîte de tri, en fournissant les informations suivantes : l'ordre des critères, le sens du tri (ascendant ou descendant), et enfin, la zone à évaluer et à trier pour chaque critère.

- Syntaxe de déclaration

sort BoxName

[ascent|descent]**On**[date(yyy/mm/dd)]**As**[Expression<Exp Str>|Field<Expression>]
[ascent|descent]**On**[String<ExpStr>]**As**[Expression<ExpS tr>| Field <Expression>]

[ascent | descent] **On** [Tag] [<Ident>]

endsort

sort .. endsort : Les mots réservés permettant d'encadrer la déclaration d'une boîte de tri.

BoxName : L'identifiant attribué à la boîte par l'utilisateur. Cet identifiant permet de faire appel à cette boîte.

On : associe le type de tri à la zone à évaluer.

As : Définit la manière de retrouver la zone de tri dans la page. Deux possibilités sont permises : Expression ou Field.

- Exemple

tag CodePostal

sort sortRegion

Ascent **on** tag CodePostal

Ascent **on** String **as** Expression L2:C90,104

endsort

Cet exemple représente la déclaration d'une boîte de tri nommée "sortRegion" ayant 2 critères. Cette boîte va trier les pages d'abord par CodePostal en ordre ascendant puis, pour chaque code postal, le tri va également s'effectuer en ordre ascendant par numéro de sécurité sociale et sur 15 positions.

Les boîtes de tri sont appelées de la manière suivante :

- Syntaxe d'appel

BoxName (Input_Flot; Output_Flot)

- Exemple

sortRegion (Flot_r; Flot_out)

1. L’ordre de déclaration des critères détermine leur priorité.
2. Le sens du tri peut être différent à chaque critère de la boîte de tri.

ASSOCIATION DES FLOTS LOGIQUES

Cette phase consiste à associer des flots externes logiques à des variables. Cette association, obligatoire, permet de connecter un ou plusieurs flots à un objet physique. Il est, en effet, possible, comme le montre l’exemple suivant, de définir une seule variable d’association et de l’utiliser pour associer plusieurs flots. Chacun de ces flots aura donc la définition de la variable.

Ces variables seront exploitées, dans le contexte d’exécution, pour définir les données physiques en entrée et en sortie.

- Syntaxe

Association :

$nom\_logique\_flot_1$= $variable\_association\_flot_1$ $nom\_logique\_flot_2$= $variable\_association\_flot_2$

.

.

$nom\_logique\_flot_n$= $variable\_association\_flot_n$

- Exemple

Association de plusieurs flots à un objet physique : flotin = _flotin

fout1 = _fout fout2 = _fout fout3 = _fout fout4 = _fout fout5 = _fout fout6 = _fout Reject = _Reject

$IFTYPE WINNT

path=”S:\Dispatch\Source\”

_flotin = “File: !(path)srcfile70.txt”

flotin = “File: !(path)caf_invoice.xml”

_fout = “File: !(path)file!(codepostal).txt”

_fout = “File: c:\stardispatch\toto.xml”

_Reject = “File: !(path)Reject.txt”

$ENDIF

(cf § le contexte d’exécution).

LE CONTEXTE D'EXECUTION

L'utilisateur peut définir plusieurs contextes et indiquer, sur la ligne de commande, celui qu'il veut activer. StarDispatch fonctionne actuellement sous Windows et sous Unix. Par défaut, le contexte actif est initialisé à "WINNT" sous Windows et à "UNIX" sous Unix.

- Syntaxe de déclaration

$IFTYPE <Nom_Contexte>

Déclaration des variables contexte RépertoireDuTravail="…" Imprimante=" …"

.

.

Déclaration des données en entrée VarAssInput1="…" VarAssInput2="…"

.

.

Déclaration des données en sortie VarAssOutput1="…" VarAssOutput2="…"

.

.

$ENDIF

$IFTYPE et $ENDIF : délimiteurs de contexte

DECLARATION DES VARIABLES DE CONTEXTE

Variables définies dans le fichier de commandes pour gérer, dans un contexte d'exécution d'un projet, une information utilisée plusieurs fois.

Le nombre maximal de variables contexte pouvant être définies dans un projet est de 32.

- Syntaxe de déclaration

<Nom_Variable_Contexte>=["chaîne de caractères" | <num>]

- Exemple

RepTravail= "c:\Stardispatch\test\" Imprimante="lpt1"

La valeur d'une variable contexte ainsi définie peut être ensuite utilisée dans la partie déclaration et définition des flots entrants ou sortants. Cette valeur est accessible par la syntaxe suivante :

- Syntaxe d’utilisation

!(Nom_Variable_Contexte)

- Exemple

!(RepTravail)

!(Imprimante)

DECLARATION DES DONNEES EN ENTREE

Les données à traiter peuvent être fournies sous trois formes différentes :

- Les fichiers ASCII.
- Les fichiers XML
- L’entrée standard (par défaut le clavier de l’utilisateur).

Dans cette partie, l’utilisateur définit précisément la nature des sources et les associe aux variables d’association représentant les flots entrants utilisés par le système.

- Syntaxe de déclaration

<Nom_VarAss_Input> = “**File:** nom_fichier_source”

ou

<Nom_VarAss_Input> = “**Stdin:**”

- Exemple

$MyInput_1$ = “**File:** c:\stardispatch\test\file.txt”

Dans cet exemple, nous avons précisé que la source est un fichier de type texte et nous l’avons associé à la variable représentant le flot entrant $MyInput_1$.

$MyInput_2$ = “**Stdin:**”

Dans cet exemple, nous avons précisé que les données proviennent de l’entrée standard (les données seront fournies directement par l’utilisateur via son clavier) et nous les avons associées à la variable représentant le flot entrant $MyInput_2$.

$MyInput_1$ = “**File:** c:\stardispatch\test\file.xml”

Dans cet exemple, nous avons précisé que la source est un fichier XML et nous l’avons associé à la variable représentant le flot entrant $MyInput_1$.

DECLARATION DES DONNEES EN SORTIE

Le résultat du traitement de StarDispatch peut être exploité comme suivant :

1. **Sous forme de fichiers finaux directement exploitables**

   - Syntaxe

<Nom_VarAss_Output> = “**File:** nom_fichier” (cf § File:).

ou dans le cas d’utilisation de l’instruction GroupBy (cf § GroupBy)

<Nom_VarAss_Output> = “**File :***nom_fichier*[!(DocNr)]”, [DocNr=<num>][, GroupNr=<num>]

Dans le cas d’une utilisation de GroupBy, les pages sont écrites dans des documents distincts. Pour pouvoir les différencier, il faut utiliser la variable StarDispatch DocNr.

1. **Sous forme de fichiers temporaires dédiés aux applications**

   - Syntaxe

<Flot de sortie> = “**Run:** Cmd !(DataFile)” (cf § Run:).

LISTE DES VARIABLES STARDISPATCH

Ces variables sont mises à la disposition de l’utilisateur qui peut donc les initialiser pour l’exploitation de fonctions spécifiques de StarDispatch.

DocNr	Cette variable est un compteur de document qui permet de numéroter les documents (ensemble de pages) traités sur chaque flot sortant. Par défaut, la numérotation des documents commence à 1. Cependant, la valeur peut être initialisée par l’utilisateur au moment de la définition du flot sortant concerné.
GroupNr	Cette variable est un compteur de groupe de documents. Sa valeur évolue à chaque changement de groupe de documents suivant la valeur du paramètre PackBy. Par défaut, sa valeur initiale est 1. Cependant, cette valeur peut être initialisée par l’utilisateur, au moment de la définition du flot sortant concerné.

Ces variables StarDispatch ne sont pas initialisables par l’utilisateur.

DATAFILE	Cette variable désigne le fichier temporaire contenant les données traitées par StarDispatch et destiné aux applications externes. Cette variable est utilisable dès lors qu’on utilise l’instruction “Run:” (cf § Run:).
SYSDATE_YEAR	Cette variable désigne l’année de la date système. Elle permet de récupérer, après concaténation, la totalité de la date système. Elle n’est utilisable que dans la définition des flots en sortie et cela même s’il n’y a pas de critère de regroupement défini.

SYSDATE_MONTH	Cette variable désigne le mois correspondant à la date système. Elle permet de récupérer, après concaténation, la totalité de la date système. Elle n'est utilisable que dans la définition des flots en sortie et cela même s'il n'y a pas de critère de regroupement défini.
SYSDATE_DAY	Cette variable désigne le jour correspondant à la date système. Elle permet, de récupérer, après concaténation, la totalité de la date système. Elle n'est utilisable que dans la définition des flots en sortie et cela même s'il n'y a pas de critère de regroupement défini.
SYSDATE_HOUR	Cette variable désigne l'heure correspondant à la date système. Elle permet, de récupérer, après concaténation, la totalité de la date système. Elle n'est utilisable que dans la définition des flots en sortie et cela même s'il n'y a pas de critère de regroupement défini.
SYSDATE_MIN	Cette variable désigne le temps en minutes correspondant à la date système. Elle permet, de récupérer, après concaténation, la totalité de la date système. Elle n'est utilisable que dans la définition des flots en sortie et cela même s'il n'y a pas de critère de regroupement défini.
SYSDATE_SEC	Cette variable désigne le temps en secondes correspondant à la date système. Elle permet, de récupérer, après concaténation, la totalité de la date système. Elle n'est utilisable que dans la définition des flots en sortie et cela même s'il n'y a pas de critère de regroupement défini.

LIGNE DE COMMANDE

StarDispatch peut être lancé via la ligne de commande sous Windows et sous Unix de la manière suivante :

- Syntaxe sous Windows

stardispatch /**c**CmdFile**.sd** [/**Options**]

- Syntaxe sous Unix

stardispatch –**c**CmdFile**.sd** [-**Options**]

c : Indique la présence d'un fichier de commandes sur la ligne de commande.

.sd : Extension du fichier de commandes StarDispatch, par convention.

Les **Options** permettent de fournir des informations complémentaires concernant l'environnement de StarDispatch décrites ou non dans le fichier de commandes.

vVar=<valeur>	Cette option permet de définir des variables contexte. Elle offre la possibilité de définir les flots entrants et/ou sortants sur la ligne de commande. Chaque définition doit être précédée par cette option.  Le nombre maximal de variables contexte pouvant être définies dans un projet est de 32.	*sous Windows :* **/v**fichier1=”sp /c file.sp” **/v**fichier2=fichier.txt **/v**path=”c:\archive\” *sous Unix :* **-v**fichier1=”sp -c file.sp” **-v**fichier2=”fichier.txt” **-v**path=”/home/archive/”
v_Flux=<valeur>	Cette option permet de définir **directement** les flots entrants lorsque le fichier entrant est inconnu.	*sous Windows :* **/v_**FileIn=”file:TheDataFile” ou FileIn est un nom de flot entrant *sous Unix :* **–v_**FileIn=”file:TheDataFile” ou FileIn est un nom de flot entrant
e<valeur>	Cette option permet de définir le contexte à exécuter. Par défaut cette valeur est initialisée à “WINNT” sous Windows et à “UNIX” sous Unix.	*sous Windows :* **/e**Serveur1 *sous Unix :* **-e**Serveur1
jN/Y	Cette option permet de demander un journal d’exécution de StarDispatch dans le fichier log. Par défaut, la fonction est désactivée et le journal non délivré.	*sous Windows :* **/j**N/Y *sous Unix :* **-j**N/Y
m <num>	Cette option permet de modifier la valeur attribuée par défaut à Maxlines “99999”, afin d’allouer de la mémoire dans le cas ou, dans la définition d’un flot entrant ou sortant, l’utilisation de Maxlines est désactivée (= -1).	

Log=<valeur**>**	Cette option permet de définir le répertoire dans lequel on doit créer le fichier log. Par défaut, ce fichier est stocké dans le répertoire de StarDispatch.	
Tmp=<valeur>	Cette option permet de définir le répertoire dans lequel sont crées les fichiers temporaires. Par défaut, ces fichiers sont stockés dans des répertoires temp (Windows) ou tmp (Unix) selon les systèmes.	
	sous Windows : /tmp c:\temp\….	*sous Unix :* tmp/home/tmp/….
l	Cette option permet d'appeler l'interface de validation du logiciel.	
	sous Windows : **/l**	*sous Unix :* **-l**
s	Cette option permet de ne pas afficher, durant l'exécution, d'informations relatives à son déroulement.	
	sous Windows : **/s**	*sous Unix :* **-s**

- Exemples

1. Exemple 1

Windows :

stardispatch /c scenario.sd

Unix :

stardispatch -c scenario.sd

Dans cet exemple, nous exécutons StarDispatch en utilisant le fichier de commandes "scenario.sd". Nous supposons ici que toutes les déclarations concernant les entrées/sorties, ainsi que les variables contexte nécessaires, sont décrites dans le fichier de commandes.

1. Exemple 2

Windows :

stardispatch /c scenario.sd **/v** path1="c:\stardispatch\archive\"

/v path2="c:\stardispatch\test\"

Unix :

stardispatch -c scenario.sd –**v**path1="/home/stardispatch/archive"

-v path2=”/home/stardispatch/test”

Dans cet exemple nous exécutons StarDispatch en utilisant le fichier de commandes “scenario.sd” et nous définissons deux variables contexte “path1” et “path2” contenant les répertoires de travail.

Windows :

stardispatch /c scenario.sd **/v** run1=”sp /c file.sp /d lpt1

!(DataFile)”

Unix :

stardispatch -c scenario.sd **-v** run1=”sp -c file.sp -d lpt1

!(DataFile)”

Dans cet exemple nous exécutons StarDispatch en utilisant le fichier de commandes “scenario.sd” et nous définissons une variable contexte “run1” qui contient la commande de lancement de l’application StarPage.

Pour valider **StarDispatch sous Unix**, appliquer la procédure suivante :

#/users/starjet>stardispatch -l #Validation of (1) Stardispatch ? -> **1**

#Product already installed. Do you want to continue (Y/N) ? **Y**

#StarJet Product Identification Number (SPIN) : 0000000000 #Temporary validation : 0 day(s) left.

#Your reference code is : #5K3474-#1TPHM4-##9IXOH-#7PS- #FB9CTU-DNG1MB-####AB5

#Please enter your validation code:

==>**1764j5v-#syoxzc-17ps+f+s**

#Temporary validation : 35 day(s) left. #Press any key to continue. #/users/starjet>

1. Exemple 5

Pour valider StarDispatch sous Windows, appliquer la procédure suivante :

#C:\StarJet\Bin32>C:\StarJet\Bin32\stardispatch.exe /l #Validation of (1) Stardispatch ? -> 1

#Product already installed. Do you want to continue (Y/N) ? Y

#StarJet Product Identification Number (SPIN) : 0000000000 #Product validated permanently.

Your reference code is : #5KECQO-#2DOOFD-##9IXOP-#7PS-#+GCAQ#- #J89ZD5

Please enter your validation code:

==>

FICHIER DE COMMANDES

Ce fichier permet la communication entre les utilisateurs et StarDispatch. L'utilisateur définit les objets et leur associe des traitements nécessaires. Ce fichier contient des informations suivantes :

Création des variables Tags
Définition des conditions
Déclaration des boîtes
Déclaration des flots entrants
Déclaration des flots sortants
Définition des traitements
Déclaration des associations
Définition du contexte d'exécution

Nous conseillons cet ordre dans un souci de lisibilité du fichier de commandes. La seule contrainte à respecter consiste à définir tous les objets avant de les utiliser.

Dans un fichier de commande, les lignes ne doivent pas comporter plus de 256 caractères.

DES INSTRUCTIONS

Les instructions de StarDispatch permettent de :

- définir les différents objets d'un projet StarDispatch,
- définir le schéma de communication entre ces objets,
- définir, pour les données, les itinéraires à emprunter,
- définir le(s) contexte(s) d'exécution d'un projet.

SELECT

L'instruction **Select** permet de définir les propriétés de l'arbitre d'une boîte de regroupement.

A noter que chaque boîte de regroupement possède son propre arbitre.

- Syntaxe de déclaration (mode ASCII)

merge …

Select [ascent|descent] [ascii|number|date] [synchronized]

FlotInput$_0$ **on** [ExpStr$_0$ | <Variable_tag_ExpStr>]

FlotInput$_n$ **on** [ExpStr$_n$|<Variable_tag_ExpStr>]

Endmerge

ascent : indique que les pages sont triées en ordre ascendant.

descent : indique que les pages sont triées en ordre descendant.

ascii : indique que l'information sur laquelle le tri est effectué est de type ascii. number : indique que l'information sur laquelle le tri est effectué est de type numérique.

date : indique que l'information sur laquelle le tri est effectué est de type date. **synchronized** : indique que les pages ne sont regroupées que si que l'information de regroupement concorde pour les différents flots entrants.

FlotInput$_i$ **on** ExpStr$_i$: indique que l'information se trouve dans la zone pointée par ExpStr$_i$ sur l'entrée FlotInput$_i$. A noter qu'une ExpStr peut être remplacée par une variable tag.

- Exemple (mode ASCII) :

Select ascent number FlotInput1 **on** L2:C90,94 FlotInput2 **on** Codepostal

Dans cet exemple, nous avons défini l'arbitre d'une boîte de regroupement à deux entrées (FlotInput1 et FlotInput2).

Sur la première ligne, nous avons précisé que les pages en entrée seront toutes triées par ordre ascendant et l'information sur laquelle ce tri est effectué, est de type entier.

Nous avons indiqué aussi que l'information se trouve dans la zone L2:C90,94 sur la première entrée (FlotInput1) et dans la zone désignée par la variable tag Codepostal sur la deuxième entrée (FlotInput2).

Dans cet exemple, l'arbitre traite implicitement les informations de manière non synchrone.

- Syntaxe de déclaration (mode XML)

merge …

Select [ascent|descent] [ascii|number|date] [synchronized]

FlotInput$_i$ **on** [ExpXPath$_i$]

Endmerge

FlotInput$_i$ **on** ExpXPath : indique que l'information se trouve sur l'élément pointé par une expression XPath sur l'entrée FlotInput$_i$.

CONDITION

L'instruction **condition** permet de créer une variable pour nommer une expression booléenne. Cette variable, ainsi créée, sera utilisée pour augmenter la lisibilité des lignes de commande. A chaque appel de cette variable, l'expression booléenne associée sera réévaluée.

- Syntaxe de déclaration

condition <Nom_variable_condition> **is** ExpBool

ExpBool (cf § L'Expression Booléenne) est une expression booléenne qui sera associée à la variable "Nom_variable_condition". On peut noter qu'il est toujours possible de faire appel à une variable condition lors de l'écriture d'une expression booléenne.

- Exemple

1. **condition** Paris **is** L3:C90,91 %= 75
2. **condition** Yvelines **is** L3:C90,91 %= 78
3. **condition** Province **is not** Paris **AND not** Yvelines

Le premier exemple associe une expression booléenne simple à la variable Paris. Chaque fois qu'on utilise la variable Paris, cette expression sera réévaluée et le résultat (vrai ou faux) sera renvoyé. Le deuxième exemple est sur le même modèle que le premier.

Le troisième exemple montre l'utilisation, dans la définition d'une nouvelle variable condition, de variables condition (Paris, Yvelines) déjà définies.

DATE

L'instruction **date** permet, dans la définition d'un tag (cf § usedtag) d'extraire une donnée de type date au format spécifié. Le résultat stocké dans le tag est interprétable par les arbitres et les opérateurs.

- Syntaxe

date("chaîne")

- Exemple

date("DDMMYYYY")

Dans cet exemple, la date est formatée sous la forme "DDMMYYYY".

D abréviation pour le jour (**D**ay).

M abréviation pour le mois (**M**onth).

Y abréviation pour l'année (**Y**ear).

La valeur d'une variable de type date sera calculée au moment de sa première lecture

(cf § usedtag).

Les formats suivants sont reconnaissables par StarDispatch :

Formats sans séparateurs	**Formats avec séparateurs**
DDMMYY DDYYMM MMDDYY MMYYDD YYMMDD YYDDMM DDMMYYYY DDYYYYMM MMDDYYYY MMYYYYDD YYYYMMDD YYYYDDMM	**DD**sep**MM**sep**YY DD**sep**YY**sep**MM MM**sep**DD**sep**YY MM**sep**YY**sep**DD YY**sep**MM**sep**DD YY**sep**DD**sep**MM DD**sep**MM**sep**YYYY DD**sep**YYYY**sep**MM MM**sep**DD**sep**YYYY MM**sep**YYYY**sep**DD YYYY**sep**MM**sep**DD YYYY**sep**DD**sep**MM

Les cinq séparateurs suivants sont autorisés : "." ou "/" ou "**:**" ou "**–**" ou "espace".

Un cas générique de formatage représenté ci-dessous augmente considérablement la flexibilité de cette instruction.

Dsep**M**sep**Y M**sep**Y**sep**D**

Dsep**Y**sep**M Y**sep**M**sep**D**

Msep**D**sep**Y Y**sep**D**sep**M**

L'utilisation de séparateurs permet de ne pas spécifier le nombre de chiffres par jour, mois ou année.

EJECTCODE

L'instruction **ejectcode**, réservée aux flots de type ASCII, permet de définir un code de saut de page.

- Syntaxe

Ejectcode =<num>

ou <num> est la valeur décimale d'un caractère de saut de page (par défaut, num> est affecté à 12, code ASCII d'un caractère de saut de page).

- Dans le cas des flots entrants, StarDispatch cherche ce caractère dans la source pour découper les entités pages.
- Dans le cas des flots sortants, StarDispatch insère ce caractère après avoir terminé le transfert du contenu de chaque entité page. Dans ce cas, ejectcode égal à 0 permet d'écrire les pages les unes après les autres sans séparateur.
- L'utilisation de cette instruction est facultative. L'utilisateur ajoute

cette ligne dans la définition des flots entrants ou sortants pour changer explicitement la valeur par défaut de cette instruction. Par défaut, cette instruction contient le code ascii d'un Form-Feed (/f équivalent à 12).

- L'utilisateur peut désactiver cette instruction en lui associant la valeur –1 dans la définition des flots entrants ou sortants auquel cas StarDispatch ne cherchera pas de code de saut de page.

EXPRESSION

L'instruction **expression** permet de définir, pour un tag, la façon d'accéder à l'information dans la page de données. Cette instruction est suivie par la définition d'une ExpStr (cf § L'Expression String) ou d'une expression XPath.

- Syntaxe (mode ASCII)

Usedtag <Nom_variable_tag> is [String|Date("<format>")] as Expression <ExpStr>

- Syntaxe (mode XML)

Usedtag <Nom_variable_tag> is [String] as Expression

<ExpXPath>

FIELD

L'instruction **field,** réservée aux flots de type ASCII, permet d'accéder aux données sans savoir où elles sont disposées sur la page. En revanche, il est indispensable de définir une séquence de caractères suffisamment remarquable pour en permettre l'identification.

- Syntaxe

Usedtag <Nom_variable_tag> is [String | Date ("<format>")] as Field "chaîne"[NOCASE]

[COL <num | num – num >] [LINE <num | num – num >] [START <<num> | <car>>]

[STOP <<num> | <car>>]

NOCASE : Le mot-clef indiquant que la recherche sur chaîne ne doit pas tenir compte des minuscules / majuscules.

COL, LINE : La recherche sur cette chaîne de caractères peut se faire sur toute la page ou se faire uniquement sur une partie de la page. Les mots clef LINE et COL permettent de définir cette zone de recherche.

LINE 1 indique que seule la ligne 1 doit être parcourue en recherche.

LINE -10 indique que la recherche doit être effectuée sur les lignes 1 à 10 de la page.

LINE 10 – 25 indique une plage située de la ligne 10 à la ligne 25.

LINE 10 – indique enfin que la zone de recherche est située de la ligne 10 à la fin de la page.

Le mot clef COL offre exactement les mêmes possibilités, en appliquant la même syntaxe.

START, STOP : Une fois la chaîne de recherche trouvée, il est possible d'indiquer la valeur à extraire, soit par l'indication de caractères de début et de fin de champ, soit par le numéro de colonne du début de la valeur, et une longueur de champ.

Ces deux possibilités peuvent être mixées, c'est à dire qu'il est possible d'extraire une variable qui serait située derrière le caractère : (deux points) et sur une longueur de

20 caractères (START ":" STOP 20). Par défaut START est situé juste après "chaîne" et STOP est la fin de la ligne courante.

- Exemple

field "TVA :"

L'information recherchée se situe après la chaîne "TVA :", et sa longueur ne sera limitée que par la fin de la ligne courante.

field "TOTAL " START "!" STOP 12

L'information recherchée se trouve après la chaîne "TOTAL". Elle commence par un point d'exclamation et sa longueur est fixée au maximum à 12 caractères.

FILE:

L'instruction **File:** permet d'identifier la source comme étant un fichier contenant les données à traiter ou la destination des données traitées.

- Syntaxe

$IFTYPE Nom_De_Plate-forme

.

$NomFlotInput_1$= "**File:** Nom_De_Fichier_Physique "

.

$NomFlotInput_n$= "**File:** Nom_De_Fichier_Physique "

…

$NomFlotOutput_1$= "**File:** Nom_De_Fichier_Physique "

.

$NomFlotOutput_m$= "**File:** Nom_De_Fichier_Physique "

.

$ENDIF

- Exemples

1. Sous Windows

$IFTYPE WINNT

….

FlotInput1 = “**File:** c:\stardispatch\test\f90t.txt” FlotInput2 = “**File:** c:\stardispatch\test\f70t.txt” FlotInput3 = “**File:** c:\stardispatch\test\f80t.txt” FlotInput4 = “**File:** c:\stardispatch\test\invoice.xml” FlotOutput1 = “**File:** c:\stardispatch\test\f789t.txt”

….

$ENDIF

Dans cet exemple, nous avons défini quatre fichiers en entrée et un fichier en sortie qui contiendra le résultat du traitement. Nos fichiers en entrée sont de type ASCII et XML et, nous avons souhaité récupérer le résultat sous la forme d’un fichier de type ASCII. Ce même exemple sous UNIX aura la forme suivante :

1. Sous Unix

$IFTYPE UNIX

…

FlotInput1 = “**File:** /home/user1/stardispatch/test/f90t.txt” FlotInput2 = “**File:** /home/user1/stardispatch/test/f70t.txt” FlotInput3 = “**File:** /home/user1/stardispatch/test/f80t.txt” FlotInput4 = “**File:** /home/user1/stardispatch/test/invoice.xml” FlotOutput1 = “**File:** /home/user1/stardispatch/test/f789t.txt”

…

$ENDIF

Dans le cas de l’utilisation de l’instruction **GroupBy** (cf § GroupBy) et pour découper le résultat du traitement de StarDispatch en documents, on utilisera la syntaxe suivante :

- Syntaxe

<FlotOutput$_i$>=”**File:**[Nom_De_Fichier_Physique]**!(DocNr)**“, [DocNr=<num>]

Nom_De_Fichier_Physique: Cette chaîne de caractères permet de donner une racine au nom du fichier.

DocNr : Variable StarDispatch permettant de numéroter le nombre de documents traités sur chaque flot de sortie et de rendre unique le document créé.

- Exemple

$IFTYPE WINNT

path=”c:\archive\”

FlotOutput$_1$= “**File:** !(path)MyDir**!(**GroupNr**)**\MyDoc**!(**DocNr).DOC”,

DocNr = 0,GroupNr = 5

$ENDIF

Dans cet exemple, nous souhaitons organiser les documents du FlotOutput1 (cf § GroupBy) de la manière suivante :

- Nous souhaitons que nos documents soient rangés dans le répertoire “c:\archive\”, d’où l’initialisation de la variable contexte **path**.
- Nous souhaitons aussi que les documents soient nommés “MyDoc” suivi du numéro de document calculé par le compteur DocNr. Ce numéro commence à zéro (d’où l’initialisation du compteur).
- Nous souhaitons que nos documents soient regroupés par paquets de 500. Cette valeur est définie par l’instruction PackBy (cf § PackBy) Chaque paquet de 500 doit être placé dans un répertoire distinct nommé “MyDir” suivi d’un numéro de groupe de document, calculé par la variable externe GroupNr. Dans cet exemple, nous souhaitons que ce numéro commence à 5.
- Si le répertoire destination n’existe pas, il est automatiquement créé.

– Le résultat de cette ligne est le suivant :

C:\archive\ MyDir5\MyDoc0

……

C:\archive\ MyDir5\MyDoc499 C:\archive\ MyDir6\MyDoc500

…….

C:\archive\ MyDir6\MyDoc999

GROUPBY

L’instruction **GroupBy** permet de découper un flot de sortie en document(s). Un document est un ensemble de pages répondant à un ou plusieurs critères.

- Pour l’identification des pages appartenant au même document, une liste de variables tags peut être définie. La rupture entre deux documents est provoquée par la différence entre la valeur d’une des variables tags de cette liste.
- Le compteur document **DocNr** utilisé à chaque instruction **GroupBy** permet

de numéroter les documents traités sur chaque flot de sortie. La numérotation des documents commence à 1 mais il est possible de changer cette valeur initiale dans la définition du contexte d'exécution.

- Syntaxe

GroupBy : Nom_variable_tag1,

[Nom_variable_tag12,… Nom_variable_tag1n]

- Exemple

Outstream FlotOutput1 **Maxlines**=66 **Ejectcode**=12

GroupBy : vartag1, vartag2, vartag3

PackBy = 500

Endstream

Outstream, Endstream : délimiteurs de flots sortants

Dans cet exemple on indique, via l'instruction **GroupBy,** que le fichier FlotOutput1 doit être découpé en documents selon la liste de variables tags (vartag1, vartag2, vartag3) donnée.

L'objectif est de regrouper les pages contenant les mêmes valeurs pour les trois variables de la liste (cf § File:).

MAXLINES

L'instruction **maxlines**, réservée aux flots de type ASCII, permet de définir le nombre maximum de lignes dans une page.

- Syntaxe

Maxlines = <Nombre_lignes> [**full**]

Dans le cas des flots entrants, StarDispatch charge la page suivante après avoir lu "Nombre_lignes" lignes dans le fichier ASCII.

Dans le cas des flots sortants, StarDispatch passe à l'écriture de la page suivante après avoir terminé l'écriture de "Nombre_lignes" lignes. Dans le cas où le nombre de lignes dans une entité page serait inférieur au nombre de lignes défini par l'instruction **maxlines**, l'option **full** permet de compléter la page cible par des lignes vides, pour atteindre le nombre de lignes donné par cette instruction. Cette option est réservée pour la définition des flots sortants.

L'utilisation de cette instruction est facultative. L'utilisateur ajoute cette ligne dans la définition des flots entrants ou sortants pour changer explicitement la valeur par défaut de cette instruction (la valeur par défaut est fixée à 66 lignes).

L'utilisateur peut désactiver cette instruction en lui associant la valeur-1

dans la définition des flots entrants ou sortants. Dans ce cas, StarDispatch attribue la valeur “99999” à cette instruction. L’attribution de cette valeur est justifiée par des contraintes techniques, mais l’utilisateur peut changer cette valeur sur la ligne de commande en utilisant l’option “m” (cf § ligne de commande).

MAXPAGES

L’instruction **Maxpages** permet de découper un flot de sortie en document(s). Un document est un ensemble de pages répondant à un ou plusieurs critères.

Cette instruction créé un document pour un nombre défini de pages (ex : toutes les n pages). Elle peut être combinée avec l’instruction GroupBy.

ON

L’instruction **on** permet de définir la zone dans laquelle une information de tri se trouve. Cette instruction est utilisée spécialement dans la définition des boîtes de regroupement.

- Syntaxe

<Nom_input> **on** [**ExpStr**|<Variable_tag_ExpStr>|**ExpXPath**]

PACKBY

L’instruction PackBy permet de définir le nombre de documents à regrouper dans le cas des instructions GroupBy ou Maxpages. Un numéro de document sera associé à chaque groupe de documents et la variable GroupNr sera incrémentée à chaque changement de groupe. Par défaut, sa valeur initiale est 1. Mais il est possible de lui préciser une valeur initiale différente.

PAGE

L’instruction **page,** réservée aux flots de type ASCII, permet de définir une condition de rupture de page. Cette instruction offre la possibilité de séparer différents groupes de données à partir d’une chaîne de caractères se trouvant à une position donnée (la colonne) dans la source.

Chaque occurrence de cette chaîne de caractères déclenche la création d’une entité page ayant cette chaîne de caractères à une position donnée (la ligne et la colonne).

- Syntaxe

Page = ExpLC **= “**chaîne de caractères**“**

ExpLC : détermine la zone où se trouve la chaîne de caractères sur la page.

- LC : Ligne Colonne
- Exemples

1. Exemple 1

Page = L1:C8,15 = “CLIENT :”

Cet exemple indique à StarDispatch qu’il faut chercher la chaîne “CLIENT :” à la 8ème colonne (L *:C8,_) de la source pour découper les données. A chaque occurrence de la chaîne “CLIENT :”, il faut créer une entité page et charger les données jusqu’à sa prochaine occurrence, de manière à ce que la chaîne “CLIENT :” se trouve à la première ligne (L1:C8,_) et commence à la 8ème colonne.

(L-x:C_,_), désigne la xème ligne par rapport la fin de la page. Cela permet à StarDispatch de découper les données n’importe où sur la page.

1. Exemple 2

Page = L-2:C8,8 = “CLIENT :”

Cet exemple indique à StarDispatch qu’il faut chercher la chaîne “CLIENT :” à la 8ème colonne (L_:C8,_) de la source pour découper les données. A chaque occurrence de la chaîne “CLIENT :”, il faut créer une entité page et charger les données jusqu’à sa prochaine occurrence, de manière à ce que la chaîne “CLIENT :” se trouve à l’avant dernière ligne (L-2:C8,_) et commence à la 8ème colonne.

- : _ équivaut à “quelque soit”

RUN:

L’instruction **Run:** permet de définir le type du flot sortant de StarDispatch. Dans ce cas, le flot est envoyé vers une application ou une commande. A la suite de cette instruction, il faut définir la ligne de commande à exécuter. Le flot de sortie sera nommé par le mot-clef DataFile.

- Syntaxe

<NomOutput> = **Run:**Cmd**!(**DataFile**)**

SOURCEPAGENUMBER

L’instruction **SourcePageNumber** permet, après création des pages, de leur attribuer un numéro dans le fichier entrant. Sa valeur sera calculée au moment de la première lecture de la page de données. Cette valeur est stockée dans la variable tag

- Syntaxe

Usedtag <Nom_variable_tag> **is String as SourcePageNumber**

STDIN:

L’instruction **Sdtin:** permet de définir le type du flot entrant dans StarDispatch. Dans ce cas, le flot est généré par l’entrée standard de la

machine (généralement le clavier).

- Syntaxe

<NomInput> = “**Stdin:**”

STDOUT:

L’instruction **Sdtout:** permet de définir le type du flot sortant de StarDispatch. Dans ce cas, le flot de sortie est envoyé vers la sortie standard de la machine (généralement l’écran).

- Syntaxe

<NomOutput> = “**Stdout:**”

STRING

L’instruction string permet de définir la valeur d’une variable Tag comme étant une chaîne de caractères (cf § Usedtag) Sa valeur sera calculée au moment de la première lecture de la page de données.

TAG

L’instruction **tag** permet de créer des variables tags. Ces variables seront définies ensuite, dans la déclaration des flots entrants par l’instruction **Usedtag**. Une même variable pourra donc avoir autant de définitions que de flots l’utilisant. Il suffit de la nommer pour y faire référence. Le système, suivant la page de données lue, saura quelle définition utiliser.

Ces variables tags peuvent être utilisées :

- pour définir des variables de condition et des expressions booléennes.
- pour définir les propriétés des boîtes.
- pour définir les propriétés de l’instruction GroupBy (cf § GroupBy).
- pour définir dans un contexte, la sortie physique d’un flot afin de récupérer une information stockée dans la page.
- Syntaxe

Tag <nom_de_variable_tag>

- Exemple

Tag RuptureClient **Tag** CodePostal **Tag** FactureDate

Dans cet exemple, on crée trois variables tags RuptureClient, CodePostal et FactureDate.

A noter que toute utilisation d’une variable tag est précédée de sa création par l’instruction tag.

USEDTAG

L'instruction **usedtag** permet de définir une variable tag (cf § la variable tag) déjà créée par l'instruction Tag (cf § Tag) Cette instruction est réservée pour la déclaration des flots entrants.

- Syntaxe (mode ASCII)

Usedtag <Nom_variable_tag> **is [String| Date("**<Format>**")] as Field**|**Expression**<ExpStr>|**SourcePageNumber**

- Exemple (mode ASCII)

usedtag RuptureClient **is string as field** "CLIENT :" **usedtag** CodePostal **is string as expression** L3:C90,94 **usedtag** FactureDate **is date** ("DDMMYYYY") **as expression** L2:C70,77

usedtag BonlivraisonDate **is date** ("DDMMYYYY") **as field**

"Date :"

usedtag NumeroPage **is string as SourcePageNumber**

La variable tag RuptureClient aura comme valeur l'information placée juste après la chaîne "CLIENT :".

La variable tag CodePostal aura comme valeur l'information se trouvant à la troisième ligne de la page, commençant à la colonne 90 et finissant à la colonne 94 (5 caractères).

La variable tag FactureDate aura comme valeur une date se trouvant à la deuxième ligne de la page, commençant à la colonne 70 et finissant à la colonne 77, et ayant le format "DDMMYYYY".

La variable tag BonlivraisonDate est de type date défini par un field (voir l'instruction Date – page 32). Cette définition indique à StarDispatch qu'il doit chercher la date après le mot "Date :" sur l'entité page et qu'elle est formatée comme suit "DDMMYYYY".

La variable tag NumeroPage aura comme valeur le numéro de page créée lors de la lecture.

- Syntaxe (mode XML)

Usedtag <Nom_variable_tag> **is String as Expression**

<ExpXPath>

- Exemple (mode XML)

Usedtag Num_Fact **is String as Expression XPath**("/*/*/*").Name

La variable tag Num_Fact aura comme valeur le nom de l'élément satisfaisant à la requête définie par l'expression XPath.

WHEN

L'instruction **when** permet d'associer une condition à une sortie, dans la définition d'une boîte de distribution.

- Syntaxe (mode ASCII)

< Nom_output > **when** [ExpBool | Condition]

- Exemple (mode ASCII)

$Output_1$ **when** L3:C90,91 %= 75 $Output_2$ **when** Paris

$Output_3$ **when** dateOk

La sortie $Output_1$ sera sélectionnée si l'information se trouvant dans la zone L3:C90,91 est égale à 75.

La sortie $Output_2$ sera sélectionnée si la condition **Paris** (cf § la variable condition) est vraie.

La sortie $Output_3$ sera sélectionnée si la condition **dateOk** (cf § la variable condition) est vraie.

- Syntaxe (mode XML)

<Nom_output> **when** ExpBool

- Exemple (mode XML)

$Output_1$ **when** Num_Fact = "Caratula"