Sage 100

V 9.00

Sage 100cloud Objets métiers

Propriété & Usage

Ce logiciel et sa documentation sont protégés par le Code de la Propriété Intellectuelle, les lois relatives au copyright et les traités internationaux applicables.

Toute utilisation non conforme du logiciel, et notamment toute reproduction ou distribution partielle ou totale du logiciel ou toute utilisation au-delà des droits acquis sur le logiciel est strictement interdite.

Toute personne ne respectant pas ces dispositions se rendra coupable de délit de contrefaçon et sera passible des peines pénales prévues par la loi.

La marque Sage est une marque protégée. Toute reproduction totale ou partielle de la marque Sage, sans l'autorisation préalable et expresse de la société Sage est donc prohibée.

Tous les noms de produits ou de sociétés, toute image, logo ou représentation visuelle mentionnés dans ce logiciel ou sa documentation et n'appartenant pas à Sage peuvent constituer des marques déposées par leurs propriétaires respectifs.

Conformité & Mise en garde

Compte tenu des contraintes inhérentes à la présentation sous forme de manuel électronique, les spécifications visées dans la présente documentation constituent une illustration aussi proche que possible des spécifications.

Il appartient au client, parallèlement à la documentation, de mettre en oeuvre le progiciel pour permettre de mesurer exactement l'adéquation de ses besoins aux fonctionnalités.

Il est important, pour une utilisation sûre et opérationnelle du progiciel, de lire préalablement la documentation.

Evolution

La documentation correspond à la version référencée. Entre deux versions, des mises à jour du logiciel peuvent être opérées sans modification de la documentation. Toutefois, un additif peut être joint à la documentation existante pour présenter les modifications et améliorations apportées à ces mises à jour.

Sage
10, Place de Belgique
92250 La Garenne Colombes

N°Azur 0 810 30 30 30

www.sage.fr

Table des matières

Généralités sur Sage 100cloud Objets Métiers

Présentation de Sage 100cloud Objets Métiers

Sage 100cloud Objets Métiers se présente sous la forme d'une bibliothèque logicielle ActiveX destinée à simplifier le développement d'applications accédant aux bases de données Sage 100c (SQL Server et Express).

Cet ActiveX met à la disposition des développeurs, des classes, des interfaces, des méthodes et des propriétés permettant l'accès aux bases de données Sage 100c, en lecture et en écriture.

Lors de l'ajout ou de la modification de données dans les bases de données Sage 100c, des contrôles de cohérence sont effectués par Sage 100cloud Objets Métiers.

De plus, Sage 100cloud Objets Métiers gèrent certains automatismes (initialisation de champs avec des valeurs par défaut ou des valeurs héritées) lors de la création de nouveaux enregistrements dans les bases de données Sage 100c (articles, tiers, documents, etc…).

Ce manuel présente le principe de fonctionnement de Sage 100cloud Objets Métiers à travers des exemples d'utilisation.

Note : Les exemples présentés ont été développés sous Microsoft Visual Studio en Visual Basic.Net ou C#. Ces exemples sont néanmoins facilement transposables sous d'autres langages.

Terminologie

Le développement d'applications utilisant Sage 100cloud Objets Métiers requiert des connaissances en programmation et plus particulièrement en programmation "objet".

Les concepts de base de la programmation objet qui seront employés dans ce manuel sont rappelés ci-dessous.

Objet

Un objet est une abstraction codée d'une entité ou d'une relation du monde réel.

Par exemple, un objet peut correspondre à :

- à une base de données,
- à un enregistrement dans une base de données. Cet enregistrement est manipulable par l'intermédiaire de cet objet.

Un objet résulte de l'instanciation d'une classe ou dans le cadre de Sage 100cloud Objets Métiers, il peut être créé par un autre objet de type "Factory".

Classe

Une classe permet de créer (ou d'instancier) un objet. Un objet est donc une instance de classe.

Une classe se décompose en deux parties :

- L'implémentation : le code et les données internes à l'objet, invisibles et inaccessibles à l'utilisateur de l'objet. C'est ce qu'on appelle l'encapsulation.
- L'interface : les méthodes et les propriétés de l'objet visibles et utilisables par l'utilisateur de l'objet.

Une classe peut donc être comparée à un "moule" destiné à fabriquer des objets.

Sage 100cloud Objets Métiers propose deux classes de type "Application" :

Classe "Application"	Description
BSCIALApplication100c	Classe permettant l'accès à une base commerciale
BSCPTAApplication100c	Classe permettant l'accès à une base comptable

L'instanciation de ces classes permet de créer des objets autorisant l'accès aux bases comptable et commerciale.

Dans une même application, il est possible de créer un nombre illimité d'objets de type "Application" pour accéder à des bases différentes.

En partant des objets "Application", d'autres objets peuvent être créés pour lire, modifier ou ajouter des données à une base

Interface

Une interface peut être considérée comme un "schéma" à partir duquel de nouveaux objets peuvent être créés. Les objets doivent respecter les spécifications de l'interface dont ils sont issus.

Sage 100cloud Objets Métiers propose deux types d'interfaces :

- Les interfaces de type "Factory" destinées à créer de nouveaux objets ;
- Les interfaces des Objets Métiers proprement dit résultant de l'appel des interfaces de type "Factory".

Exemples d'interfaces Sage 100cloud Objets Métiers :

Classe "Application"	Interface "Factory"	Interface objets métier	Description
BSCPTAApplication100c	IBPDeviseFactory2	IBPDevise2	Création d'un objet Devise.
BSCPTAApplication100c	IBOTiersFactory3	IBOTiers3	Création d'un objet Tiers.
BSCIALApplication100c	IBOArticleFactory3	IBOArticle3	Création d'un objet Article.

Méthode

Une méthode est comparable à une fonction ou à une procédure qui serait exécutée depuis un objet.

Une méthode permet à un objet de renvoyer une valeur (généralement calculée) ou d'exécuter une action.

Exemples de méthodes proposées par Sage 100cloud Objets Métiers :

Interface objet Métier	Méthode	Description
IBOTiers3	Solde()	Retourne le solde de l'objet Tiers.
IBOTiers3	Remove()	Supprime l'enregistrement correspondant à l'objet Tiers dans la base de données.
IBOArticle3	StockATerme()	Retourne le stock à terme de l'objet Article.
IBOArticle3	Write()	Enregistre l'objet Article dans la base de données.

Propriété

Une propriété d'un objet correspond à l'état ou à la valeur d'un élément d'un objet. La valeur d'une propriété peut être sous la forme d'une valeur numérique, d'une chaîne de caractères, d'un booléen ou même d'un objet.

Une propriété peut généralement être lue et modifiée.

Exemples de propriétés proposées par Sage 100cloud Objets Métiers :

Interface objet Métier	Propriété	Description
IBOTiers3	CT_Intitule()	Intitulé de l'objet Tiers sous forme d'une chaîne de caractères. L'intitulé est accessible en lecture et en écriture.
IBOClient3	TiersPayeur()	TiersPayeur() (sous forme d'un objet de type IBOTiersPart3) de l'objet Client. Le tiers payeur est accessible en lecture et en écriture.
IBOArticle3	IsModified()	Renvoi True (vrai) si au moins une des propriétés de l'objet Article a été modifiée. Dans le cas contraire False (faux) est renvoyé. La propriété IsModified() peut être lue, mais ne peut pas être modifiée.

Collection

Une collection est un objet qui regroupe plusieurs objets de même type.

Une collection peut être parcourue et il est possible d'accéder à un élément particulier à partir de son index.

Exemples de collection Sage 100cloud Objets Métiers :

Une collection (interface IBICollection) regroupe plusieurs tiers différents (interface IBOTiers3) : Carat Sarl, Billot, Breloque Sarl, etc…

Processus

Un processus est une entité métier permettant de gérer certains automatismes sur un ou une collection d'enregistrements. Les processus implémentent une méthode permettant de vérifier la cohérence des enregistrements (*CanProcess()*) avant l'insertion réelle des données dans les bases Sage 100c (*Process()*). Les erreurs détectées sur les enregistrements du processus peuvent être interceptées et parcourues dans une collection d'erreurs : *IFailInfoCol*.

Identifiants

Les identifiants sont des clés uniques créées pour chaque objet. Ces identifiants, sous forme de stream, peuvent être stockés puis utilisés pour accéder à l'objet auquel l'identifiant correspond. Ces identifiants, générés sur des objets persistants, garantissent d'accéder toujours au même objet même après déconnexion puis connexion du développement spécifique. Sur les objets non persistants, il est également possible de générer un identifiant, mais celui-ci ne sera utilisable que pendant la durée où le développement spécifique sera connecté à la base Sage 100c. En effet, après déconnexion, l'objet non persistant sera détruit et son identifiant devriendra alors inexploitable.

Un exemple d'utilisation des identifiants est présenté sous le chapitre *Exemples avancés*.

Description de Sage 100cloud Objets Métiers

Pré-requis

Ce manuel décrit l'utilisation de Sage 100cloud Objets Métiers à travers des exemples développés en Visual Basic.Net ou C# (environnement Microsoft Visual Studio).

Les exemples sont présentés sous la forme de fonctions et de procédures. Ils peuvent être utilisés dans le cadre des projets de développement, après avoir été éventuellement modifiés. Afin de faciliter la compréhension des développements utilisant Sage 100cloud Objets Métiers, les exemples donnés s'exécutent en mode console, notamment pour l'affichage des messages d'erreur. Les notions de programmation Windows (fenêtres, menus, listes déroulantes, etc…) dépassent le cadre de Sage 100cloud Objets Métiers. Ils ne sont donc pas abordés dans ce manuel.

Les exemples présentés sont aisément adaptables à d'autres langages.

L'environnement d'utilisation de Sage 100cloud Objets Métiers doit être au minimum Microsoft Windows 7 SP1.

Avant d'aborder le développement d'applications utilisant Sage 100cloud Objets Métiers, certains points vont être détaillés :

- Compatibilité avec les bases Sage 100c (SQL Server et Express) ;
- Installation du kit de développement ;
- Référencement de la bibliothèque *Objets100c.dll* ;
- Importation de l'espace de nom *Objets100cLib* ;
- Déploiement des applications sur les postes clients ;
- Récupération et traitement des erreurs ;
- Conventions de présentation des exemples de code source.

Compatibilité avec les bases Sage 100c

Sage 100cloud Objets Métiers version 3.10 permet l'accès aux bases de données Sage 100c Essentials, Sage 100c Standard et Sage 100c Premium version 3.xx.

La plupart des enregistrements des tables suivantes sont accessibles en lecture et en écriture :

- les tables "paramètres" (Fichier / Paramètres société),
- les comptes généraux, analytiques et reportings,

- les écritures générales et analytiques,
- les modèles de saisie,
- les modèles de grille,
- les modèles de règlement,
- les codes journaux et codes journaux analytiques,
- les banques,
- les familles et articles,
- les ressources et centres de charges,
- les tiers,
- les documents des ventes, des achats, des stocks et les documents internes (accès aux informations de valorisation pour les documents de vente et d'achat),
- les dépôts de stockage et emplacements,
- les lots,
- les collaborateurs,
- les glossaires, barèmes.

Sage 100cloud Objets Métiers publie également des processus métiers permettant de simplifier la création d'enregistrements tout en garantissant l'intégrité des données. Ces processus portent sur les traitements suivants :

- Création de pièce comptable,
- Lettrage d'écritures comptables,
- Création de document commercial,
- Contrôle qualité,
- Application des barêmes sur un document commercial,
- Transformation des documents commerciaux de vente et achat,
- Gestion du colisage sur les préparations de livraison,
- Prélèvement des série/lot sur les préparations de livraison,
- Transfert d'un article d'un dépôt/emplacement vers un autre,
- Création de ligne de sous-total,

- Recalcul du prix de revient des nomenclatures fabrication,
- Sortie d'un article géré par lot,
- Conversion d'un prospect en client.

A partir de Sage 100cloud Objets Métiers, il n'est pas possible de créer les types de document suivants :

- Facture Comptabilisée

- Archive

- Dépréciation de stock

- Tickets (ces documents ne sont pas accessibles en lecture)

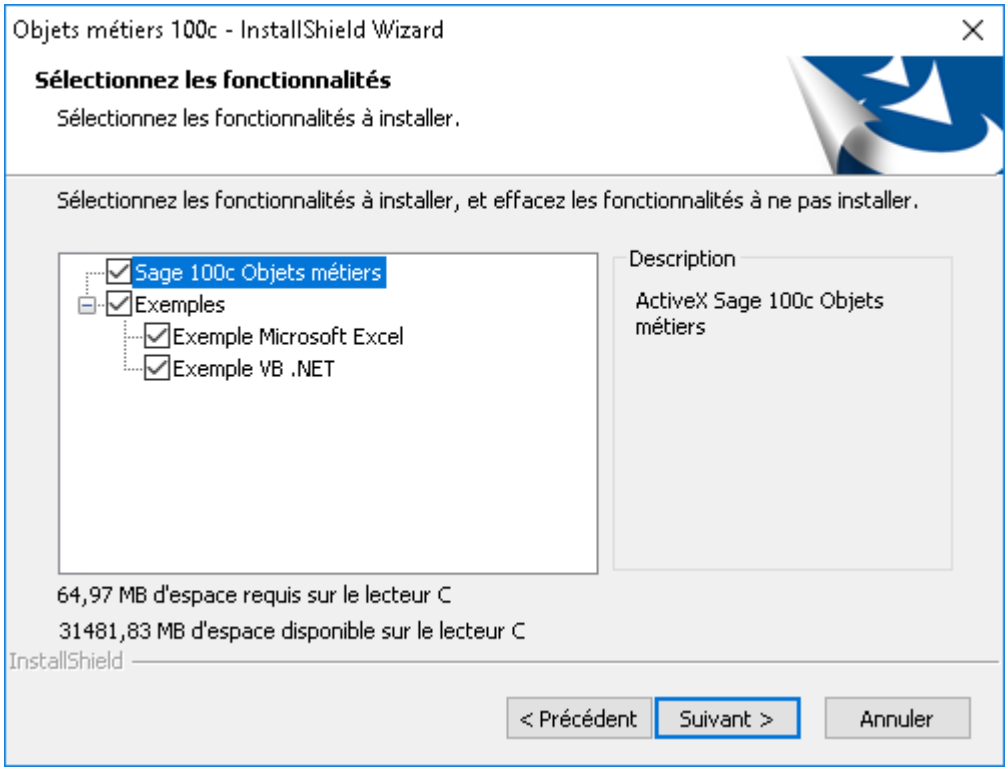
Installation du kit de développement

Le Kit de développement est destiné aux développeurs d'applications. Il permet d'installer sur un poste de travail, tous les éléments nécessaires au développement d'une application utilisant Sage 100cloud Objets Métiers. Les fichiers installés sont les suivants :

- Objets100c.dll : ActiveX de Sage 100cloud Objets Métiers
- CA Articles.xls : Fichier d'exemple excel
- Annuaire : Répertoire comprenant un projet Vb.Net (ExempleWindows.sln)

L'installation du Kit de développement s'effectue sans contrôle de licence. Ainsi, durant la phase d'installation, il n'est pas nécessaire de renseigner une clé d'installation.

Pour installer le Kit de développement Sage 100cloud Objets Métiers, il faut exécuter le fichier *kitobjets100c_100.exe* du Kit Objets métiers. L'écran de personnalisation permet de sélectionner les exemples à installer :



La distribution d'une application utilisant Sage 100cloud Objets Métiers, nécessite également d'installer l'ActiveX Objets100c.dll. Cependant, cette librairie ne doit pas être installée par le Kit de développement. Cette librairie doit être installée à partir du Runtime utilisateur. Cf. Déploiement d'applications sur les postes clients.

Description des exemples :

1. CA Articles.xls

Cet exemple a été développé sous Microsoft Excel. Il permet d'afficher un listing articles avec le chiffre d'affaires y correspondant. L'accès aux données s'effectue uniquement en lecture.

Les fonctions mises en place permettent :

- La connexion aux fichiers Sage 100c,
- La recherche de données (Clients, Familles),
- L'utilisation de méthodes associées aux objets (Chiffre d'affaires article).

2. Annuaire

Cet exemple est un projet Microsoft Visual Studio VB .Net permettant d'afficher une liste de clients et fournisseurs. L'accès aux données s'effectue en lecture et écriture.

Les fonctions mises en place permettent :

- L'ouverture d'un fichier comptable Sage 100c,
- L'affichage d'une liste de clients et fournisseurs,
- La création de nouveaux clients et fournisseurs,
- La modification des fiches tiers,
- La création de nouvelles banques et contacts tiers,
- La suppression de tiers, banques et contacts.

Types d'installation

L'activeX des objets métiers est automatiquement installé de deux manières différentes :

Installation standard

L'activeX est installé sous *Program Files\Fichiers Communs\Sage\Objets métiers\Objets100c.dll* et est référencé en base de registre sur le poste de travail. Ce type d'installation permet de référencer l'activeX à partir des outils de développement (fonctionnement non Side by side).

Installation en Side By Side

Ce type de fonctionnement, introduit depuis Microsoft Windows XP, permet de faire cohabiter sur un même poste, plusieurs versions d'un même fichier (principalement des dll). Cela permet aux développeurs de spécifier pour leurs applications, une version spécifique d'une dll side by side à utiliser. Ainsi, cette application continuera de fonctionner, même si une nouvelle version de la dll utilisée par cette application est installée sur le poste. En effet, cette nouvelle version est installée dans un nouveau répertoire (***%Windir%\WinSXS***) et ne remplace donc pas la précédente version.

Exemple :

Version Objets métiers	Nom du répertoire
1.00	X86_com.sage.cobj.100c_77d7af533b8e0189_1.0.0.1_none_xxxx
2.00	X86_com.sage.cobj.100c_77d7af533b8e0189_2.0.0.1_none_xxxx
3.00	X86_com.sage.cobj.100c_77d7af533b8e0189_3.0.0.1_none_xxxx

3.10	X86_com.sage.cobj.100c_77d7af533b8e0189_3.1.0.1_none_xxxx

Utilisation des Objets Métiers en Side By Side

Plusieurs méthodes peuvent être mises en œuvre pour utiliser l'activeX Sage 100cloud Objets Métiers en Side By side.

Ajout d'un manifest de dépendance

Pour utiliser l'activeX des Objets Métiers en Side by side, une solution consiste, depuis l'environnement de développement (.net par exemple), à ajouter une dépendance sur l'activeX dans le fichier manifest de l'application. Ce fichier manifest sera intégré à l'assembly .net de l'application et il devra contenir au minimum les instructions suivantes :

```
<dependency>
	<dependentAssembly>
		<assemblyIdentity type='win32' name='com.sage.cobj.100c' version='3.1.0.1' processorArchitecture='x86'
		publicKeyToken='77d7af533b8e0189' />
	</dependentAssembly>
</dependency>
```

Contexte d'application

Une autre solution consiste à instancier un contexte d'application. Cette procédure peut être utilisée lorsque le développeur n'est pas maître du programme exécutable, comme par exemple, lors de l'exécution d'un script Visual Basic (un fichier .vbs est exécuté par ***wscript.exe***, qui est un composant de Microsoft Windows).

Pour ce type d'exemple, il est nécessaire dans le script vbs, d'instancier un contexte en lui spécifiant le fichier manifest à utiliser :

```
' VBScript File
Dim ObjetCpta
Dim o
  set o = CreateObject("Microsoft.Windows.ActCtx")
  o.manifest = "c:\objets100\com.sage.cobj.100c.dep.manifest"

  set ObjetCpta = o.CreateObject("Objets100c.Cpta.Stream.1")
  ObjetCpta.Name = "c:\Temp\Bijou.mae"
```

…

Le fichier manifest référençant l'activeX Sage 100cloud Objets Métiers (***com.sage.cobj.dep.manifest*** dans cet exemple) devra contenir au minimum les instructions suivantes :

```
<?xml version='1.0' encoding='utf-8' standalone='yes'?>

<assembly xmlns='urn:schemas-microsoft-com:asm.v1' manifestVersion='1.0'>

	<dependency>

		<dependentAssembly>

			<assemblyIdentity type='win32' name='com.sage.cobj.100c' version='3.1.0.1' processorArchitecture='x86'
			publicKeyToken='77d7af533b8e0189' />

		</dependentAssembly>

	</dependency>

</assembly>
```

Manifest à côté de l'exécutable

Une troisième solution consiste à créer un fichier manifest pour le programme exécutable sur lequel le développeur n'est pas maître. Pour reprendre l'exemple précédent sur l'exécution de scripts vbs, il sera nécessaire, de créer un fichier manifest pour le programme ***wscript.exe***, de placer ce fichier manifest à côté de l'exécutable et de nommer ce fichier ***wscript.exe.manifest***. Ce fichier devra contenir comme dans l'exemple précédent, les instructions suivantes :

```
 <?xml version='1.0' encoding='utf-8' standalone='yes'?>

<assembly xmlns='urn:schemas-microsoft-com:asm.v1' manifestVersion='1.0'>

	<dependency>

		<dependentAssembly>

			<assemblyIdentity type='win32' name='com.sage.cobj.100c' version='3.1.0.1' processorArchitecture='x86'
			publicKeyToken='77d7af533b8e0189' />

		</dependentAssembly>

	</dependency>

</assembly>
```

Le fichier script quant à lui n'instanciera plus de contexte mais fera appel directement à l'interface stream de l'application Objets Métiers auquel il doit accéder :

```
Dim ObjetCpta

   set ObjetCpta = CreateObject("Objets100c.Cpta.Stream.1")
   ObjetCpta.Name = "c:\Temp\Bijou.mae"
         …
```

Préconisations

Il est préconisé que les développements utilisant Sage 100cloud Objets Métiers référencent l'activeX par une méthode Side By Side. En effet, ce type de référencement permet de garantir que les développements utiliseront toujours la version de l'activeX Objets Métiers pour laquelle ils ont été conçus, ceci même lorsqu'une nouvelle version majeure de l'activeX sera installée sur le poste. De plus, dans le cas d'une installation d'une version corrective des Objets Métiers, aucune modification ne sera nécessaire dans les développements puisque cette corrective sera installée de manière à ce qu'elle soit automatiquement utilisée à la place de la version qu'elle corrige (mise en place d'un fichier ***policy***).

Se reporter à la documentation Microsoft pour de plus amples détails sur la création et utilisation des fichiers manifest.

Référencement de la bibliothèque *Objets100c.dll*

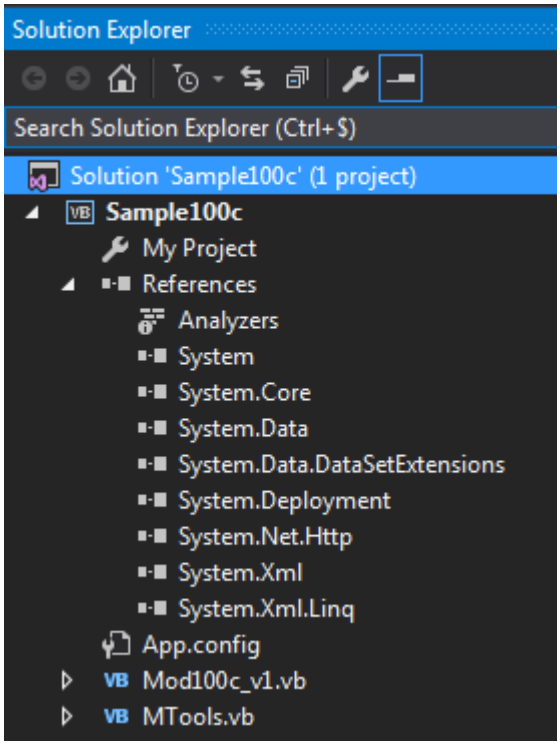
Sage 100cloud Objets Métiers se présente sous la forme d'un objet COM (ActiveX) utilisable par une majorité d'environnements de développement.

Pour utiliser un objet COM dans un nouveau projet de développement sous l'environnement Microsoft Visual Studio, il faut préalablement le référencer.

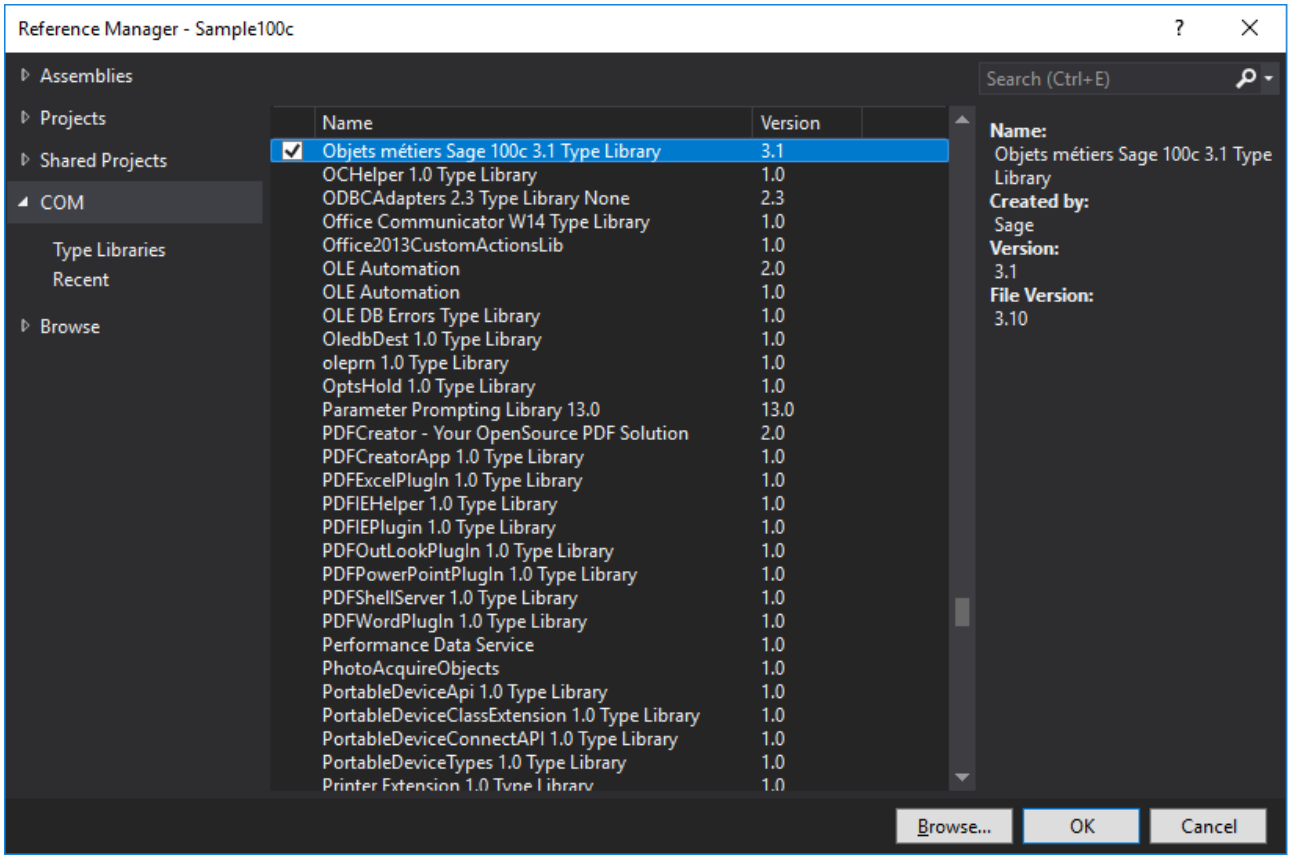
Le référencement d'un objet COM consiste à l'encapsuler dans un assembly d'interopérabilité. Cet assembly représente l'objet COM et permet de l'utiliser dans le développement d'applications .Net.

Pour référencer la bibliothèque *Objets100c.dll*, il faut procéder de la façon suivante :

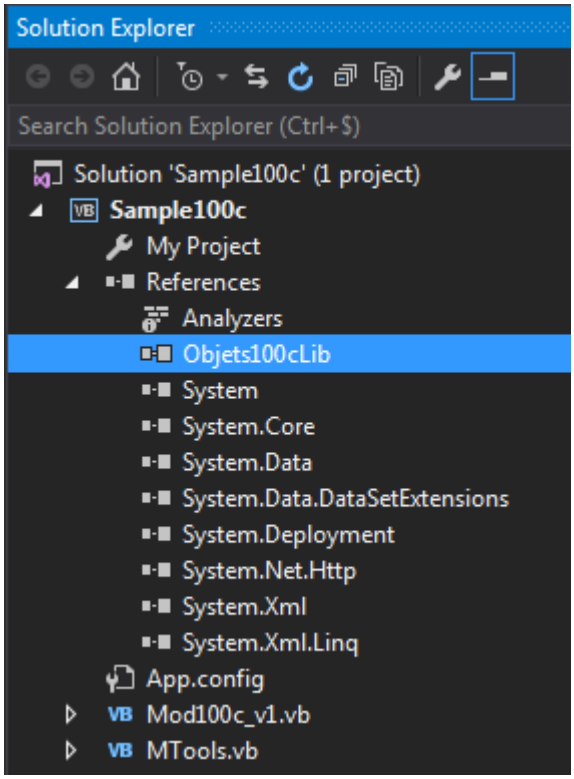
- Afficher l'Explorateur de solutions :



- Sélectionner "References" puis dans le menu contextuel : "Add reference". La fenêtre d'ajout de références s'affiche. Dans le volet "COM", sélectionner le composant "Objets métiers Sage 100c x.x Type Library" puis cliquer sur le bouton "Sélectionner". Le composant apparaît dans la liste des composants sélectionnés :

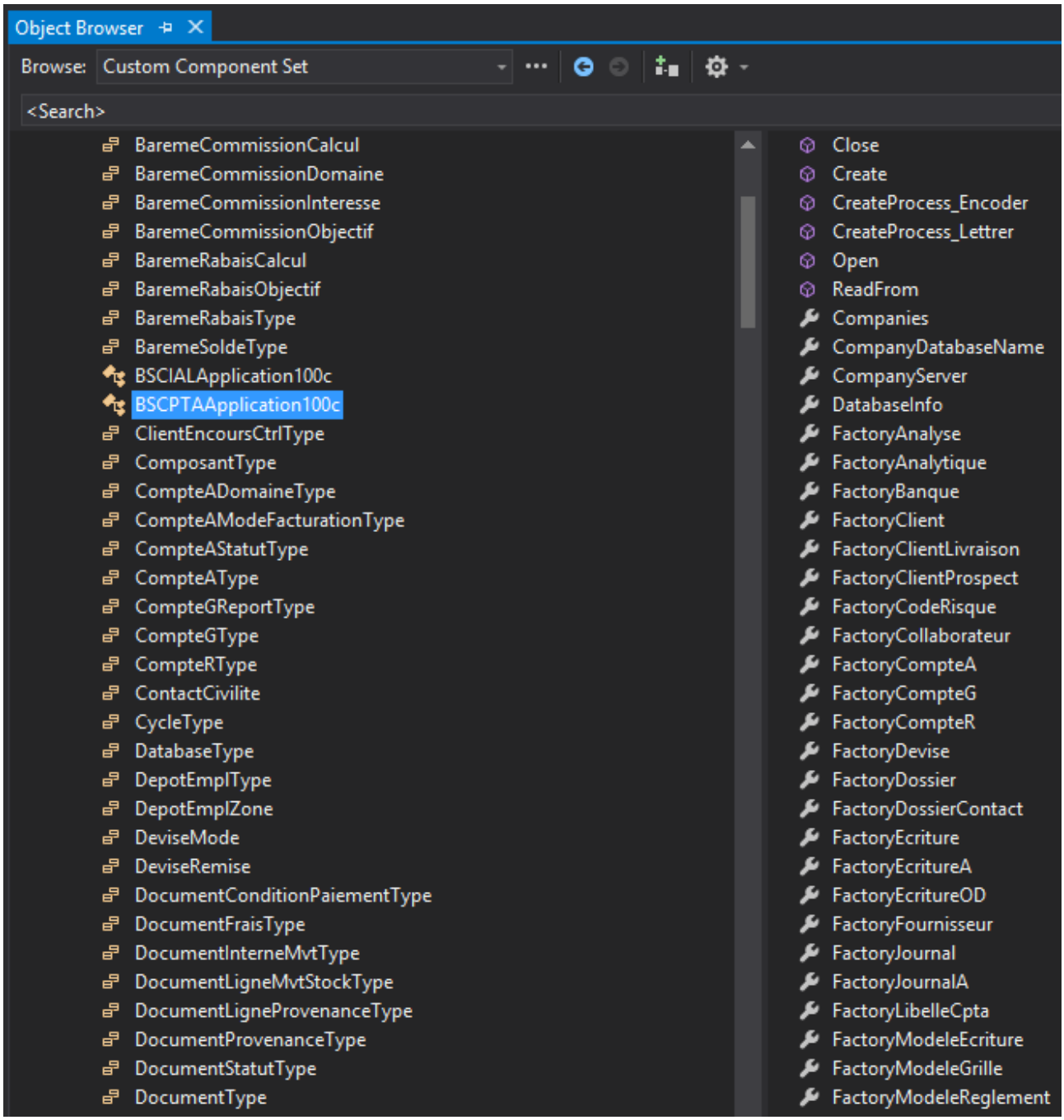


- Après validation de la fenêtre d'ajout de référence, le composant apparaît dans l'Explorateur de solutions :



Les fonctionnalités de la bibliothèque *Objets100c.dll* sont maintenant utilisables dans l'application.

Pour visualiser les différentes fonctionnalités proposées par cette bibliothèque, il faut sélectionner "Objets100cLib", puis par l'intermédiaire du menu contextuel, afficher l'explorateur d'objet :



Importation de l'espace de nom *Objets100cLib*

Pour pouvoir utiliser Sage 100cloud Objets Métiers dans un projet, il est nécessaire d'importer dans le code source l'espace de nom (*Namespace*) correspondant à la bibliothèque *Objets100c.dll*.

L'espace de nom correspondant à Sage 100cloud Objets Métiers s'intitule : **Objets100cLib**

Exemple :

```
Imports Objets100cLib

Imports System


Module Exemple


    Sub Main()

            …

    End Sub


End Module
```

Options de compilation

L'activeX Sage 100cloud Objets Métiers est une librairie 32 bits. Ainsi, pour qu'un développement s'appuyant sur Sage 100cloud Objets Métiers puisse également fonctionner sur environnement 64 bits, il faut impérativement que ce développement soit compilé en plaçant le mode de compilation sur 32 bits (x86).

Par exemple, sous Visual Studio, cette option de compilation est placée dans les paramètres de compilation, sous l'entrée *Target CPU* :

Déploiement des applications sur les postes clients

Le processus d'installation du Runtime utilisateur peut être lancé sous deux formes :

- Standard : durant la phase d'installation, il faudra valider les différents écrans d'installation.
- Silencieuse : l'ActiveX sera installé de manière transparente. C'est-à-dire qu'aucune interface ne sera affichée pendant l'installation. Cette procédure à l'avantage de s'intégrer facilement dans une installation d'applications développées par les partenaires Sage (exécution d'un fichier *.bat* par exemple).

Pour l'offre Sage 100cloud Objets Métiers, deux types de setup sont disponibles :

- kitobjets100c_100.exe : installation du Kit Revendeur (*cf. installation du Kit de développement*)
- objets100c_100.exe : installation du Runtime Utilisateur.

Installation Standard

Pour procéder à l'installation du Runtime utilisateur de manière standard, il suffit d'exécuter le setup objets100c_100.exe.

Exemple :

c:\redist\objets100c_100.exe

Génération d'un setup silencieux

Pour générer un setup silencieux, il faut créer un fichier *.iss* (*InstallShield Silent*). La création d'un fichier *iss* s'effectue en exécutant une ligne de commande.

Exemple :

c:\redist\objets100c_100.exe -r
Cet exemple suppose que l'emplacement du fichier objets100c.exe *est le répertoire* C:\redist*.*

A l'exécution de cette ligne de commande, le setup s'exécute graphiquement et il faudra valider les différents écrans d'installation. A la fin de l'installation, un fichier *Setup.iss* est généré dans le répertoire *Windows*.

Le génération du fichier .iss doit être réalisée sur un système d'exploitation de version minimale supportée par Sage 100cloud Objets Métiers : Windows 7 SP1.

Installation du setup silencieux

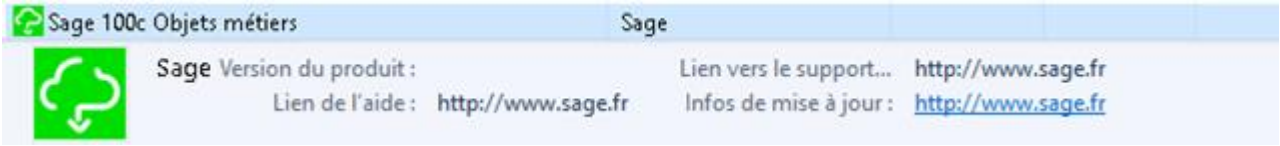
Pour installer le setup silencieux, il faudra préalablement récupérer le fichier *setup.iss* généré au paragraphe précédent, et exécuter une ligne de commande.

Exemple :

c:\redist\objets100c_100.exe -s -f1c:\test\setup.iss -f2c:\test\setup.log

f1 doit être suivi du chemin complet d'accès au fichier .iss*.*
f2 doit être suivi de l'emplacement sur lequel le fichier de trace (.log*) sera créé.*

Après exécution de la ligne de commande, le programme d'installation de Sage 100cloud Objets Métiers sera visible à partir de l'ajout/suppression de programmes de Windows :



Récupération et traitement des erreurs

Lorsqu'une méthode est appelée ou que la valeur affectée à une propriété est non conforme à ce qui est attendu par l'objet, une erreur se produit.

Une erreur déclenche une exception qui peut être traitée afin, par exemple, d'afficher un message d'erreur et de quitter l'application (après avoir correctement fermé les éventuels fichiers ouverts).

Pour récupérer et traiter une erreur lors de l'exécution d'une application, Visual Basic .Net (à l'instar d'autres langages de développement) propose les fonctions **Try … Catch … End Try.**

Exemple :

L'affectation d'une chaîne de caractères de plus de 17 caractères à la propriété CT_Num *d'un objet Tiers provoque une erreur de cohérence :*

```
Try
    Tiers.CT_Num = "NUMERO_DE_COMPTE_TIERS_DE_PLUS_DE_17_CARACTERES"
Catch ex As Exception
    Console.WriteLine("Erreur ! {0}", ex.Message)
End Try
```

Lorsque des erreurs se produisent dans un bloc **Try**, elles sont interceptées et les instructions situées dans le bloc **Catch** sont exécutées.

Dans cet exemple, l'exception est récupérée dans un objet ***ex*** dont la propriété ***Message*** retourne la chaîne de caractères "Erreur de cohérence !".

Convention de présentation des exemples de codes sources

Afin de faciliter la compréhension des développements utilisant Sage 100cloud Objets Métiers, les différents exemples donnés dans ce manuel sont présentés sous forme de fonctions.

Chaque fonction peut accepter un ou plusieurs paramètres et retourne systématiquement un booléen indiquant si le traitement s'est correctement terminé.

Dans ces fonctions, les erreurs éventuelles sont interceptées par les instructions **Try … Catch … End Try**. En cas d'erreur, le message renvoyé par l'erreur est affiché sur la console.

Exemple :

Interception d'une erreur de division entière (division par zéro).
Dans l'exemple ci-dessous, le programme tente de diviser 1 par zéro. Ceci déclenche logiquement une exception qui sera gérée par le bloc ***Catch.***

```
Imports System

Module Exemple

  Sub Main()
    Dim Nombre_1, Nombre_2, Resultat As Integer
    Nombre_1 = 1
    Nombre_2 = 0
    If DivisionEntiere(Nombre_1, Nombre_2, Resultat) Then
      Console.WriteLine("{0} divisé par {1} donne {2}", Nombre_1, Nombre_2, Resultat)
    End If
  End Sub

  Function DivisionEntiere(ByVal Nombre_1 As Integer, ByVal Nombre_2 As Integer, ByRef Resultat As Integer) As Boolean
    Try
      Resultat = Nombre_1 \ Nombre_2
      Return True
```

```
        Catch ex As Exception
            Console.WriteLine("Erreur ! {0}", ex.Message)
            Return False
        End Try
    End Function
End Module
```

Accès aux bases de données Sage 100c

L'accès aux bases de données Sage 100c s'effectue par l'intermédiaire d'objets proposant des méthodes permettant essentiellement de créer, ouvrir et fermer les bases de données.

Les points suivants vont être décrits en détail :

- L'interface IBIPersistStream ;
- Création d'une nouvelle base de données comptable ;
- Ouverture et fermeture d'une base de données comptable ;
- Création d'une nouvelle base de données commerciale ;
- Ouverture et fermeture d’une base de données commerciale.

L'interface IBIPersistStream

L'accès aux bases de données Sage 100c via Sage 100cloud Objets Métiers s'effectue par l'intermédiaire de classes basées sur l'interface *IBIPersistStream* :

- **BSCPTAApplication100c** : accès à une base de données comptable
- **BSCIALApplication100c** : accès à une base de données commerciale

Les objets issus de l'instanciation de ces classes permettent d'accéder aux bases de données Sage 100c.

Création d'une nouvelle base comptable

Une base comptable se présente sous la forme d'un objet issu de l'instanciation de la classe *BSCPTAApplication100c (1)*.

Pour cet objet, la propriété *Name()* permet de renseigner le chemin d’accès et le nom de la base comptable à créer *(2)*.

La création de la base sera effective après l’appel de la méthode *Create()* depuis l'objet représentant la base comptable *(3)*.

Exemple :

```
Option Strict Off

Imports Objets100cLib

Imports System


Module CreationBaseCpta


    Sub Main()

        ' (1)

        Dim BaseCpta As New BSCPTAApplication100c

        If CreeBaseCpta(BaseCpta, "C:\temp\test1.mae") Then

            Console.WriteLine("Base comptable correctement créée !")

        End If

    End Sub


    Function CreeBaseCpta(ByRef BaseCpta As BSCPTAApplication100c, ByVal NomBaseCpta As String) As Boolean

        Try

            ' (2)

            BaseCpta.Name = NomBaseCpta

            ' (3)

            BaseCpta.Create()

            Return True

        Catch ex As Exception

            Console.WriteLine("Erreur en création de base comptable : {0}", ex.Message)

            Return False

        End Try

    End Function


End Module
```

Ouverture et fermeture d'une base comptable

Pour ouvrir une base de données comptable existante, il est nécessaire dans un premier temps, d'instancier une classe *BSCPTAApplication100c (1),* puis d'affecter à sa propriété *Name* le chemin d'accès et le nom de la base comptable à ouvrir *(2)*.

L'objet issu de l'instanciation de *BSCPTAApplication100c* est doté d'une propriété *Loggable()* comportant deux propriétés (*UserName* et *UserPwd*). A ces propriétés, doivent être affectés le nom d'utilisateur et le mot de passe nécessaires pour ouvrir la base de données *(3)*.

Enfin, la base de données est ouverte par appel de la méthode *Open()* de l'objet issu de l'instanciation de *BSCPTAApplication100c (4)*.

L'appel de la méthode *Close()* permet de refermer la base de données *(5)*.

Exemple :

```
Option Strict Off

Imports Objets100cLib

Imports System


Module OuvertureFermetureBaseCpta


    Sub Main()

        ' (1)

        Dim BaseCpta As New BSCPTAApplication100c

        If OuvreBaseCpta(BaseCpta, "C:\temp\Bijou.mae", "USER1", "1234") Then

            Console.WriteLine("Base comptable ouverte !")

            If FermeBaseCpta(BaseCpta) Then

                Console.WriteLine("Base comptable fermée !")

            End If

        End If

    End Sub


    Function OuvreBaseCpta(ByRef BaseCpta As BSCPTAApplication100c, ByVal NomBaseCpta As String, Optional ByVal Utilisateur
As String = "<Administrateur>", Optional ByVal MotDePasse As String = "") As Boolean

        Try

            ' (2)

            BaseCpta.Name = NomBaseCpta
```

```
        ' (3)
        BaseCpta.Loggable.UserName = Utilisateur
        BaseCpta.Loggable.UserPwd = MotDePasse
        ' (4)
        BaseCpta.Open()
        Return True
      Catch ex As Exception
        Console.WriteLine("Erreur en ouverture de base comptable : {0}", ex.Message)
        Return False
      End Try
    End Function

    Function FermeBaseCpta(ByRef BaseCpta As BSCPTAApplication100c) As Boolean
      Try
        ' (5)
        BaseCpta.Close()
        Return True
      Catch ex As Exception
        Console.WriteLine("Erreur en fermeture de base comptable : {0}", ex.Message)
        Return False
      End Try
    End Function
End Module
```

A la place de la propriété ***Name***, il est également possible d'utiliser les propriétés ***CompanyServer*** et ***CompanyDatabaseName*** pour affecter respectivement le serveur/instance SQL et la base de données SQL.

Création d'une nouvelle base commerciale

Une base comptable est toujours liée à une base commerciale.

Par conséquent, lors de la création d'une nouvelle base commerciale, il est nécessaire d'instancier deux classes *(1)* :

- *BSCPTAApplication100c* faisant référence à la base comptable ;

- *BSCIALApplication100c* faisant référence à la base commerciale.

La base comptable doit ensuite être ouverte (Cf. *Ouverture et fermeture d'une base comptable*) *(2)*.

L'objet issu de l'instanciation de *BSCIALApplication100c* dispose d'une propriété *CptaApplication*. Cette propriété faisant référence à la base comptable liée à la base commerciale, il est nécessaire de lui affecter l'objet issu de l'instanciation de *BSCPTAApplication100c (3)*.

La création effective de la base commerciale est identique à la création de la base comptable :

- Affectation du nom et du chemin de la base commerciale à la propriété *Name* de l'objet issu de *BSCIALApplication100c (4).*
- Appel de la méthode *Create()* pour créer physiquement la base commerciale *(5)*.

> La base comptable doit être refermée après création de la base commerciale (6)

Exemple :

```
Option Strict Off
Imports Objets100cLib
Imports System
Imports System.IO

Module CreationBaseCial

  Sub Main()
    ' (1)
    Dim BaseCpta As New BSCPTAApplication100c
    Dim BaseCial As New BSCIALApplication100c
    If CreeBaseCial(BaseCial, "C:\temp\test2.gcm", BaseCpta, "C:\temp\test2.mae") Then
      Console.WriteLine("Base commerciale créée !")
    End If
  End Sub

  Function CreeBaseCial(ByRef BaseCial As BSCIALApplication100c, ByVal NomBaseCial As String, _
    ByRef BaseCpta As BSCPTAApplication100c, ByVal NomBaseCpta As String) As Boolean
```

```
    Try
        ' (2)
        If Not File.Exists(NomBaseCpta) Then
            CreeBaseCpta(BaseCpta, NomBaseCpta)
        End If
        If OuvreBaseCpta(BaseCpta, NomBaseCpta) Then
            ' (3)
            BaseCial.CptaApplication = BaseCpta
            ' (4)
            BaseCial.Name = NomBaseCial
            ' (5)
            BaseCial.Create()
            ' (6)
            FermeBaseCpta(BaseCpta)
            Return True
        Else
            Return False
        End If
    Catch ex As Exception
        Console.WriteLine("Erreur en création de base commerciale : {0}", ex.Message)
        Return False
    End Try
End Function

Function CreeBaseCpta(ByRef BaseCpta As BSCPTAApplication100c, ByVal NomBaseCpta As String) As Boolean
    Try
        BaseCpta.Name = NomBaseCpta
        BaseCpta.Create()
        Return True
    Catch ex As Exception
        Console.WriteLine("Erreur en création de base comptable : {0}", ex.Message)
        Return False
    End Try
```

```
    End Function

    Function OuvreBaseCpta(ByRef BaseCpta As BSCPTAApplication100c, _
     ByVal NomBaseCpta As String, Optional ByVal Utilisateur As String = "<Administrateur>", _
     Optional ByVal MotDePasse As String = "") As Boolean
        Try
            BaseCpta.Name = NomBaseCpta
            BaseCpta.Loggable.UserName = Utilisateur
            BaseCpta.Loggable.UserPwd = MotDePasse
            BaseCpta.Open()
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur en ouverture de base comptable : {0}", ex.Message)
            Return False
        End Try
    End Function

    Function FermeBaseCpta(ByRef BaseCpta As BSCPTAApplication100c) As Boolean
        Try
            BaseCpta.Close()
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur en fermeture de base comptable : {0}", ex.Message)
            Return False
        End Try
    End Function
End Module
```

Ouverture et fermeture d'une base commerciale

Une base comptable est toujours liée à une base commerciale.

Par conséquent, il est nécessaire d'instancier les classes *BSCPTAApplication100c* et *BSCIALApplication100c,* puis d'affecter les noms et chemins d’accès aux bases de données à l’aide de la propriété *Name* de chaque objet *(1a) (1b) (1c)*.

L'ouverture des bases comptable et commerciale étant soumise à une autorisation d'accès, il faut ensuite créer deux variables de type *IBILoggable* et affecter à ces variables les objets issus de l'instanciation de *BSCPTAApplication100c* et *BSCIALApplication100c (2a) (2b) (2c)*.

Si les fichiers sont protégés, il faudra renseigner les noms d'utilisateur et mots de passe (propriétés *UserName* et *UserPwd*) de chaque objet issus de *IBILoggable (3a) (3b)*.

Si aucune autorisation d'accès (autre que <Administrateur> *sans mot de passe) n'a été paramétrée au niveau des bases de données, il n'est pas nécessaire de renseigner ces propriétés.*

L'ouverture de la base commerciale (et de la base comptable liée) est effective après appel de la méthode *Open()* depuis l'objet issu de l'instanciation de *BSCIALApplication100c (4)*.

La fermeture d'une base commerciale (et de la base comptable liée) s'effectue en appelant la méthode *Close() (5)*.

Exemple :

```
Option Strict Off
Imports Objets100cLib
Imports System

Module OuvertureFermetureBaseCial

    Structure ParamBase
        Dim NomBase As String
        Dim Utilisateur As String
        Dim MotDePasse As String
    End Structure

    Sub Main()
        ' (1a)
        Dim BaseCial As New BSCIALApplication100c
        Dim BaseCpta As New BSCPTAApplication100c
        Dim ParamBaseCpta, ParamBaseCial As ParamBase
```

```
    With ParamBaseCpta
        .NomBase = "C:\temp\bijou.mae"
        .Utilisateur = "USER1"
        .MotDePasse = "1234"
    End With
    With ParamBaseCial
        .NomBase = "C:\temp\bijou.gcm"
        .Utilisateur = "USER1"
        .MotDePasse = "1234"
    End With
    If OuvreBaseCial(BaseCial, BaseCpta, ParamBaseCial, ParamBaseCpta) Then
        Console.WriteLine("Base commerciale ouverte !")
        If FermeBaseCial(BaseCial) Then
            Console.WriteLine("Base commerciale fermée !")
        End If
    End If
End Sub

Function OuvreBaseCial(ByRef BaseCial As BSCIALApplication100c, ByRef BaseCpta As BSCPTAApplication100c, _
    ByVal ParamBaseCial As ParamBase, ByVal ParamBaseCpta As ParamBase) As Boolean
    ' (2a)
    Dim LoginCpta, LoginCial As IBILoggable
    Try
        With ParamBaseCpta
            ' (1b)
            BaseCpta.Name = .NomBase
            ' (2b)
            LoginCpta = BaseCpta
            If Not .Utilisateur Is Nothing Then
                ' (3a)
                LoginCpta.UserName = .Utilisateur
                LoginCpta.UserPwd = .MotDePasse
            End If
```

```
        End With
        BaseCial.CptaApplication = BaseCpta
        With ParamBaseCial
            ' (1c)
            BaseCial.Name = .NomBase
            ' (2c)
            LoginCial = BaseCial
            If Not .Utilisateur Is Nothing Then
                ' (3b)
                LoginCial.UserName = .Utilisateur
                LoginCial.UserPwd = .MotDePasse
            End If
        End With
        ' (4)
        BaseCial.Open()
        Return True
    Catch ex As Exception
        Console.WriteLine("Erreur en ouverture de base commerciale : {0}", ex.Message)
        Return False
    End Try
End Function

Function FermeBaseCial(ByRef BaseCial As BSCIALApplication100c) As Boolean
    Try
        ' (5)
        BaseCial.Close()
        Return True
    Catch ex As Exception
        Console.WriteLine("Erreur en fermeture de base commerciale : {0}", ex.Message)
        Return False
    End Try
End Function
```

End Module

Manipulation des enregistrements des bases de données Sage 100c

Après avoir créé ou ouvert une base de données Sage 100c, il est possible d'y créer de nouveaux enregistrements ou d'accéder à des enregistrements existants.

A partir de Sage 100cloud Objets Métiers, chaque enregistrement d'une base de données est vu comme étant un objet. Pour accéder à un enregistrement de la base de données, il est donc nécessaire de fabriquer un objet par l'intermédiaire d'une propriété dite "Factory".

Les points suivants vont être détaillés ci-après :

- Les interfaces IBIPersistObject et IBITypeObjectFactory ;
- Création d'un nouvel objet ;
- Lecture d'un enregistrement ;
- Le cache en lecture ;
- Lecture d'un ensemble d'enregistrements : les collections d'objets ;
- Méthodes et propriétés des collections d'objets ;
- Enregistrement d'un objet dans la base de données ;
- La notion de persistance des objets ;
- Verrouillages en modification d'enregistrements ;
- Suppression d'un enregistrement ;
- Notifications réseau en modification et en suppression d'enregistrement ;
- Les liens entre les Objets Métiers ;
- Initialisation des objets métier.

Les interfaces IBIPersistObject et IBITypeObjectFactory

Au cours du chapitre précédent, il a été vu qu'une base de données Sage 100c est représentée par un objet issu de l'interface *IBIPersistStream*.

Sage 100cloud Objets Métiers met à disposition du développeur un grand nombre d'objets permettant l'accès aux différents enregistrements des bases de données Sage 100c.

Ces objets dérivent tous de deux interfaces principales :

- **IBIPersistObject** : les objets dérivés de cette interface correspondent à une représentation en mémoire des enregistrements de la base de données. Ces objets sont dits "persistants" car liés à un enregistrement de la base de données.
- **IBITypeObjectFactory** : les objets dérivés de cette interface permettent de fabriquer de nouveaux objets dérivés de l'interface *IBIPersistObject*. Ces objets sont dits "Factory".

L'interface IBIPersistObject

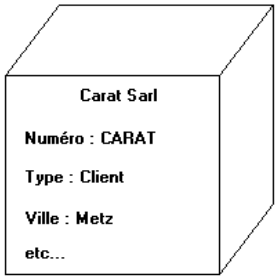
Une base de données est un ensemble de tables reliées entre elles. Chaque table est divisée en enregistrements, eux-mêmes constitués de champs.

Vue depuis Sage 100cloud Objets Métiers, une base de données est un ensemble d'objets représentant les différents enregistrements de cette base de données. Chacun de ces objets est issu de l'interface ***IBIPersistObject***.

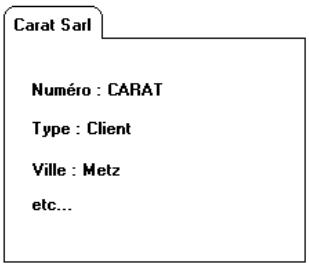
Les propriétés de ces objets représentent les différents champs des enregistrements de la base de données.

Exemple :

Un objet IBOTiers3*, issu de l'interface* IBIPersistObject*, correspond à un enregistrement de la table F_COMPTET (exemple : CARAT). Chacune des propriétés de l'objet* IBOTiers3 *correspond à un champ de l'enregistrement :*



Objet Tiers



Enregistrement Tiers

La création d'un nouvel enregistrement, ou l'accès à un enregistrement existant dans la base de données, s'effectue par l'intermédiaire de l'interface ***IBITypeObjectFactory*** implémentée par chaque objet.

L'interface IBITypeObjectFactory

La plupart des objets issus de l'interface *IBIPersistObject*, implémentent une interface *IBITypeObjectFactory*.

Cette interface se présente sous la forme de propriétés dites "propriétés Factory", correspondant à chaque type d'objet à créer.

Exemples :

- *La classe BSCPTAApplication100c dispose d'une propriété intitulée FactoryCompteG à partir de laquelle il est possible de lire, de créer, de modifier ou supprimer un compte général (interface IBOCompteG3) ;*
- *L'objet IBOTiers3 dispose d'une propriété intitulée FactoryTiersBanque à partir de laquelle il est possible de lire, de créer, de modifier ou supprimer une banque tiers (interface IBOTiersBanque3).*

Chaque propriété Factory dispose d'un certain nombre de méthodes et propriétés permettant principalement de créer, de lire ou d'écrire un enregistrement dans une base de données :

Méthode / propriété	Description
Create() As IBIPersistObject	Création d'un nouvel objet
List() AsIBICollection	Création d'une nouvelle collection d'objets
Read(ByVal pOID As IBIObjectID) As IBIPersistObject	Lecture d'un enregistrement correspondant à l'identifiant passé en paramètre. L'enregistrement est retourné sous forme d'un objet.

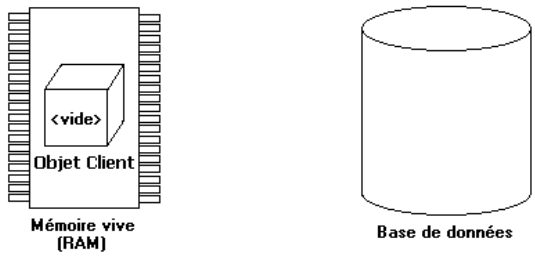
Création d'un nouvel objet métier

Les objets dérivés de l'interface *IBIPersistStream*, par l'intermédiaire de leurs propriétés *Factory,* proposent une méthode *Create()* permettant de créer de nouveaux objets.

Ces objets de premier niveau (issus de *IBIPersistStream*) constituent les **Objets Métiers** proprement dits.

Un objet venant d'être créé a les caractéristiques suivantes :

- Il est *non persistant* car il ne fait référence à aucun enregistrement de la base de données : il n'existe qu'en mémoire (Cf. *La notion de persistance des objets*).
- Ses propriétés sont vides ou initialisées avec des valeurs par défaut.



Exemple :

Création d'un objet Client (table F_COMPTET)

- *Déclaration d'une variable correspondant au type de donnée renvoyé par la méthode Create() (1).*
- *Appel de la méthode Create() retournant un nouvel objet stocké dans la variable venant d'être déclarée (2).*

```
Option Strict Off

Imports Objets100cLib

Imports System


Module CreationObjetClient


    Sub Main()

        Dim BaseCpta As New BSCPTAApplication100c

        ' (1)

        Dim ObjetClient As IBOClient3 = Nothing

        If OuvreBaseCpta(BaseCpta, "C:\temp\Bijou.mae") Then

            If CreeObjetClient(BaseCpta, ObjetClient) Then

                Console.WriteLine("Objet Client créé !")

            End If

            FermeBaseCpta(BaseCpta)

        End If

    End Sub


    Function CreeObjetClient(ByRef BaseCpta As BSCPTAApplication100c, ByRef ObjetClient As IBOClient3) As Boolean

        Try

            ' (2)

            ObjetClient = BaseCpta.FactoryClient.Create()

            Return True

        Catch ex As Exception

            Console.WriteLine("Erreur en création d'un nouvel objet Client : {0}", ex.Message)

            Return False

        End Try

    End Function
```

```
    Function OuvreBaseCpta(ByRef BaseCpta As BSCPTAApplication100c, _
     ByVal NomBaseCpta As String, Optional ByVal Utilisateur As String = "<Administrateur>", _
     Optional ByVal MotDePasse As String = "") As Boolean
        Try
            BaseCpta.Name = NomBaseCpta
            BaseCpta.Loggable.UserName = Utilisateur
            BaseCpta.Loggable.UserPwd = MotDePasse
            BaseCpta.Open()
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur en ouverture de base comptable : {0}", ex.Message)
            Return False
        End Try
    End Function

    Function FermeBaseCpta(ByRef BaseCpta As BSCPTAApplication100c) As Boolean
        Try
            BaseCpta.Close()
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur en fermeture de base comptable : {0}", ex.Message)
            Return False
        End Try
    End Function
End Module
```

Lecture d'un enregistrement

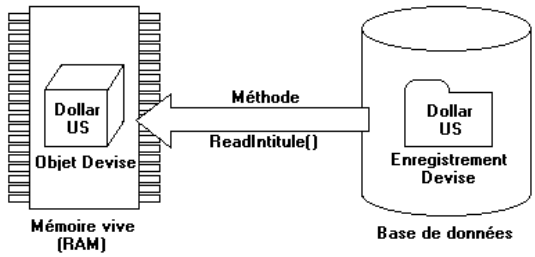
La lecture d'un enregistrement s'effectue par l'intermédiaire d'une méthode de type *Readxxx*().

> En fonction de la propriété *Factory* utilisée, il existe des variantes de la méthode *Readxxx()*.

Exemples :

Factory	Méthode	Description
IBITypeObjectFactory	Read(ByVal pOID As IBIObjectID) As IBIPersistObject	Retourne un objet IBIPersistObject correspondant à l'identifiant d'un objet persistant.
IBOArticleFactory3	ReadReference(ByVal sReference As String) As IBOArticle3	Retourne un objet IBOArticle3 correspondant à la référence article passée en paramètre.
IBOClientFactory3	ReadNumero(ByVal sNum As String) As IBOClient3	Retourne un objet IBOClient3 correspondant au numéro de client passé en paramètre.
IBPDevise2	ReadIntitule(ByVal sIntitule As String) As IBPDevise	Retourne un objet IBPDevise2 correspondant à l'intitulé de devise passé en paramètre.

L'objet créé par la méthode *Readxxx()* reprend sous forme de propriétés les différents champs de l'enregistrement lu :



Un enregistrement lu avec la méthode Read() n'est pas verrouillé. C'est-à-dire qu'il reste accessible à d'autres utilisateurs de la base de données.

Exemple :

Lecture d'une devise (table P_Devise*)*

- *Déclaration d'une variable correspondant au type de donnée renvoyé par la méthode Readxxx() (1).*
- *Appel de la méthode Readxxx() retournant l'enregistrement lu sous forme d'un objet stocké dans la variable venant d'être déclarée (2).*

```
Option Strict Off

Imports Objets100cLib
```

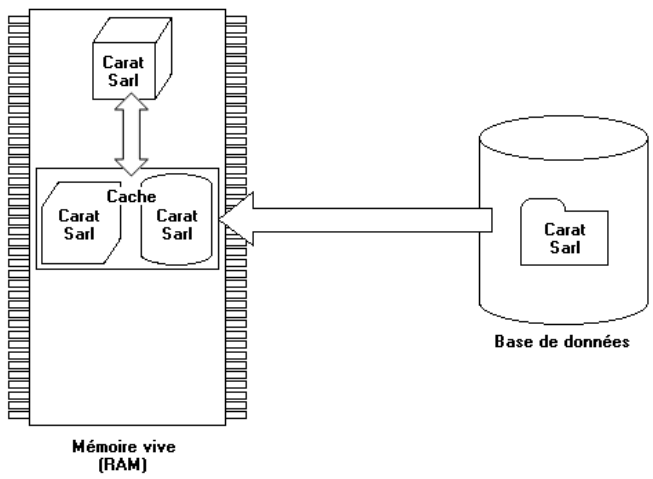
```
Imports System


Module LectureDevise
    Sub Main()
        Dim BaseCpta As New BSCPTAApplication100c
        Dim IntituleDevise As String
        ' (1)
        Dim Devise As IBPDevise2
        If OuvreBaseCpta(BaseCpta, "C:\temp\Bijou.mae") Then
            IntituleDevise = "Dollar US"
            Try
                ' (2)
                Devise = BaseCpta.FactoryDevise.ReadIntitule(IntituleDevise)
                Console.WriteLine("Lecture de l'enregistrement Devise {0} !", IntituleDevise)
            Catch ex As Exception
                Console.WriteLine("Erreur en lecture de l'enregistrement {0} de la table P_DEVISE : {1}", IntituleDevise, ex.Message)
            Finally
                FermeBaseCpta(BaseCpta)
            End Try
        End If
    End Sub


    Function OuvreBaseCpta(ByRef BaseCpta As BSCPTAApplication100c, _
  ByVal NomBaseCpta As String, Optional ByVal Utilisateur As String = "<Administrateur>", _
  Optional ByVal MotDePasse As String = "") As Boolean
        Try
            BaseCpta.Name = NomBaseCpta
            BaseCpta.Loggable.UserName = Utilisateur
            BaseCpta.Loggable.UserPwd = MotDePasse
            BaseCpta.Open()
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur en ouverture de base comptable : {0}", ex.Message)
```

```
        Return False
    End Try
End Function

Function FermeBaseCpta(ByRef BaseCpta As BSCPTAApplication100c) As Boolean
    Try
        BaseCpta.Close()
        Return True
    Catch ex As Exception
        Console.WriteLine("Erreur en fermeture de base comptable : {0}", ex.Message)
        Return False
    End Try
End Function

End Modul
```

Le cache en lecture

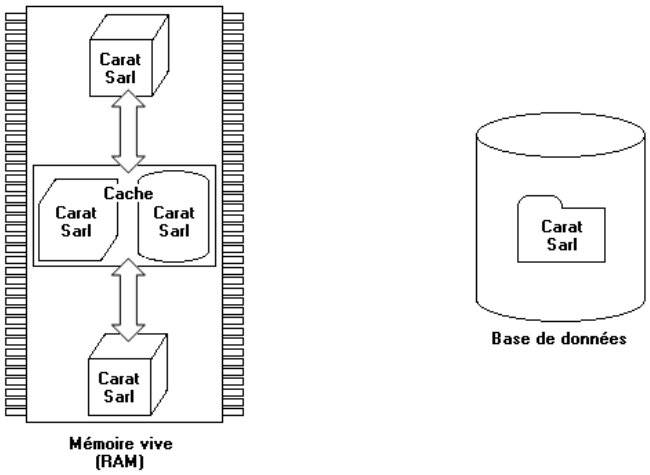
Lorsque qu'un enregistrement d'une base de données est lu par l'intermédiaire de Sage 100cloud Objets Métiers, un objet correspondant à cet enregistrement est créé en mémoire.

De plus, les identifiants de l'objet et de l'enregistrement sont stockés dans un cache en mémoire :



Si par la suite Sage 100cloud Objets Métiers accède au même enregistrement par l'intermédiaire d'un autre objet, le cache est consulté afin de contrôler qu'il n'existe pas déjà un objet faisant référence à ce même enregistrement.

Si c'est le cas, le nouvel objet est créé en reprenant les propriétés du premier objet déjà chargé en mémoire :



Les principaux avantages du cache sont les suivants :

- Accélération de la lecture des enregistrements qui existent déjà en mémoire sous forme d'objets ;
- Lorsque plusieurs objets font référence à un même enregistrement, leurs propriétés sont identiques ;
- Si les propriétés d'un objet sont modifiées, les propriétés des autres objets faisant référence au même enregistrement sont également modifiées.

Si l'enregistrement consulté n'existe pas en mémoire, l'enregistrement est relu depuis la base de données.

Il est possible de forcer la relecture de l'enregistrement en utilisant la méthode **Read**() disponible pour tous les objets de type **IBIPersistObject**. A l'appel de cette méthode, les données sont systématiquement relues dans la base pour mettre à jour le cache. A noter que les objets de type **IBIPersistOject** publient également une méthode **Refresh()** permettant de ne relire les données en base, que si celles-ci sont différentes de celles présentes dans le cache.Cette méthode permet donc de limiter les accès en base.

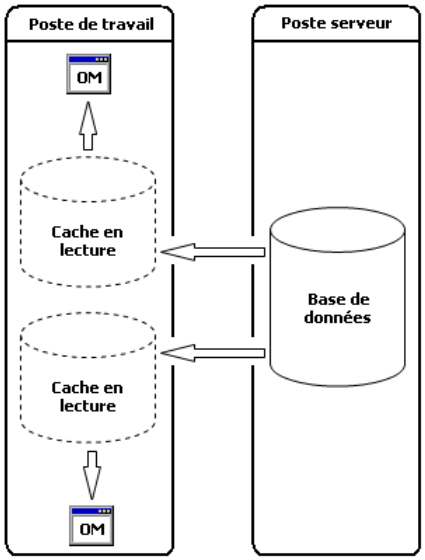
Les méthodes de type **Readxxx()** disponibles au niveau des objets de type **IBOxxxFactory,** accèdent aux données situées dans le cache. Par conséquent, en cas de modification d'un enregistrement par une autre application, il est nécessaire d'appeler la méthode **Read()** ou **Refresh()** de l'objet de type **IBIPersistObject()** pour mettre à jour le cache.

Particularités du cache en accès multi-utilisateurs

Lorsqu'une base de données est utilisée par plusieurs utilisateurs simultanément, il existe certaines particularités concernant la gestion du cache :

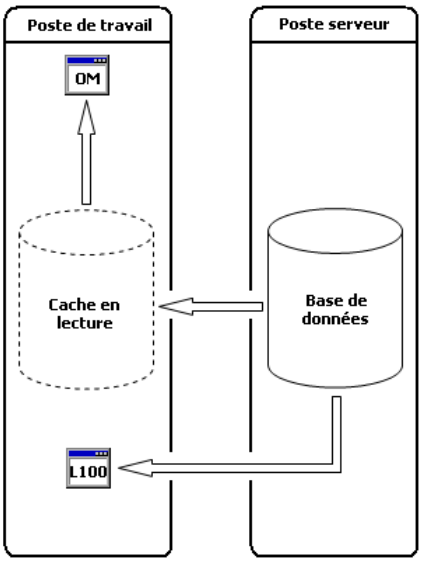
- Deux applications développées avec Sage 100cloud Objets Métiers sont exécutées sur un même poste de travail et accèdent à la même base de données.

  Il existe un cache distinct par Stream (représentation de la base de données sous forme d'objet par instanciation de *IBIPersistStream*). Par conséquent, deux applications utilisant Sage 100cloud Objets Métiers, situées sur le même poste de travail, ont chacune un cache distinct :



- Deux applications, une application Sage 100c et une application utilisant Sage 100cloud Objets Métiers, sont exécutées sur un même poste de travail ou sur deux postes de travail différents et accèdent à la même base de données ;

  Le cache n'est utilisé que par l'application développée avec Sage 100cloud Objets Métiers. L'application Sage 100c n'utilise jamais le cache. Par conséquent, seule l'application développée avec Sage 100cloud Objets Métiers accède au cache en lecture. L'application Sage 100c accède directement à la base de données sans passer par le cache :



Si l'application Sage 100c modifie ou supprime un enregistrement, elle le notifie (Cf. Notifications réseau en modification d'enregistrement et Notifications réseau en suppression d'enregistrement).

Lecture d'un ensemble d'enregistrements : les collections

Il est souvent nécessaire d'accéder non pas à un enregistrement particulier, mais à un ensemble d'enregistrements :

Exemples :

- *Affichage d'une liste de tiers, d'articles, de comptes ;*
- *Statistiques sur un ensemble d'écritures comptables ou des lignes de documents.*

Plutôt que de créer un objet par enregistrement, il est plus simple et rapide de générer une collection d'objets (interface *IBICollection*).

Une collection d'objets est un ensemble d'objets de même type. Elle peut être parcourue et chaque objet est accessible individuellement.

Sage 100cloud Objets Métiers propose, en fonction du type d'objet *Factory*, différentes méthodes ou propriétés permettant de créer des collections.

Exemples :

Factory	Méthode / propriété	Description
IBITypeObjectFactory	List() As IBICollection	Retourne une collection d'objets IBIPersistObject

IBOArticleFactory3	QueryFamille(ByVal pFamille As IBOFamille3) As IBICollection	Retourne une collection d'objets IBOArticle3 appartenant à la famille passée en paramètre.
IBOCompteGFactory3	ListOrderType() As IBICollection	Retourne une collection d'objets IBOCompteG3 classés par type.
IBOTiersFactory3	QueryTypeOrderNumero(ByVal sType As TiersType) As IBICollection	Retourne une collection d'objets IBOTiers3 d'un type donné classés par numéro.

Exemple :

Création d'une collection d'objets tiers

- *Déclaration d'une nouvelle variable correspondant à une collection d'objets (interface IBICollection) (1) ;*
- *Appel de la propriété List(). La totalité des enregistrements correspondant au type de Factory est retournée par cette propriété et est stockée dans la collection (2).*

```
Option Strict Off

Imports Objets100cLib

Imports System


Module CreationCollection


  Sub Main()

    Dim BaseCpta As New BSCPTAApplication100c

    ' (1)

    Dim CollTiers As IBICollection = Nothing

    If OuvreBaseCpta(BaseCpta, "C:\temp\Bijou.mae") Then

      If CreeCollTiers(BaseCpta, CollTiers) Then

        Console.WriteLine("Collection d'objets Tiers créée !")

      End If

      FermeBaseCpta(BaseCpta)

    End If

  End Sub


  Function CreeCollTiers(ByRef BaseCpta As BSCPTAApplication100c, ByRef CollTiers As IBICollection) As Boolean
```

```
    Try
        '(2)
        CollTiers = BaseCpta.FactoryTiers.List
        Return True
    Catch ex As Exception
        Console.WriteLine("Erreur en création d'une collection de tiers : {0}", ex.Message)
        Return False
    End Try
End Function

Function OuvreBaseCpta(ByRef BaseCpta As BSCPTAApplication100c, _
    ByVal NomBaseCpta As String, Optional ByVal Utilisateur As String = "<Administrateur>", _
    Optional ByVal MotDePasse As String = "") As Boolean
    Try
        BaseCpta.Name = NomBaseCpta
        BaseCpta.Loggable.UserName = Utilisateur
        BaseCpta.Loggable.UserPwd = MotDePasse
        BaseCpta.Open()
        Return True
    Catch ex As Exception
        Console.WriteLine("Erreur en ouverture de base comptable : {0}", ex.Message)
        Return False
    End Try
End Function

Function FermeBaseCpta(ByRef BaseCpta As BSCPTAApplication100c) As Boolean
    Try
        BaseCpta.Close()
        Return True
    Catch ex As Exception
        Console.WriteLine("Erreur en fermeture de base comptable : {0}", ex.Message)
        Return False
    End Try
```

```
    End Function
End Module
```

Méthodes et propriétés des collections d'objets

Les collections d'objets fournissent des méthodes et des propriétés simplifiant la gestion des objets contenus dans la collection.

Outre le parcours des collections par **For Each … In … Next**, les collections d'objets offrent les fonctionnalités suivantes :

- La propriété *Count()* qui renvoi le nombre d'éléments de la collection ;
- La méthode *Item(index)* qui permet d'accéder à un objet particulier de la collection en fonction de son index.

Exemple :

Affichage du nombre d'éléments d'une collection de tiers et affichage du dernier élément de la collection :

```
Option Strict Off
Imports Objets100cLib
Imports System

Module LectureCollection

    Sub Main()
        Dim BaseCpta As New BSCPTAApplication100c
        Dim CollTiers As IBICollection = Nothing
        Dim Tiers As IBOTiers3
        If OuvreBaseCpta(BaseCpta, "C:\temp\Bijou.mae") Then
            If CreeCollTiers(BaseCpta, CollTiers) Then
                Try
                    Console.WriteLine("La base de données contient {0} tiers.", CollTiers.Count)
                    Tiers = CollTiers.Item(CollTiers.Count)
                    Console.WriteLine("Le dernier tiers se nomme {0}.", Tiers.CT_Intitule.ToString)
```

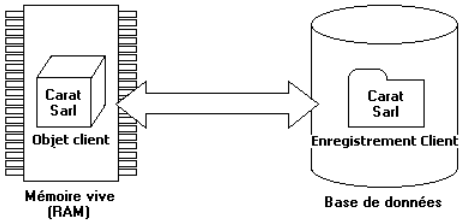
```
            Console.WriteLine(vbNewLine & "Liste des tiers :")
            For Each Tiers In CollTiers
                Console.WriteLine("{0}", Tiers.CT_Num)
            Next
        Catch ex As Exception
            Console.WriteLine("Erreur en lecture de la collection de tiers : {0}", ex.Message)
        End Try
    Else
        Console.WriteLine("La base de données ne contient aucun tiers.")
    End If
    FermeBaseCpta(BaseCpta)
  End If
End Sub

Function CreeCollTiers(ByRef BaseCpta As BSCPTAApplication100c, ByRef CollTiers As IBICollection) As Boolean
  Try
    '(2)
    CollTiers = BaseCpta.FactoryTiers.List
    Return True
  Catch ex As Exception
    Console.WriteLine("Erreur en création d'une collection de tiers : {0}", ex.Message)
    Return False
  End Try
End Function

Function OuvreBaseCpta(ByRef BaseCpta As BSCPTAApplication100c, _
  ByVal NomBaseCpta As String, Optional ByVal Utilisateur As String = "<Administrateur>", _
  Optional ByVal MotDePasse As String = "") As Boolean
  Try
    BaseCpta.Name = NomBaseCpta
    BaseCpta.Loggable.UserName = Utilisateur
    BaseCpta.Loggable.UserPwd = MotDePasse
    BaseCpta.Open()
```

```
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur en ouverture de base comptable : {0}", ex.Message)
            Return False
        End Try
    End Function

    Function FermeBaseCpta(ByRef BaseCpta As BSCPTAApplication100c) As Boolean
        Try
            BaseCpta.Close()
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur en fermeture de base comptable : {0}", ex.Message)
            Return False
        End Try
    End Function

End Module
```

Enregistrement d'un objet dans la base de données

L'enregistrement d'un objet dans la base de données s'effectue par l'intermédiaire de la méthode *Write()*. Cette méthode est disponible pour tous les objets implémentant l'interface *IBIPersistObject*.

Il existe une autre méthode similaire : WriteDefault(). *Son rôle est d'effectuer des traitements par défaut (exemple : création d'enregistrements liés dans d'autres tables) avant d'enregistrer l'objet dans la base de données (Cf.* Initialisation des objets métiers*).*



Lorsque l'objet est écrit dans la base de données, il devient persistant, c'est-à-dire qu'il est lié à un enregistrement de la base de données.

Exemple :

Enregistrement d'un objet Client dans la base de données.

```
Option Strict Off

Imports Objets100cLib

Imports System


Module EnregistrementObjet


  Sub Main()

    Dim BaseCpta As New BSCPTAApplication100c

    Dim ObjetClient As IBOClient3 = Nothing

    If OuvreBaseCpta(BaseCpta, "C:\temp\Bijou.mae") Then

      If CreeObjetClient(BaseCpta, ObjetClient) Then

        Console.WriteLine("Nouvel objet Client créé !")

        If EnregistreObjetClient(ObjetClient, "BOLLE", "Client BOLLE") Then

          Console.WriteLine("Objet Client {0} enregistré !", ObjetClient.CT_Num.ToString)

        End If

      End If

      FermeBaseCpta(BaseCpta)

    End If

  End Sub


  Function EnregistreObjetClient(ByRef ObjetClient As IBOClient3, ByVal NumCompteClient As String, _
```

```
    ByVal IntituleCompteClient As String) As Boolean
    Try
        ObjetClient.SetDefault()
        ObjetClient.CT_Num = NumCompteClient
        ObjetClient.CT_Intitule = IntituleCompteClient
        ObjetClient.Write()
        Return True
    Catch ex As Exception
        Console.WriteLine("Erreur en enregistrement d'un objet Client : {0}", ex.Message)
        Return False
    End Try
End Function

Function CreeObjetClient(ByRef BaseCpta As BSCPTAApplication100c, ByRef ObjetClient As IBOClient3) As Boolean
    Try
        ObjetClient = BaseCpta.FactoryClient.Create()
        Return True
    Catch ex As Exception
        Console.WriteLine("Erreur en création d'un nouvel objet Client : {0}", ex.Message)
        Return False
    End Try
End Function

Function OuvreBaseCpta(ByRef BaseCpta As BSCPTAApplication100c, _
 ByVal NomBaseCpta As String, Optional ByVal Utilisateur As String = "<Administrateur>", _
 Optional ByVal MotDePasse As String = "") As Boolean
    Try
        BaseCpta.Name = NomBaseCpta
        BaseCpta.Loggable.UserName = Utilisateur
        BaseCpta.Loggable.UserPwd = MotDePasse
        BaseCpta.Open()
        Return True
    Catch ex As Exception
```

```
            Console.WriteLine("Erreur en ouverture de base comptable : {0}", ex.Message)
            Return False
        End Try
    End Function

    Function FermeBaseCpta(ByRef BaseCpta As BSCPTAApplication100c) As Boolean
        Try
            BaseCpta.Close()
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur en fermeture de base comptable : {0}", ex.Message)
            Return False
        End Try
    End Function

End Module
```

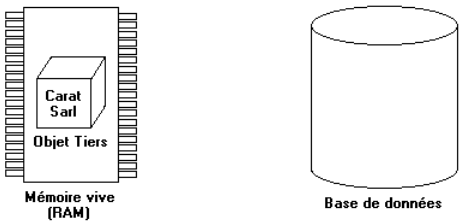
La notion de persistance des objets

Tout objet possède un état qui peut être *persistant* ou *non persistant*.

L'état d'un objet peut être connu en interrogeant sa propriété *IsPersistant()* qui renvoie un booléen.

Les objets non persistants

Un objet non persistant est un objet qui n'existe qu'en mémoire, c'est-à-dire qu'il n'est lié à aucun enregistrement de la base de données :

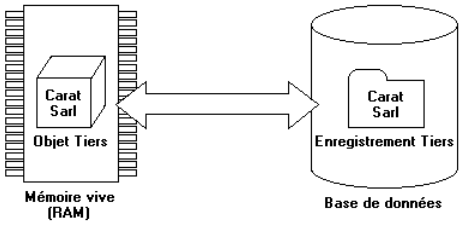


- Un objet est non persistant lorsqu'il vient d'être créé par une méthode *Create()*. Il ne fait donc référence à aucun enregistrement de la base de données.
- Un objet non persistant, ainsi que toutes ses propriétés, disparaît s'il n'est pas enregistré dans la base de données avant fermeture de l'application.

- Un objet passe de l'état non persistant à l'état persistant lorsqu'il a été écrit dans la base de données.

Les objets persistants

Un objet est persistant lorsqu'il est lié à un enregistrement de la base de données :



- Les objets issus de la lecture d'enregistrements de la base de données (méthodes *Read()*, *Query()*, propriété *List()*) sont par définition persistants ;
- Les objets non persistants deviennent persistants lorsqu'ils sont écrits dans la base de données (méthode *Write()*)

Particularité : Les Objets Métiers paramètre

Certains objets sont persistants dès leur création. Il s'agit des objets correspondant aux paramètres des fonctions de *Paramètres société* du menu *Fichier* des applications Sage 100c.

Ces objets se nomment *Objets Métiers paramètre* (interface *IBPxxx*) :

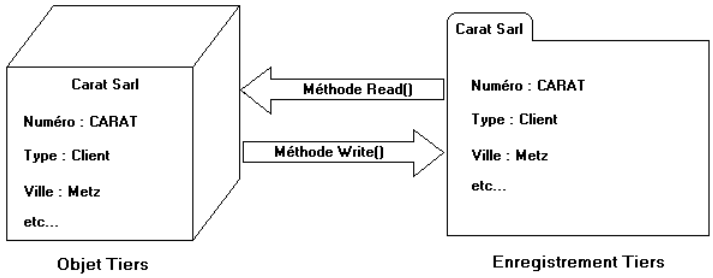
- les devises (interface *IBPDevise2*)
- les gammes (interface *IBPGamme*)
- les modes d'expédition (interface *IBPExpedition3*)
- etc…

Verrouillages en modification d'enregistrements

Bien que les différentes propriétés d'un objet persistant fassent référence aux champs de l'enregistrement lié, la modification des propriétés n'affecte pas immédiatement les champs de l'enregistrement.

Il est nécessaire d'utiliser la méthode *Write()* afin de répercuter les modifications sur l'enregistrement.

Inversement, si on souhaite forcer la relecture des champs de l'enregistrement, il est possible d'utiliser la méthode *Read()* :

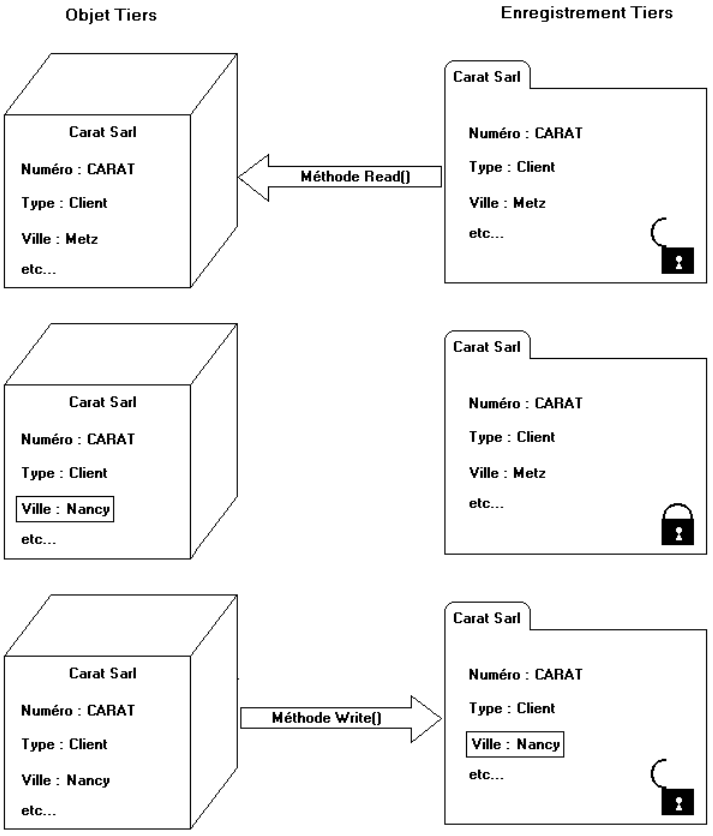


Lorsqu'une propriété d'un objet persistant est modifiée, l'enregistrement correspondant à cet objet est verrouillé, c'est-à-dire qu'il n'est plus accessible aux autres utilisateurs.

Ce verrou (en anglais : lock) est levé lorsque :

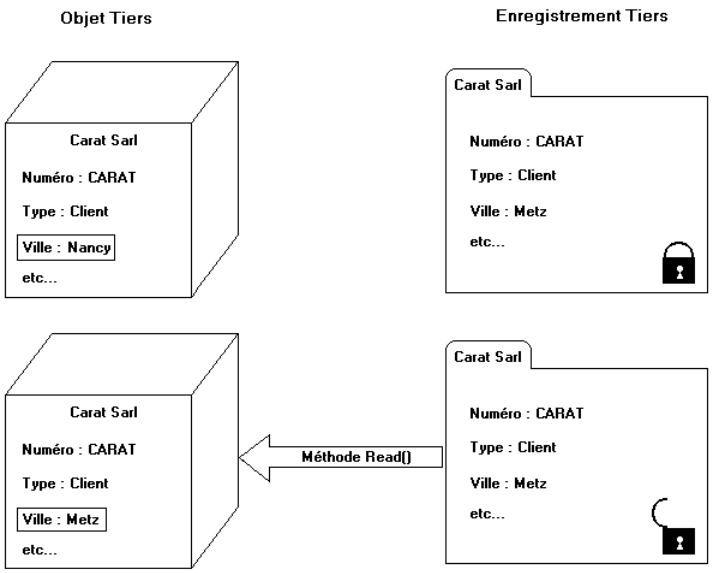
- L'objet est enregistré dans la base de données (méthode *Write()*) ;
- Ou lorsque l'enregistrement est relu (méthode *Read()*).

Par conséquent, afin de diminuer la fréquence des verrouillages, il est recommandé d'écrire les données dans la base de données immédiatement après leur modification :



En cas d'échec avant ou lors de l'écriture de l'objet dans la base de données, l'objet reste verrouillé. Par conséquent, les propriétés de l'objet diffèrent des champs de l'enregistrement correspondant.

Afin de déverrouiller et de réinitialiser les propriétés de l'objet avec les champs de l'enregistrement, il est recommandé d'utiliser la méthode *Read()* :



Exemple :

Modification de l'adresse d'un tiers
*Dans cet exemple, si une erreur survient lors de la modification d'une propriété de l'objet tiers, elle est interceptée afin de rétablir les propriétés initiales de l'objet. Pour ce faire, une procédure de relecture des données de l'enregistrement (*RelisEnreg*) utilisant la méthode* Read() *a été intégrée dans le bloc* Catch.

```
Option Strict Off

Imports Objets100cLib

Imports System


Module ModificationEnregistrement


    Structure Adresse

        Dim Adresse As String

        Dim Complement As String

        Dim CP As String

        Dim Ville As String

        Dim Region As String

        Dim Pays As String
```

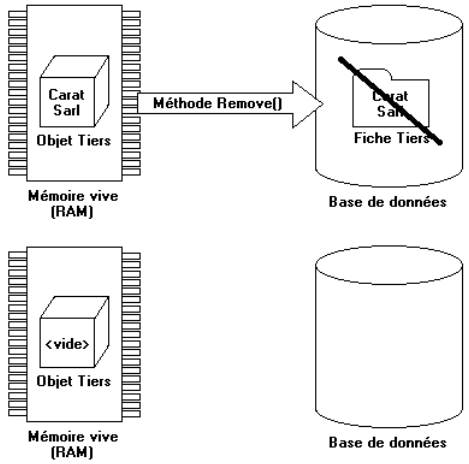
```
End Structure


Sub Main()
    Dim BaseCpta As New BSCPTAApplication100c
    Dim ObjetTiers As IBOTiers3 = Nothing
    Dim NouvAdresse As Adresse = Nothing
    If OuvreBaseCpta(BaseCpta, "C:\temp\Bijou.mae") Then
        If LisEnregTiers(BaseCpta, ObjetTiers, "SAGE") Then
            With NouvAdresse
                .Adresse = "1 rue Labrosse-Venner"
                .Complement = "BP 95231"
                .CP = "57070"
                .Ville = "St Julien lès Metz"
                .Region = "Lorraine"
            End With
            If ModifieAdresseTiers(ObjetTiers, NouvAdresse) Then
                Console.WriteLine("Adresse modifiée !")
            End If
        End If
        FermeBaseCpta(BaseCpta)
    End If
End Sub


Function LisEnregTiers(ByRef BaseCpta As BSCPTAApplication100c, ByRef Tiers As IBOTiers3, ByVal Numero As String) As
Boolean
    Try
        Tiers = BaseCpta.FactoryTiers.ReadNumero(Numero)
        Return True
    Catch ex As Exception
        Console.WriteLine("Erreur en lecture d'un enregistrement de la table F_COMPTET : {0}", ex.Message)
        Return False
    End Try
End Function
```

```
Function ModifieAdresseTiers(ByRef ObjetTiers As IBOTiers3, ByVal NouvAdresse As Adresse) As Boolean
    Try
        With NouvAdresse
            ObjetTiers.Adresse.Adresse = IIf(.Adresse Is Nothing, ObjetTiers.Adresse.Adresse, .Adresse)
            ObjetTiers.Adresse.Complement = IIf(.Complement Is Nothing, ObjetTiers.Adresse.Complement, .Complement)
            ObjetTiers.Adresse.CodePostal = IIf(.CP Is Nothing, ObjetTiers.Adresse.CodePostal, .CP)
            ObjetTiers.Adresse.Ville = IIf(.Ville Is Nothing, ObjetTiers.Adresse.Ville, .Ville)
            ObjetTiers.Adresse.CodeRegion = IIf(.Region Is Nothing, ObjetTiers.Adresse.CodeRegion, .Region)
            ObjetTiers.Adresse.Pays = IIf(.Pays Is Nothing, ObjetTiers.Adresse.Pays, .Pays)
            ObjetTiers.Write()
        End With
        Return True
    Catch ex As Exception
        Console.WriteLine("Erreur en modification d'adresse tiers : {0}", ex.Message)
        RelisEnreg(ObjetTiers)
        Return False
    End Try
End Function

Function RelisEnreg(ByRef Objet As IBIPersistObject) As Boolean
    Try
        If Objet.IsPersistant Then
            Objet.Read()
        End If
        Return True
    Catch ex As Exception
        Console.WriteLine("Erreur en lecture d'enregistrement : {0}", ex.Message)
        Return False
    End Try
End Function

Function OuvreBaseCpta(ByRef BaseCpta As BSCPTAApplication100c, _
 ByVal NomBaseCpta As String, Optional ByVal Utilisateur As String = "<Administrateur>", _
```

```
    Optional ByVal MotDePasse As String = "") As Boolean
        Try
            BaseCpta.Name = NomBaseCpta
            BaseCpta.Loggable.UserName = Utilisateur
            BaseCpta.Loggable.UserPwd = MotDePasse
            BaseCpta.Open()
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur en ouverture de base comptable : {0}", ex.Message)
            Return False
        End Try
    End Function

    Function FermeBaseCpta(ByRef BaseCpta As BSCPTAApplication100c) As Boolean
        Try
            BaseCpta.Close()
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur en fermeture de base comptable : {0}", ex.Message)
            Return False
        End Try
    End Function
End Module
```

Suppression d'un enregistrement

La suppression d'un enregistrement dans une base de données s'effectue en appelant la méthode *Remove()* disponible pour tous les objets implémentant l'interface *IBIPersistObject* :



:

L'objet qui faisait référence à l'enregistrement n'est pas supprimé. Il existe toujours mais il est vide (valeur *Nothing*) car il ne fait plus référence à aucun enregistrement.

Exemple :

Suppression d'un tiers :

```
Option Strict Off
Imports Objets100cLib
Imports System

Module SuppressionTiers

  Sub Main()
    Dim BaseCpta As New BSCPTAApplication100c
    If OuvreBaseCpta(BaseCpta, "C:\temp\Bijou.mae") Then
      If SupprimeTiers(BaseCpta, "ZAN") Then
        Console.WriteLine("Tiers supprimé !")
      End If
      FermeBaseCpta(BaseCpta)
    End If
  End Sub

  Function SupprimeTiers(ByRef BaseCpta As BSCPTAApplication100c, ByVal NumTiers As String) As Boolean
```

```
    Dim ObjetTiers As IBOTiers3
    Try
        ObjetTiers = BaseCpta.FactoryTiers.ReadNumero(NumTiers)
        ObjetTiers.Remove()
        Return True
    Catch ex As Exception
        Console.WriteLine("Erreur en suppression de l'enregistrement {0} dans la table F_COMPTET : {1}", NumTiers, ex.Message)
        Return False
    End Try
End Function

Function OuvreBaseCpta(ByRef BaseCpta As BSCPTAApplication100c, _
 ByVal NomBaseCpta As String, Optional ByVal Utilisateur As String = "<Administrateur>", _
 Optional ByVal MotDePasse As String = "") As Boolean
    Try
        BaseCpta.Name = NomBaseCpta
        BaseCpta.Loggable.UserName = Utilisateur
        BaseCpta.Loggable.UserPwd = MotDePasse
        BaseCpta.Open()
        Return True
    Catch ex As Exception
        Console.WriteLine("Erreur en ouverture de base comptable : {0}", ex.Message)
        Return False
    End Try
End Function

Function FermeBaseCpta(ByRef BaseCpta As BSCPTAApplication100c) As Boolean
    Try
        BaseCpta.Close()
        Return True
    Catch ex As Exception
        Console.WriteLine("Erreur en fermeture de base comptable : {0}", ex.Message)
        Return False
```

```
    End Try
  End Function

End Module
```

En VB .Net, il est possible d'éviter l'utilisation de la variable temporaire *ObjetTiers*, en remplaçant les instructions suivantes :

```
Dim ObjetTiers As IBOTiers3

ObjetTiers = BaseCpta.FactoryTiers.ReadNumero(NumTiers)

ObjetTiers.Remove()
```

Par cette ligne d'instructions :

```
CType(BaseCpta.FactoryTiers.ReadNumero(NumTiers), IBOTiers3).Remove()
```

Notifications réseau en modification et en suppression d'enregistrement

Il a été vu précédemment qu'une application Sage 100c (Comptabilité, Gestion commerciale) n'utilise pas le cache en lecture d'une application utilisant Sage 100cloud Objets Métiers (Cf. *Particularités du cache en accès multi-utilisateurs*).

Par conséquent, lorsqu'une application Sage 100c modifie ou supprime un enregistrement, le cache ne reflète plus l'état ou la valeur des champs de l'enregistrement.

Pour indiquer que la valeur contenue dans le cache ne correspond plus à l'enregistrement, un mécanisme de notification a été mis en place.

On distingue deux cas de figure :

- Lorsqu'une application Sage 100c modifie un enregistrement, elle envoie une notification à l'objet correspondant à l'enregistrement qui passe alors à l'état "modifié".

  Si par la suite, on tente de modifier une propriété de l'objet ou de l'écrire dans la base de données (méthode *Write()*), une exception est déclenchée.

- Lorsqu'une application Sage 100c supprime un enregistrement, elle envoie une notification à l'objet correspondant à l'enregistrement qui passe alors à l'état "supprimé".

  Si par la suite, on tente de modifier une propriété de l'objet, de le lire (méthode *Read()*) ou de l'écrire dans la base de données (méthode *Write()*), une exception est déclenchée.

Le traitement de ces exceptions peut s'effectuer de la façon suivante :

- Si l'enregistrement correspondant à l'objet existe dans la base de données, il faut forcer la relecture de l'enregistrement (appel de la méthode *Read()*) afin de réinitialiser le cache ;

- Si l'enregistrement correspondant à l'objet n'existe plus dans la base de données, il faut supprimer la référence à l'objet (affectation de *Nothing* à l'objet).

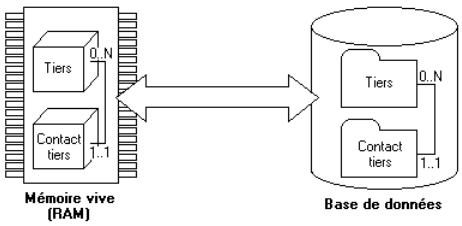
Exemple :

```
Public Sub MAJCacheTiers(ByRef ObjTiers As IBOTiers3)
    Try
      If CType(ObjTiers.Stream, BSCPTAApplication100c).FactoryTiers.ExistNumero(ObjTiers.CT_Num) Then
        ObjTiers.Read()
      Else
        ObjTiers = Nothing
      End If
    Catch ex As Exception
      Console.WriteLine("Erreur : " & ex.Message)
    End Try
  End Sub
```

Les liens entre les objets

Les différentes tables d'une base de données sont liées entre elles par des clés primaires et étrangères.

Par conséquent, les objets (correspondant à des enregistrements dans les tables) sont eux-mêmes liés à d'autres objets (correspondant à des enregistrements dans les tables liées).

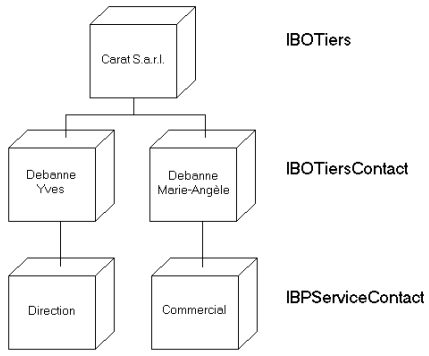


Lorsque deux objets sont liés, on distingue "l'objet maître" du "sous objet" ; un sous-objet ne peut être créé qu'à partir d'un objet maître.

La fabrication de sous objets s'effectue par l'intermédiaire de propriétés *Factory* disponibles au niveau des objets maîtres. Le principe est identique aux propriétés *Factory* des classes *Application*.

Généralement, un sous objet est lui-même l'objet maître d'un autre sous objet.

Exemple :



Un sous objet ne peut-être créé que si son objet maître est persistant. Toute tentative de création d'un sous objet pour un objet maître non persistant se solde par le message d'erreur suivant : L'objet n'est pas persistant ! Il existe toutefois une particularité pour le processus de création de document (**IPMDocument**). En effet, lors de la création d'un document par le biais de ce processus, il est possible d'ajouter des lignes de documents (**IBODocumentLigne3**) à l'objet maître (**IBODocument3)** alors que celui-ci n'est pas persistant.

Exemple :

Ajout d'un contact à un tiers existant.
L'exemple ci-dessous, illustre la liaison d'un sous objet à un objet maître par l'ajout d'un contact à un tiers. L'objet "Contact" est lui-même l'objet maître auquel le sous objet "Service contact" est rattaché.

```
Option Strict Off

Imports Objets100cLib

Imports System


Module CreationContactTiers


    Structure InfoContact

        Dim Nom, Prenom, Fonction, NumTel, NumPort, NumTelecop, Email As String

        Dim Service As IBPServiceContact

    End Structure
```

```
Sub Main()
    Dim BaseCpta As New BSCPTAApplication100c
    Dim Tiers As IBOTiers3 = Nothing
    Dim Contact As IBOTiersContact3 = Nothing
    Dim InfoContact As InfoContact = Nothing
    Dim NumTiers As String

    NumTiers = "CARAT"
    If OuvreBaseCpta(BaseCpta, "C:\temp\Bijou.mae") Then
        If LisEnregTiers(BaseCpta, Tiers, NumTiers) Then
            If CreeObjetContactTiers(Tiers, Contact) Then
                Console.WriteLine("Nouvel objet contact créé pour le tiers {0} !", NumTiers)
                With InfoContact
                    .Nom = "Vinson"
                    .Prenom = "Virginie"
                    .Fonction = "Comptable"
                    .NumTel = "12 34 56 78 90"
                    .NumPort = "06 12 34 56 78"
                End With
                If AffecteServiceContact(BaseCpta, InfoContact.Service, "Direction") Then
                    If EnregistreContactTiers(Contact, InfoContact) Then
                        Console.WriteLine("Contact {0} {1} enregistré dans la base de données !", InfoContact.Nom, InfoContact.Prenom)
                    End If
                End If
            End If
        End If
        FermeBaseCpta(BaseCpta)
    End If
End Sub

Function CreeObjetContactTiers(ByVal ObjetTiers As IBOTiers3, ByRef ContactTiers As IBOTiersContact3) As Boolean
    Try
```

```
            ContactTiers = ObjetTiers.FactoryTiersContact.Create
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur en création d'un nouveau contact tiers : {0}", ex.Message)
            Return False
        End Try
    End Function

    Function AffecteServiceContact(ByRef BaseCpta As BSCPTAApplication100c, ByRef Service As IBPServiceContact, ByVal Intitule
As String) As Boolean
        Try
            Service = BaseCpta.FactoryServiceContact.ReadIntitule(Intitule)
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur en affectation du service {0} à un contact : {1}", Intitule, ex.Message)
            Return False
        End Try
    End Function

    Function EnregistreContactTiers(ByVal Contact As IBOTiersContact3, ByVal InfoContact As InfoContact) As Boolean
        Try
            With InfoContact
                Contact.Nom = IIf(.Nom Is Nothing, Contact.Nom, .Nom)
                Contact.Prenom = IIf(.Prenom Is Nothing, Contact.Prenom, .Prenom)
                Contact.ServiceContact = IIf(.Service Is Nothing, Contact.ServiceContact, .Service)
                Contact.Fonction = IIf(.Fonction Is Nothing, Contact.Fonction, .Fonction)
                Contact.Telecom.Telephone = IIf(.NumTel Is Nothing, Contact.Telecom.Telephone, .NumTel)
                Contact.Telecom.Portable = IIf(.NumPort Is Nothing, Contact.Telecom.Portable, .NumPort)
                Contact.Telecom.Telecopie = IIf(.NumTelecop Is Nothing, Contact.Telecom.Telecopie, .NumTelecop)
                Contact.Telecom.EMail = IIf(.Email Is Nothing, Contact.Telecom.EMail, .Email)
                Contact.Write()
            End With
            Return True
        Catch ex As Exception
```

```
            Console.WriteLine("Erreur en enregistrement du contact tiers {0} {1} : {0}", InfoContact.Nom, InfoContact.Prenom, ex.Message)
            RelisEnreg(Contact)
            Return False
        End Try
    End Function

    Function OuvreBaseCpta(ByRef BaseCpta As BSCPTAApplication100c, _
     ByVal NomBaseCpta As String, Optional ByVal Utilisateur As String = "<Administrateur>", _
     Optional ByVal MotDePasse As String = "") As Boolean
        Try
            BaseCpta.Name = NomBaseCpta
            BaseCpta.Loggable.UserName = Utilisateur
            BaseCpta.Loggable.UserPwd = MotDePasse
            BaseCpta.Open()
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur en ouverture de base comptable : {0}", ex.Message)
            Return False
        End Try
    End Function

    Function FermeBaseCpta(ByRef BaseCpta As BSCPTAApplication100c) As Boolean
        Try
            BaseCpta.Close()
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur en fermeture de base comptable : {0}", ex.Message)
            Return False
        End Try
    End Function

    Function LisEnregTiers(ByRef BaseCpta As BSCPTAApplication100c, ByRef Tiers As IBOTiers3, ByVal Numero As String) As
Boolean
        Try
```

```
            Tiers = BaseCpta.FactoryTiers.ReadNumero(Numero)
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur en lecture d'un enregistrement de la table F_COMPTET : {0}", ex.Message)
            Return False
        End Try
    End Function

    Function RelisEnreg(ByRef Objet As IBIPersistObject) As Boolean
        Try
            If Objet.IsPersistant Then
                Objet.Read()
            End If
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur en lecture d'enregistrement : {0}", ex.Message)
            Return False
        End Try
    End Function
End Module
```

Initialisation des objets

Lorsqu'un nouvel objet est créé, ses différentes propriétés sont initialisées avec des valeurs par défaut.

Il est ensuite possible d'initialiser à tout moment les propriétés des objets persistants ou non persistants par invocation de la méthode *SetDefault()*.

Lors de l'enregistrement d'un nouvel objet ou de la mise à jour dans la base de données, les méthodes *Write()* et *WriteDefault()* réalisent des traitements d'initialisation de propriétés ou de création d'enregistrements.

Initialisation des objets lors de leur création

Lors de leur création, les objets sont généralement initialisés avec les valeurs par défaut suivantes :

Contrôle Windows dans l'application	Type de propriété	Valeur par défaut de la propriété
Calendrier	Date	Date du jour ou 1ere date autorisée.
Case à cocher	Booléen (Boolean)	False
Liste déroulante	Enumérateur	1er membre de l'énumérateur
Liste déroulante	Sous-objet (IBIxxx, IBOxxx, IBPxxx, etc...)	Nothing
Zone de saisie	Chaîne de caractère (String)	""
Zone de saisie	Nombre ou valeur (Short, Integer, Double, etc...)	0 ou première valeur autorisée.

Cette liste est non exhaustive. En fonction du contexte (ex : héritage de propriétés d'autres objets), certaines propriétés peuvent être initialisées de façon différente.

La méthode SetDefault()

Cette méthode peut être invoquée à tout moment depuis un objet persistant ou non persistant.

Son rôle est d'initialiser certaines propriétés à condition que ces propriétés contiennent une valeur par défaut.

Les propriétés ne contenant aucune valeur par défaut, c'est-à-dire les propriétés auxquelles une valeur a été affectée, ne sont pas initialisées par la méthode *SetDefault()*.

Exemple :

L'invocation de la méthode SetDefault() *depuis un objet de type* IBOTiers3 *initialise les champs suivants :*

Propriété	Valeur	Description
CT_Intitule() As String	Si CT_Intitule = "" Alors CT_Intitule = "Tiers à créer"	Affectation de la propriété Numéro du tiers à la propriété Intitulé du tiers si celle-ci n'est pas renseignée.
TiersPayeur() As IBOTiers	Si TiersPayeur = Nothing Alors TiersPayeur = Tiers	Affectation de l'objet Tiers à la propriété TiersPayeur si celle-ci n'est pas renseignée.

CompteGPrinc() As IBOCompteG	Si CompteGPrinc = Nothing Alors CompteGPrinc = Compte collectif par défaut	Affectation du compte collectif par défaut (situé dans Paramètres société / Tiers / Compte collectif) à la propriété Compte général principal si celle-ci n'est pas renseignée.

- Voir les Annexes pour le détail des traitements effectués par la méthode SetDefault() de chaque objet.

La méthode WriteDefault()

En fonction de l'objet appelant, la méthode *WriteDefault()* exécute les traitements suivants :

- Héritage de propriétés d'autres objets.

  Exemple : Appelée depuis un objet issu de l'interface *IBOArticle3*, la méthode *WriteDefault()* permet d'hériter des tarifs de la famille associée à l'article.

- Création d'enregistrements liés à l'objet.

  Exemple : Appelée depuis un objet issu de l'interface *IBOClient3*, la méthode *WriteDefault()* créé automatiquement une adresse de livraison principale.

- Modification de propriétés d'enregistrements liés à l'objet.

  Exemple : Appelée depuis un objet issu de l'interface *IBOEcriture3*, la méthode *WriteDefault()* proratise les valeurs des écritures analytiques en cas de modification du montant de l'écriture générale.

Exemple :

Recalcul automatique des valeurs des écritures analytiques.
L'utilisation de la méthode WriteDefault() *permet par exemple, d'automatiser certaines opérations de calcul. Ceci simplifie considérablement le développement d'applications.*
Ainsi, lorsque le montant d'une ligne d'écriture comptable est modifié alors qu'une répartition analytique existe pour cette ligne, il est possible de répercuter automatiquement cette modification sur les valeurs de l'analytique en utilisant la méthode WriteDefault()*.*
Cette méthode recalcule les valeurs en proratisant le montant de la ligne d'écriture générale en fonction des anciennes valeurs de l'analytique.

En Comptabilité, les lignes d'écriture suivantes sont saisies dans le journal de vente (VTE) de mars 2012 :

Jour	N° compte général	N°compte tiers	Débit	Crédit
1	4110000	CARAT	1 000,00	

1	**701020**			829,19
1	4457120			170,81

La répartition analytique suivante est effectuée sur la ligne d'écriture 701020 (plan Activité) :

Section	Montant
921SI1	290,22
922LY1	331,68
922ME1	207,29

Le montant des écritures générales est ensuite augmenté de 10 % :

Jour	N° compte général	N°compte tiers	Débit	Crédit
1	4110000	CARAT	1 100,00	
1	**701020**			912,11
1	4457120			187,89

*L'appel de la méthode **WriteDefault()** répercute la modification du montant de la ligne d'écriture 701020 sur les sections analytiques suivantes :*

Section	Montant
921SI1	319,24
922LY1	364,85
922ME1	228,02

Code source :

```
Option Strict Off

Imports Objets100cLib

Imports System


Module RecalculAnalytique


    Sub Main()

        Dim BaseCpta As New BSCPTAApplication100c

        Dim Ecriture As IBOEcriture3

        If OuvreBaseCpta(BaseCpta, "C:\temp\Bijou.mae") Then

            Try

                For Each Ecriture In BaseCpta.FactoryEcriture.QueryJournalPeriode(BaseCpta.FactoryJournal.ReadNumero("VTE"),
#3/1/2012#)
```

```
            With Ecriture
                If .FactoryEcritureA.List.Count > 0 Then
                    Console.WriteLine("Répartition analytique")
                    Console.WriteLine(vbNewLine & "Avant modification du montant ({0}):", .EC_Montant)
                    AfficheRepartAna(Ecriture)
                    .EC_Montant = .EC_Montant * 1.1
                    .WriteDefault()
                    Console.WriteLine(vbNewLine & "Après modification du montant ({0}):", .EC_Montant)
                    AfficheRepartAna(Ecriture)
                Else
                    .EC_Montant = .EC_Montant + 1.1
                    .Write()
                End If
            End With
        Next
    Catch ex As Exception
        Console.WriteLine(ex.Message)
    End Try
    FermeBaseCpta(BaseCpta)
    Console.ReadLine()
  End If
End Sub

Sub AfficheRepartAna(ByVal Ecriture As IBOEcriture3)
  Dim SectionAna As IBOEcritureA3
  For Each SectionAna In Ecriture.FactoryEcritureA.List
    Console.WriteLine("{0} : {1}", SectionAna.CompteA.CA_Num, SectionAna.EA_Montant)
  Next
End Sub

Function OuvreBaseCpta(ByRef BaseCpta As BSCPTAApplication100c, _
 ByVal NomBaseCpta As String, Optional ByVal Utilisateur As String = "<Administrateur>", _
 Optional ByVal MotDePasse As String = "") As Boolean
```

```
        Try
            BaseCpta.Name = NomBaseCpta
            BaseCpta.Loggable.UserName = Utilisateur
            BaseCpta.Loggable.UserPwd = MotDePasse
            BaseCpta.Open()
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur en ouverture de base comptable : {0}", ex.Message)
            Return False
        End Try
    End Function

    Function FermeBaseCpta(ByRef BaseCpta As BSCPTAApplication100c) As Boolean
        Try
            BaseCpta.Close()
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur en fermeture de base comptable : {0}", ex.Message)
            Return False
        End Try
    End Function

End Module
```

Utilisation des énumérateurs

Sage 100cloud Objets Métiers propose un grand nombre de constantes regroupées sous formes d'énumérateurs.

Il est recommandé d'utiliser ces constantes d'énumérateurs plutôt que les valeurs numériques correspondant à celles-ci. En effet, les valeurs des constantes sont susceptibles d’être modifiées au cours de l'évolution des produits Sage 100c. Les intitulés des constantes sont quant à eux fixes, ainsi ils ne seront pas modifiés dans les futures versions de Sage 100cloud Objets Métiers ; tout au plus, de nouvelles constantes pourraient apparaître. Pour cette raison, il est déconseillé d’utiliser les constantes des énumérateurs pour effectuer des opérations arithmétiques.

Les différents énumérateurs proposés par Sage 100cloud Objets Métiers sont détaillés dans la section *Annexes – Enumérateurs*.

Processus

Comme évoqué précedemment, un processus est une entité métier permettant de gérer certains automatismes sur un ou une collection d'enregistrements. Suivant le type de processus, les données manipulées peuvent être persistantes ou non persistantes.

Par exemple, le processus de création de documents (*IPMDocument*) manipule des documents et lignes de documents non persistants, alors que le processus de transformation de documents (*IPMDocTransformer*) ne pourra s'appliquer que sur des documents et lignes de documents persistants.

La validation d'un processus avec insertion des données réelles en base ne s'effectue pas en invoquant les méthodes *Write()* et *WriteDefault()* des interfaces manipulées dans le processus. Cette validation s'effectue à l'appel de la méthode *Process()*. De plus, tous les processus publient une méthode permettant de valider la cohérence des données avant de les écrire en base, il s'agit de la méthode *CanProcess()*.

Exemple : Saisie de pièce comptable - IPMEncoder

Le processus de saisie de pièce comptable permet d'insérer une collection d'écritures comptables en une seule fois. Ceci permet, à la différence d'insertion d'écritures directement à partir d'objets *IBOEcriture3*, de garantir que les écritures insérées par le processus ne pourront pas générer de déséquilibre de la période, et qu'aucunes écritures insérées par une autre application ne pourront s'intercaler entre deux écritures du processus.

Ce processus publie également certaines méthodes permettant de générer automatiquement, les écritures analytiques, les écritures d'échéances des comptes tiers ainsi que les écritures HT, TTC, TVA et solde (*AddTiersPart*, *Equilibrer*) en fonction de certains paramètres.

Vous retrouverez la description des propriétés et méthodes publiées par le processus *IPMEncoder* sous le paragraphe *Annexes / Objets100cLib.xxxx.Stream.1 / Interfaces Processus* de ce document.

Le processus *IPMEncoder* est composé de propriétés globales, d'éléments écritures et de méthodes permettant de parcourir les collections d'écritures avant et après insertion des données dans la base.

Propriétés globales

Les propriétés globales sont des propriétés qui sont reportées automatiquement sur les éléments écritures ajoutés au processus. Elles sont les suivantes :

- Journal,
- Date,
- Devise,
- EC_Parite,
- EC_Intitule,
- EC_Piece,
- EC_RefPiece,
- EC_Reference.

Pour ajouter des éléments écritures au processus, il faut obligatoirement renseigner les propriétés *Journal* et *Date*.

Eléments écritures

Les éléments écritures d'un processus sont des objets *IBOEcriture3* non persistants (n'existe qu'en mémoire). Les propriétés publiées par les éléments écritures du processus Saisie de pièce comptable sont donc les mêmes que ceux de l'objet *IBOEcriture3*. Cependant, certaines propriétés sont non modifiables car elles sont définies sur les propriétés globales du processus. Ces propriétés sont les suivantes :

- Journal,
- Date,
- Devise,
- Parite.

Collections d'écritures

Avant validation du processus, la collection des écritures ajoutées au processus peut être récupérée par l'appel de la propriété *FactoryEcritureIn.List*. Cette collection est une collection d'écritures non persistantes dans la base.

Lorsque les écritures sont insérées dans la base (appel de la méthode *Process()*), la collection d'écritures persistantes peut être récupérée par l'appel de la propriété *ListEcrituresOut()*.

Collection d'erreur

Lors de l'appel des méthodes *CanProcess()* et *Process()*, une collection d'erreurs est générée dans le cas où les écritures du processus présentent certaines incohérences (champs obligatoires

non renseignés, pièce déséquilibrée…). Cette collection peut être parcourue par l'intermédiaire de l'interface *IFailInfoCol*.

Exemple d'utilisation : création d'un règlement

*L'exemple suivant permet d'insérer une pièce comptable de règlement d'une facture dans un journal de banque. Cet exemple met également en œuvre une méthode permettant de récupérer la collection des éventuelles erreurs (*RecupErrors()*).*

Code source

```
Option Strict Off

Imports Objets100cLib

Imports System


Module ModEncoder

    'Objet base comptable

    Dim oCpta As BSCPTAApplication100c

    'emplacement du fichier comptable

    Dim sPathMae As String = "C:\temp\Bijou.mae"


    Sub Main()

        Try

            'Instanciation de l'objet base comptable

            oCpta = New BSCPTAApplication100c

            'Ouverture de la base

            If OpenBase(oCpta, sPathMae) Then

                'Création d'un objet processus Saisie de pièce comptable

                Dim mProcessEncoder As IPMEncoder = oCpta.CreateProcess_Encoder

                'Génération de la pièce

                ReglementFacture(mProcessEncoder)

                'Si les écritures du processus ne sont pas correctes

                If Not mProcessEncoder.CanProcess Then

                    'Récupération des erreurs

                    RecupError(mProcessEncoder)

                Else

                    'Gérération de la pièce dans la base
```

```
                mProcessEncoder.Process()
            End If
        End If
    Catch ex As Exception
        Console.WriteLine("Erreur : " & ex.Message)
    End Try
    CloseBase(oCpta)
End Sub

Public Sub ReglementFacture(ByRef mP As IPMEncoder)
    Dim mEcTiers, mEcCompte As IBOEcriture3
    Dim dMnt As Double = 10000
    Try
        'Affectation des propriétés globales
        'Journal
        mP.Journal = oCpta.FactoryJournal.ReadNumero("BEU")
        'Date
        mP.Date = #3/22/2009#
        'Numéro de pièce
        mP.EC_Piece = mP.Journal.NextEC_Piece(#3/22/2009#)
        'Intitulé
        mP.EC_Intitule = "Reglement Facture"

        'Création de l'écriture tiers
        mEcTiers = mP.FactoryEcritureIn.Create
        'Affectation compte général
        mEcTiers.CompteG = oCpta.FactoryCompteG.ReadNumero("4010000")
        'Affectation compte tiers
        mEcTiers.Tiers = oCpta.FactoryTiers.ReadNumero("HOLDI")
        'Affectation sens de l'écriture
        mEcTiers.EC_Sens = EcritureSensType.EcritureSensTypeDebit
        'Affectation du montant
        mEcTiers.EC_Montant = dMnt
```

```
        'Ajout de l'écriture au processus (écriture mémoire non persistante)
        mEcTiers.WriteDefault()

        'Création de l'écriture générale
        mEcCompte = mP.FactoryEcritureIn.Create
        'Affectation compte général
        mEcCompte.CompteG = oCpta.FactoryCompteG.ReadNumero("5120")
        'Affectation sens de l'écriture
        mEcCompte.EC_Sens = EcritureSensType.EcritureSensTypeCredit
        'Affectation du montant
        mEcCompte.EC_Montant = dMnt
        'Ajout de l'écriture au processus (écriture mémoire non persistante)
        mEcCompte.WriteDefault()
    Catch ex As Exception
        Console.WriteLine("Erreur : " & ex.Message)
    End Try
End Sub

Public Sub RecupError(ByRef mP As IPMEncoder)
    Try
        'Boucle sur les erreurs contenues dans la collection
        For i As Integer = 1 To mP.Errors.Count
            'Récupération des éléments erreurs
            Dim iFail As IFailInfo = mP.Errors.Item(i)
            'récupération du numéro d'erreur, de l'indice et de la description de l'erreur
            Console.WriteLine("Code Erreur : " & iFail.ErrorCode & " Indice : " & iFail.Indice & _
            " Description : " & iFail.Text)
        Next
    Catch ex As Exception
        Console.WriteLine("Erreur : " & ex.Message)
    End Try
End Sub
```

```
    Public Function OpenBase(ByRef BaseCpta As BSCPTAApplication100c, ByVal sMae As String, _
    Optional ByVal sUid As String = "<Administrateur>", _
        Optional ByVal sPwd As String = "") As Boolean
        Try
            'Affectation de l'emplacement du fichier comptable
            BaseCpta.Name = sMae
            'Affectation du code utilisateur
            BaseCpta.Loggable.UserName = sUid
            'Affectation du mot de passe
            BaseCpta.Loggable.UserPwd = sPwd
            'Ouverture de la base comptable
            BaseCpta.Open()
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur lors de l'ouverture du fichier mae : " & ex.Message)
            Return False
        End Try
    End Function

    Private Function CloseBase(ByRef BaseCpta As BSCPTAApplication100c) As Boolean
        Try
            'Si la base est ouverte, alors fermeture de la base
            If BaseCpta.IsOpen Then BaseCpta.Close()
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur lors de la fermeture de la base : " & ex.Message)
            Return False
        End Try
    End Function
End Module
```

Sérialisation des identifiants

La sérialisation des identifiants est un mécanisme permettant de générer une clé unique pour chaque objet persistant dans les bases de données Sage 100c. Cette clé se manipule en implémentant l'interface **IStream** (cf. http://msdn.microsoft.com/en-us/library/windows/desktop/aa380034(v=vs.85).aspx), en vue d'être stockée (un fichier ou une table par exemple). A noter que la sérialisation des identifiants fonctionne également sur les objets mémoire (objets créés dans les processus). Cependant, la clé générée sur un objet non persistant, n'est valable que pour la connexion en cours. Ainsi, après fermeture de la connexion à la base, cette clé ne sera plus exploitable puisque l'objet mémoire n'existera plus.

Classe d'implémentation de l'interface IStream

Pour utiliser la sérialisation des identifiants dans un développement spécifique, il est nécessaire de créer une classe implémentant l'interface **IStream**.

Afin d'éviter une réécriture de cette classe pour chacun des développements devant utiliser la sérialisation des identifiants Objets métiers, il est conseillé de créer une assembly publiant cette classe, et de référencer cette assembly pour chacun des développements spécifiques.

Exemple C#

Cette classe publie les méthodes implémentées par l'interface IStream, et en plus publie la méthode ***ToString()***, permettant de récupérer l'identifiant de l'objet sous forme de chaîne de caractères.

```
//Classe d'implémentation de l'interface IStream
public class ImplementIStream : IStream, IDisposable
{
    private Stream _mStream;

    public ImplementIStream(Stream s)
    {
        _mStream = s;
    }

    public void Clone(out IStream ppstm)
    {
        MemoryStream memBuffer = new MemoryStream();
        byte[] memByte = new byte[_mStream.Length];
```

```
            _mStream.Read(memByte, 0, memByte.Length);
            memBuffer.Write(memByte, 0, memByte.Length);
            ppstm = new ImplementIStream(memBuffer);
        }

        public void Commit(uint grfCommitFlags)
        {
            _mStream.Flush();
        }

        public void LockRegion(_ULARGE_INTEGER libOffset, _ULARGE_INTEGER cb, uint dwLockType)
        {
        }

        public void RemoteCopyTo(IStream pstm, _ULARGE_INTEGER cb, out _ULARGE_INTEGER pcbRead, out _ULARGE_INTEGER
pcbWritten)
        {
            pcbRead.QuadPart = pcbWritten.QuadPart = 0;
        }

        public unsafe void RemoteRead(out byte pv, uint cb, out uint pcbRead)
        {
            fixed (byte* bBuffer = &pv)
            {
                byte* pByte = bBuffer;
                long remainsRead = _mStream.Length - _mStream.Position;
                uint remains = (uint)(remainsRead > cb ? cb : remainsRead);
                for (uint i = 0; i < remains; i++)
                {
                    *pByte = (byte)_mStream.ReadByte();
                    pByte++;
                }
                pcbRead = remains;
            }
```

```
    }

    public unsafe void RemoteWrite(ref byte pv, uint cb, out uint pcbWritten)
    {
        byte[] arBuf = new byte[cb];
        fixed (byte* bBuffer = &pv)
        {
            byte* pByte = bBuffer;
            for (int i = 0; i < cb; i++)
            {
                arBuf[i] = *pByte;
                _mStream.WriteByte(*pByte);
                pByte++;
            }
        }
        pcbWritten = cb;
    }

    public void RemoteSeek(_LARGE_INTEGER dlibMove, uint dwOrigin, out _ULARGE_INTEGER plibNewPosition)
    {
        ulong pos = (ulong)_mStream.Seek(dlibMove.QuadPart, (SeekOrigin)dwOrigin);
        plibNewPosition.QuadPart = pos;
    }

    public void Revert()
    {
    }

    public void SetSize(_ULARGE_INTEGER libNewSize)
    {
        _mStream.SetLength((long)libNewSize.QuadPart);
    }
```

```
    public void Stat(out tagSTATSTG pstatstg, uint grfStatFlag)
    {
        pstatstg = new tagSTATSTG();
    }

    public void UnlockRegion(_ULARGE_INTEGER libOffset, _ULARGE_INTEGER cb, uint dwLockType)
    {
    }

    public override string ToString()
    {
        StreamReader streamReader = new StreamReader(_mStream);
        _mStream.Seek(0, new SeekOrigin());
        return streamReader.ReadToEnd();
    }

    public void Dispose()
    {
        if (_mStream == null)
        {
            return;
        }
        _mStream.Dispose();
        _mStream = null;
    }
}

public class StreamIStreamWrapper : IStream, IDisposable
{
    private Stream _mStream;

    public StreamIStreamWrapper(Stream s)
    {
```

```
            _mStream = s;
        }

        public void Clone(out IStream ppstm)
        {
            MemoryStream buffer = new MemoryStream();
            byte[] pMem = new byte[_mStream.Length];
            _mStream.Read(pMem, 0, pMem.Length);
            buffer.Write(pMem, 0, pMem.Length);
            ppstm = new ImplementIStream(buffer);
        }

        public void Commit(uint grfCommitFlags)
        {
            _mStream.Flush();
        }

        public void LockRegion(_ULARGE_INTEGER libOffset, _ULARGE_INTEGER cb, uint dwLockType)
        {
        }

        public void RemoteCopyTo(IStream pstm, _ULARGE_INTEGER cb, out _ULARGE_INTEGER pcbRead, out _ULARGE_INTEGER
pcbWritten)
        {
            pcbRead.QuadPart = pcbWritten.QuadPart = 0;
        }

        public unsafe void RemoteRead(out byte pv, uint cb, out uint pcbRead)
        {
            fixed (byte* pBuf = &pv)
            {
                byte* pd = pBuf;

                long RemainsToReadFromStream = _mStream.Length - _mStream.Position;
```

```
            uint Remains = (uint)(RemainsToReadFromStream > cb ? cb : RemainsToReadFromStream);
            for (uint i = 0; i < Remains; i++)
            {
                *pd = (byte)_mStream.ReadByte();
                pd++;
            }
            pcbRead = Remains;
        }
    }

    public unsafe void RemoteWrite(ref byte pv, uint cb, out uint pcbWritten)
    {
        byte[] buf = new byte[cb];
        fixed (byte* pBuf = &pv)
        {
            byte* pd = pBuf;
            for (int i = 0; i < cb; i++)
            {
                buf[i] = *pd;
                _mStream.WriteByte(*pd);

                pd++;
            }
        }
        pcbWritten = cb;
    }

    public void RemoteSeek(_LARGE_INTEGER dlibMove, uint dwOrigin, out _ULARGE_INTEGER plibNewPosition)
    {
        ulong position = (ulong)_mStream.Seek(dlibMove.QuadPart, (SeekOrigin)dwOrigin);
        plibNewPosition.QuadPart = position;
    }
```

```
public void Revert()
{
}

public void SetSize(_ULARGE_INTEGER libNewSize)
{
    _mStream.SetLength((long)libNewSize.QuadPart);
}

public void Stat(out tagSTATSTG pstatstg, uint grfStatFlag)
{
    pstatstg = new tagSTATSTG();
}

public void UnlockRegion(_ULARGE_INTEGER libOffset, _ULARGE_INTEGER cb, uint dwLockType)
{
}

public override string ToString()
{
    StreamReader reader = new StreamReader(_mStream);
    _mStream.Seek(0, new SeekOrigin());
    return reader.ReadToEnd();
}

public void Dispose()
{
    if (_mStream == null)
    {
        return;
    }
    _mStream.Dispose();
    _mStream = null;
```

```
    }
  }
```

Sérialisation des objets

Lorsqu'une classe implémentant l'interface **IStream** aura été integrée dans la solution .net (soit sous forme de classe, soit sous forme d'assembly), il est alors possible d'appeler la méthode de sérialisation des objets métiers : **WriteTo**.

Cette méthode va permettre de sérialiser l'identifiant de l'objet dans un stream (FileStream, MemoryStream…). Charge au développement spécifique de stocker ce stream (dans un fichier, une table…) pour qu'il puisse être relu et permette d'accéder à l'objet y correspondant.

Ces stream, lorsqu'ils sont convertis sous forme de chaîne de caractères, ont la structure suivante :

[Numéro de version d'interface]|[Interface]|[Identifiant Sage 100c]

Exemple d'identifiant du tiers CARAT : 1114112|com.sage.om.cpta.CLIENT|CARAT

Désérialisation des objets

La désérialisation va permettre d'extraire la clé stockée sous forme de stream, afin de pouvoir recharger l'objet qui correspond à cette clé. La désérialisation s'effectue par l'appel de la méthode **ReadFrom**. Cette clé étant unique pour chaque objet, elle garantit d'accéder à l'objet qui l'a généré.

Pour les objets pour lesquels il existe un factory de base au niveau du stream applicatif (BSCPTAApplication100c et BSCIALApplication100c), il est possible d'accéder à l'objet en utilisant la méthode **ReadFrom** du stream applicatif. Par exemple, pour accéder directement à un contact tiers, il est possible d'utiliser BSCPTAApplication100c.ReadFrom. Pour les sous-objets pour lesquels il n'existe pas de factory de base (tarif article client par exemple), il faudra d'abord récupérer l'objet maitre (IBOArticle3) pour ensuite accéder au sous objet (IBOArticleTarifClient3).

Exemples avancés

Les exemples avancés décrits ci-après mettent en avant la facilité et la rapidité de développement à l'aide de Sage 100cloud Objets Métiers :

- Les tiers ;
- La gestion des coûts des articles ;
- Les gammes de remise par catégorie tarifaire ;
- Les articles à gammes ;

- Les informations libres ;
- Les écritures comptables ;
- Les modèles de saisie ;
- Développement .Net avec Sage 100cloud Objets Métiers ;
- L'encodage d'écritures avec le processus IPMEncoder ;
- La création de document avec le processus IPMDocument ;
- Contrôle qualité avec le processus IPMControleQualite ;
- Application des barèmes avec le processus IPMAppliquerBareme ;
- Gestion du colisage avec le processus IPMColiser ;
- Prélèvement Série/lot avec le processus IPMPreleverLot ;
- Transfert d'un article avec le processus IPMDocTransferer ;
- Transformation de documents de vente et achat avec le processus IPMDocTransformer ;
- Le lettrage d'écritures comptables avec le processus IPMLettrer ;
- Ajout d'une ligne de sous-total avec le processus IPMDocInsererSousTotal,
- Recalcul du prix de revient d'une nomenclature fabrication avec IPMDocRecalculPrixCompose,
- Sortie d'un article géré par lot,
- Gestion des identifiants,
- Conversion d'un prospect en client.

Les tiers

On distingue 4 types de tiers : les clients, les fournisseurs, les salariés et les autres tiers.

Les clients et les fournisseurs sont respectivement représentés par des objets de type *IBOClient3* et *IBOFournisseur3*. De plus, ils partagent des propriétés et méthodes communes disponibles au niveau des objets de type *IBOTiersPart3* et *IBOTiers3*.

Par contre, les salariés et les autres tiers ne sont pas représentés par des objets spécifiques, mais par un objet de type *IBOTiers3*.

Tous les tiers ont en commun de partager les propriétés et méthodes disponibles au niveau des objets de type *IBOTiers3*.

Par exemple, il est possible de créer un nouveau client (type : *IBOClient3*) puis d'y faire référence par l'intermédiaire d'un objet de type *IBOTiers3*.

Exemple :

L'exemple ci-dessous permet de créer de nouveaux tiers de différents types, puis d'y accéder par l'intermédiaire d'un objet de type IBOTiers3.

Code source :

```
Option Strict Off

Imports Objets100cLib


Module Tiers

  Sub Main()

    Dim BaseCpta As New BSCPTAApplication100c

    If OuvreBaseCpta(BaseCpta, "C:\Temp\Bijou.mae") Then

      CreeTiers(BaseCpta, "DUBOIS", TiersType.TiersTypeFournisseur)

      CreeTiers(BaseCpta, "DURANT", TiersType.TiersTypeClient)

      CreeTiers(BaseCpta, "DUPONT", TiersType.TiersTypeSalarie)

      AfficheTiers(BaseCpta, "DUBOIS")

      AfficheTiers(BaseCpta, "DURANT")

      AfficheTiers(BaseCpta, "DUPONT")

      Console.ReadLine()

      FermeBaseCpta(BaseCpta)

    End If

  End Sub


  Sub CreeTiers(ByVal BaseCpta As BSCPTAApplication100c, ByVal NumTiers As String, ByVal TypeTiers As TiersType)

    Try

      If Not BaseCpta.FactoryTiers.ExistNumero(NumTiers) Then

        Dim Tiers As IBOTiers3 = Nothing

        Select Case TypeTiers

          Case TiersType.TiersTypeClient

            Tiers = BaseCpta.FactoryClient.Create

          Case TiersType.TiersTypeFournisseur
```

```
                Tiers = BaseCpta.FactoryFournisseur.Create
            Case TiersType.TiersTypeSalarie
                Tiers = BaseCpta.FactoryTiersSalarie.Create
            Case TiersType.TiersTypeAutre
                Tiers = BaseCpta.FactoryTiersAutre.Create
        End Select
        With Tiers
            .CT_Num = NumTiers
            .SetDefault()
            .WriteDefault()
        End With
    End If
  Catch ex As Exception
    Console.WriteLine("Erreur : " & ex.Message)
  End Try
End Sub

Sub AfficheTiers(ByVal BaseCpta As BSCPTAApplication100c, ByVal NumTiers As String)
  Try
    If BaseCpta.FactoryTiers.ExistNumero(NumTiers) Then
      Dim Tiers As IBOTiers3 = BaseCpta.FactoryTiers.ReadNumero(NumTiers)
      Console.WriteLine(vbNewLine & "Numero : " & Tiers.CT_Num)
      Console.WriteLine("Type : " & Tiers.CT_Type.ToString)
      Console.WriteLine("Compte général principal : " & Tiers.CompteGPrinc.CG_Num)
    End If
  Catch ex As Exception
    Console.WriteLine("Erreur : " & ex.Message)
  End Try
End Sub

Function OuvreBaseCpta(ByRef BaseCpta As BSCPTAApplication100c, _
  ByVal NomBaseCpta As String, Optional ByVal Utilisateur As String = "<Administrateur>", _
  Optional ByVal MotDePasse As String = "") As Boolean
```

```
        Try
            BaseCpta.Name = NomBaseCpta
            BaseCpta.Loggable.UserName = Utilisateur
            BaseCpta.Loggable.UserPwd = MotDePasse
            BaseCpta.Open()
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur en ouverture de base comptable : {0}", ex.Message)
            Return False
        End Try
    End Function

    Function FermeBaseCpta(ByRef BaseCpta As BSCPTAApplication100c) As Boolean
        Try
            BaseCpta.Close()
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur en fermeture de base comptable : {0}", ex.Message)
            Return False
        End Try
    End Function
End Module
```

La gestion des coûts des articles

La liste déroulante du volet *Stock* du fichier *article* permet de gérer jusqu'à 3 coûts différents (coût de stockage, de transport, etc...). Ces coûts permettent de calculer le prix de revient de l'article. Ils se définissent depuis une fiche article, sur le volet ***Paramètres*** dans le paragraphe ***Frais fixes*** de l'élément ***Logistique***.

Pour chaque coût, il est possible de renseigner jusqu'à 3 types de frais différents qui peuvent être cumulés :

- Frais unitaire (U) ;
- Frais forfaitaires (F) ;
- Frais en pourcentage (%).

Exemple :

12U+45F+10%
Ces frais sont gérés par l'intermédiaire d'objets issus des interfaces suivantes :

Interface	Description
IFrais2	Les différents frais d'un article (jusqu'à 3 frais différents). Exemple : L'article BAAR01 possède 3 frais différents : Coût de transport, Coût de transport et Coût de manutention.
IFraisElem2	Frais particulier d'un article. Exemple : Le premier frais de l'article BAAR01 est : Coût de transport.
IRemise2	Mode de calcul d'un frais particulier. Exemple : Le mode de calcul du frais Coût de transport est : 10%+1U
IREmiseElem2	Elément de calcul d'un frais particulier. Exemple : Le second élément de calcul du frais Coût de transport est : 1U

Exemple :

L'exemple ci-dessous permet de modifier et de visualiser les différents coûts affectés à l'article BAAR01 et de calculer le montant unitaire des frais de l'article :

Code source :

```
Option Strict Off

Imports Objets100cLib

Imports System


Module FraisRemise


    Structure ParamBase

        Dim NomBase As String

        Dim Utilisateur As String
```

```
    Dim MotDePasse As String
  End Structure


  Sub Main()
    Dim BaseCpta As New BSCPTAApplication100c
    Dim BaseCial As New BSCIALApplication100c
    Dim Article As IBOArticle3
    Dim ParamBaseCpta As ParamBase = Nothing
    Dim ParamBaseCial As ParamBase = Nothing

    ParamBaseCpta.NomBase = "C:\temp\Bijou.mae"
    ParamBaseCial.NomBase = "C:\temp\Bijou.gcm"
    If OuvreBaseCial(BaseCial, BaseCpta, ParamBaseCial, ParamBaseCpta) Then
      Try
        If BaseCial.FactoryArticle.ExistReference("BAAR01") Then
          Dim PU, Qte As Double

          ' Paramétrage des 3 frais de l'article BAAR01 :
          Article = BaseCial.FactoryArticle.ReadReference("BAAR01")
          CreeFraisArticle(Article, 1, Nothing, "2U")  ' Coût de stockage
          CreeFraisArticle(Article, 2, Nothing, "10%+1U") ' Coût de transport
          CreeFraisArticle(Article, 3, "Coût de manutention", "8%+50F+10U")
          Article.Write()

          ' Calcul et affichage du montant unitaire des frais en fonction du PU et de la Qte :
          PU = 100 : Qte = 10
          AfficheCalculTotalFrais(Article, PU, Qte) ' Coût total
          AfficheCalculDetailFrais(Article, 1, PU, Qte) ' Coût de stockage
          AfficheCalculDetailFrais(Article, 2, PU, Qte) ' Coût de transport
          AfficheCalculDetailFrais(Article, 3, PU, Qte) ' Coût de manutention
        End If
      Catch ex As Exception
        Console.WriteLine(ex.Message)
```

```
        Finally
            FermeBaseCial(BaseCial)
        End Try
    End If
    Console.ReadLine()
End Sub

Sub CreeFraisArticle(ByRef Article As IBOArticle3, ByVal Index As Integer, ByVal Intitule As String, ByVal ModeCalcul As String)
    With Article.AR_Frais.Frais(Index)
        .FR_Denomination = IIf(Intitule Is Nothing, .FR_Denomination, Intitule)
        .Remise.FromString(IIf(ModeCalcul Is Nothing, .Remise.ToString, ModeCalcul))
    End With
End Sub

Sub AfficheCalculDetailFrais(ByRef Article As IBOArticle3, ByVal Index As Integer, ByVal MontantUnitaire As Double, ByVal Qte As Double)
    Console.WriteLine(vbNewLine & Article.AR_Frais.Frais(Index).FR_Denomination & _
        " : " & Article.AR_Frais.Frais(Index).Remise.ToString)
    Console.WriteLine("Montant des frais unitaires pour {0} articles {1} à {2} euro/unité : {3}", _
        Qte, Article.AR_Ref, MontantUnitaire, Article.AR_Frais.Frais(Index).Calcul(MontantUnitaire, Qte, 2) - MontantUnitaire)
End Sub

Sub AfficheCalculTotalFrais(ByRef Article As IBOArticle3, ByVal MontantUnitaire As Double, ByVal Qte As Double)
    Console.WriteLine(vbNewLine & Article.AR_Ref & " : ")
    Console.WriteLine("Montant total des frais unitaires pour {0} articles {1} à {2} euro/unité : {3}", _
        Qte, Article.AR_Ref, MontantUnitaire, Article.AR_Frais.Calcul(MontantUnitaire, Qte, 2) - MontantUnitaire)
End Sub

Function OuvreBaseCial(ByRef BaseCial As BSCIALApplication100c, ByRef BaseCpta As BSCPTAApplication100c, _
    ByVal ParamBaseCial As ParamBase, ByVal ParamBaseCpta As ParamBase) As Boolean
    Dim LoginCpta, LoginCial As IBILoggable
    Try
        With ParamBaseCpta
            BaseCpta.Name = .NomBase
```

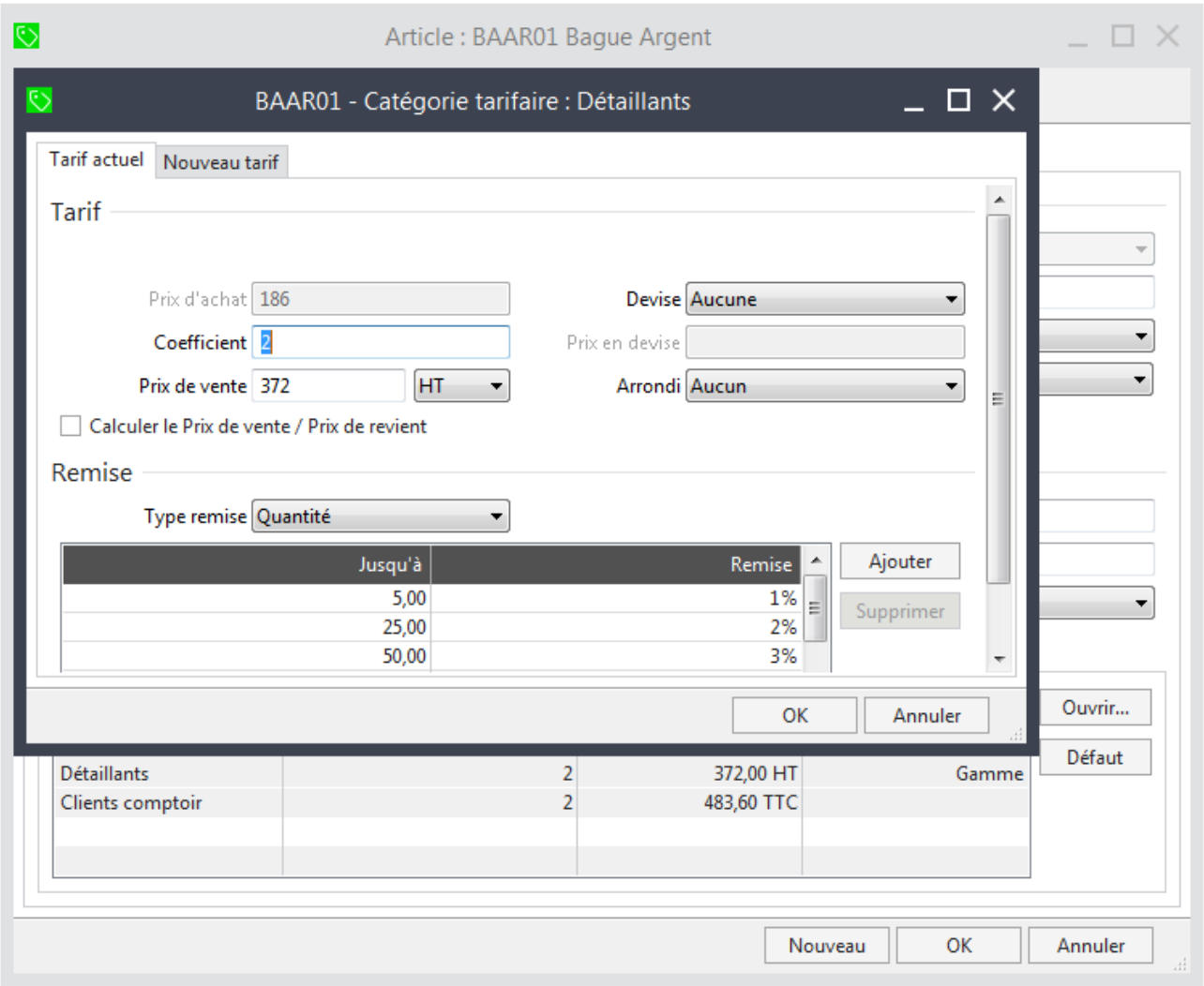
```
            LoginCpta = BaseCpta
            If Not .Utilisateur Is Nothing Then
                LoginCpta.UserName = .Utilisateur
                LoginCpta.UserPwd = .MotDePasse
            End If
        End With
        BaseCial.CptaApplication = BaseCpta
        With ParamBaseCial
            BaseCial.Name = .NomBase
            LoginCial = BaseCial
            If Not .Utilisateur Is Nothing Then
                LoginCial.UserName = .Utilisateur
                LoginCial.UserPwd = .MotDePasse
            End If
        End With
        BaseCial.Open()
        Return True
    Catch ex As Exception
        Console.WriteLine("Erreur en ouverture de base commerciale : {0}", ex.Message)
        Return False
    End Try
End Function

Function FermeBaseCial(ByRef BaseCial As BSCIALApplication100c) As Boolean
    Try
        BaseCial.Close()
        Return True
    Catch ex As Exception
        Console.WriteLine("Erreur en fermeture de base commerciale : {0}", ex.Message)
        Return False
    End Try
End Function
```

End Module

Les gammes de remise par catégorie tarifaire

Chaque article peut posséder jusqu'à 32 catégories tarifaires de vente différentes.

Pour chaque catégorie tarifaire, il est possible de paramétrer un tarif qui peut être unique ou se présenter sous forme d'une gamme de remise, c'est-à-dire varier en fonction du montant ou des quantités vendues, sous forme d'une remise ou d'un prix net :



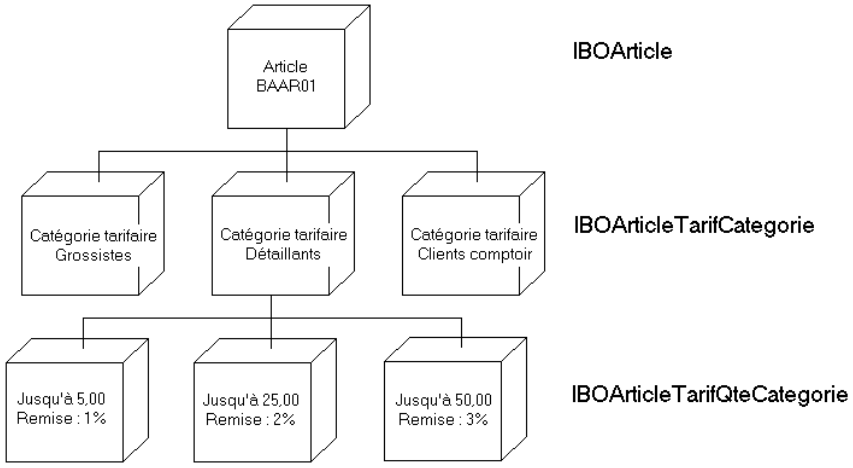
La gestion des gammes de remise par catégorie tarifaire s'effectue par l'intermédiaire des objets suivants :

Objet	Objet maître	Factory	Description
IBOArticleTarifCategorie3	IBOArticle3	FactoryArticleTarifCategorie	Tarif par catégorie tarifaire article. Exemple : la catégorie tarifaire Détaillants.
IBOArticleTarifQteCategorie3	IBOArticleTarifCategorie3	FactoryArticleTarifQte	Tarif par quantité, montant ou prix net pour une catégorie tarifaire. Exemple : Jusqu'à 25 produits vendus, remise de 2 %

Le principe est identique pour :

- Les tarifs client
- Les tarifs fournisseur

Exemple :



Le code source ci-dessous permet de modifier le montant de la gamme de remise "Quantité" préalablement affectée à la catégorie tarifaire "Détaillants" de l'article "BAAR01" :

Code source :

```
Option Strict Off

Imports Objets100cLib

Imports System
```

```
Module GammeRemiseCatTarifaire


    Structure ParamBase
        Dim NomBase As String
        Dim Utilisateur As String
        Dim MotDePasse As String
    End Structure


    Sub Main()
        Dim BaseCial As New BSCIALApplication100c
        Dim BaseCpta As New BSCPTAApplication100c
        Dim ParamBaseCial As ParamBase = Nothing
        Dim ParamBaseCpta As ParamBase = Nothing
        Dim Article As IBOArticle3 = Nothing
        'Dim Famille As IBOFamille3
        Dim RefArticle, RefCatTarifaire, RefGammeRemise As String
        Dim Remise As New ArrayList ' Liste d'objets


        ParamBaseCial.NomBase = "C:\temp\Bijou.gcm"
        ParamBaseCpta.NomBase = "C:\temp\Bijou.mae"
        RefArticle = "BAAR01"
        RefCatTarifaire = "Détaillants"
        RefGammeRemise = "Quantité"
        With Remise 'Stockage dans la liste des différentes remises
            .Add("1%")
            .Add("2%")
            .Add("3%")
            .Add("4%")
            .Add("5%")
            .Add("6%")
            .Add("7%")
            .Add("8%")
        End With
```

```
    If OuvreBaseCial(BaseCial, BaseCpta, ParamBaseCial, ParamBaseCpta) Then
        If LisArticle(BaseCial, Article, RefArticle) Then
            Console.WriteLine("Avant modification :")
            AfficheCatTarifArticle(Article)
            Console.WriteLine(vbNewLine & "Après modification des gammes de remise :" & vbNewLine)
            ModifieGammeRemiseCatTarifArticle(Article, RefCatTarifaire, RefGammeRemise, Remise)
            AfficheCatTarifArticle(Article)
            Console.ReadLine()
        End If
        FermeBaseCial(BaseCial)
    End If
End Sub

Function LisArticle(ByRef BaseCial As BSCIALApplication100c, ByRef Article As IBOArticle3, ByVal RefArticle As String) As Boolean
    Try
        ' S'il existe, retourne l'objet article correspondant
        ' à la référence article :
        Article = BaseCial.FactoryArticle.ReadReference(RefArticle)
        Return True
    Catch ex As Exception
        Console.WriteLine("Erreur en lecture d'article : {0}", ex.Message)
        Return False
    End Try
End Function

Sub AfficheCatTarifArticle(ByRef Article As IBOArticle3)
    Dim TarifCategorie As IBOArticleTarifCategorie3
    Dim TarifQteCategorie As IBOArticleTarifQteCategorie3
    Try
        Console.WriteLine("Article : {0}", Article.AR_Ref)
        ' Affiche les informations de chaque catégorie tarifaire de l'article
        For Each TarifCategorie In Article.FactoryArticleTarifCategorie.List
```

```
            Console.WriteLine(vbNewLine & "Catégorie tarifaire : {0}", TarifCategorie.CategorieTarif.CT_Intitule.ToString)
            If TarifCategorie.FactoryArticleTarifQte.List.Count > 0 Then
                Console.WriteLine("Gamme de remise : {0}", TarifCategorie.GammeRemise.G_Intitule)
                ' Affiche le détail des gammes de remise de l'article s'il en existe
                For Each TarifQteCategorie In TarifCategorie.FactoryArticleTarifQte.List
                    Console.WriteLine("Jusqu'à {0} : {1}", TarifQteCategorie.BorneSup, TarifQteCategorie.Remise.ToString)
                Next
            Else
                Console.WriteLine("Aucune gamme de remise")
            End If
        Next
    Catch ex As Exception
        Console.WriteLine("Erreur en affichage de catégorie tarifaire article : {0}", ex.Message)
    End Try
End Sub

Function ModifieGammeRemiseCatTarifArticle(ByRef Article As IBOArticle3, ByVal RefCatTarifaire As String, ByVal
RefGammeRemise As String, ByVal Remise As ArrayList) As Boolean
    Dim CatTarifArticle As IBOArticleTarifCategorie3
    Dim CatTarifaire As IBPCategorieTarif
    Dim TarifQteCategorie As IBOArticleTarifQteCategorie3
    Dim NbBorneSup, i As Integer
    Try
        CatTarifaire = CType(Article.Stream, BSCIALApplication100c).FactoryCategorieTarif.ReadIntitule(RefCatTarifaire)
        ' Si la catégorie tarifaire (CatTarifaire) n'est pas déjà associée à l'article, ajout de la catégorie tarifaire à l'article
        If Not LisCatTarifArticle(Article, CatTarifArticle, CatTarifaire) Then
            AjouteCatTarifArticle(Article, CatTarifArticle, CatTarifaire)
        End If
        ' Créé une gamme de remise article :
        CatTarifArticle.GammeRemise = CType(CatTarifArticle.Stream,
BSCIALApplication100c).FactoryGamme.ReadIntitule(RefGammeRemise)
        CatTarifArticle.WriteDefault()
        ' Affecte à chaque borne de la gamme de remise, une valeur de remise :
        NbBorneSup = CatTarifArticle.FactoryArticleTarifQte.List.Count()
```

```
            Do While i < NbBorneSup AndAlso i < Remise.Count
                TarifQteCategorie = CatTarifArticle.FactoryArticleTarifQte.List.Item(i + 1)
                TarifQteCategorie.Remise.FromString(Remise(i))
                TarifQteCategorie.Write()
                i += 1
            Loop
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur en modification d'une gamme de remise de catégorie tarifaire article : {0}", ex.Message)
            Return False
        End Try
    End Function

    Function LisCatTarifArticle(ByRef Article As IBOArticle3, ByRef CatTarifArticle As IBOArticleTarifCategorie3, ByRef CatTarifaire As
IBPCategorieTarif) As Boolean
        Try
            ' S'il existe, retourne l'objet catégorie tarifaire (CatTarifArticle)
            ' correspondant à l'intitulé CatTarifaire :
            For Each CatTarifArticle In Article.FactoryArticleTarifCategorie.List
                If CatTarifArticle.CategorieTarif Is CatTarifaire Then
                    Return True
                End If
            Next
            Return False
        Catch ex As Exception
            Console.WriteLine("Erreur en lecture de catégorie tarifaire article : {0}", ex.Message)
            Return False
        End Try
    End Function

    Function AjouteCatTarifArticle(ByRef Article As IBOArticle3, ByRef CatTarifArticle As IBOArticleTarifCategorie3, ByRef CatTarifaire
As IBPCategorieTarif) As Boolean
        Try
            ' Ajoute une catégorie tarifaire (CatTarifaire) à l'article :
```

```
            CatTarifArticle = Article.FactoryArticleTarifCategorie.Create
            CatTarifArticle.CategorieTarif = CatTarifaire
            CatTarifArticle.Write()
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur en ajout d'une catégorie tarifaire article : {0}", ex.Message)
            Return False
        End Try
    End Function

    Function OuvreBaseCial(ByRef BaseCial As BSCIALApplication100c, ByRef BaseCpta As BSCPTAApplication100c, _
        ByVal ParamBaseCial As ParamBase, ByVal ParamBaseCpta As ParamBase) As Boolean
        Dim LoginCpta, LoginCial As IBILoggable
        Try
            With ParamBaseCpta
                BaseCpta.Name = .NomBase
                LoginCpta = BaseCpta
                If Not .Utilisateur Is Nothing Then
                    LoginCpta.UserName = .Utilisateur
                    LoginCpta.UserPwd = .MotDePasse
                End If
            End With
            BaseCial.CptaApplication = BaseCpta
            With ParamBaseCial
                BaseCial.Name = .NomBase
                LoginCial = BaseCial
                If Not .Utilisateur Is Nothing Then
                    LoginCial.UserName = .Utilisateur
                    LoginCial.UserPwd = .MotDePasse
                End If
            End With
            BaseCial.Open()
            Return True
```

```
    Catch ex As Exception
      Console.WriteLine("Erreur en ouverture de base commerciale : {0}", ex.Message)
      Return False
    End Try
  End Function

  Function FermeBaseCial(ByRef BaseCial As BSCIALApplication100c) As Boolean
    Try
      BaseCial.Close()
      Return True
    Catch ex As Exception
      Console.WriteLine("Erreur en fermeture de base commerciale : {0}", ex.Message)
      Return False
    End Try
  End Function

End Module
```

Les articles à gammes

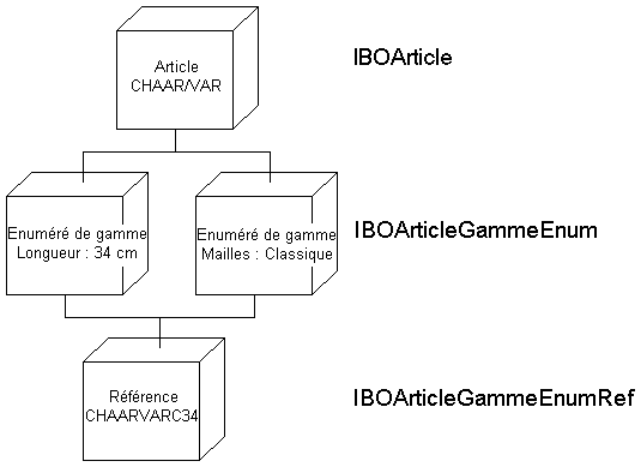
Un article peut être géré sous forme de gamme simple ou double.

Pour chaque énuméré de gamme, il est possible de paramétrer une référence, un code barre et un prix d'achat spécifique.

La gestion des gammes s'effectue par l'intermédiaire des objets suivants :

Objet	Objet maître	Factory	Description
IBOArticleGammeEnum3	IBOArticle3	FactoryArticleGammeEnum1 FactoryArticleGammeEnum2	Enuméré de gamme 1 ou 2.
IBOArticleGammeEnumRef3	IBOArticleGammeEnum3	FactoryArticleGammeEnumRef	Référence de l'énuméré de gamme 1 ou 2.

Exemple :



Le code source suivant permet :

- l'affichage du détail des gammes d'un article donné ;
- la modification d'une gamme d'un article donné ;
- la création de nouveaux énumérés de gamme ;
- le paramétrage de nouvelles gammes.

Code source :

```
Option Strict Off

Imports System

Imports Objets100cLib


Module GammesArticle


    Structure ParamBase

        Dim NomBase As String

        Dim Utilisateur As String

        Dim MotDePasse As String

    End Structure


    Structure Gamme

        Dim Reference As String

        Dim CodeBarre As String
```

```
    Dim PrixAchat As Double
    Dim RefGammeEnum1 As String
    Dim RefGammeEnum2 As String
End Structure

Sub Main()
    Dim BaseCial As New BSCIALApplication100c
    Dim BaseCpta As New BSCPTAApplication100c
    Dim ParamCial As ParamBase = Nothing
    Dim ParamCpta As ParamBase = Nothing
    Dim Article As IBOArticle3
    Dim ParamGamme As Gamme
    Try
        ParamCial.NomBase = "C:\Temp\Bijou.gcm"
        ParamCpta.NomBase = "C:\Temp\Bijou.mae"
        If OuvreBaseCial(BaseCial, BaseCpta, ParamCial, ParamCpta) Then
            If BaseCial.FactoryArticle.ExistReference("CHAAR/VAR") Then
                Article = BaseCial.FactoryArticle.ReadReference("CHAAR/VAR") ' Article double gamme
                AfficheGammeArticle(Article)
                ' Modifie le prix d'achat de la gamme CHAARVARC54 :
                ParamGamme = Nothing
                ParamGamme.PrixAchat = 130
                ModifieGammeArticle(Article, "CHAARVARC54", ParamGamme)
                ' Modifie la référence et le code barre de la gamme CHAARVARF42
                ParamGamme = Nothing
                ParamGamme.Reference = "CHAARVARF42B"
                ParamGamme.CodeBarre = 38141027
                ModifieGammeArticle(Article, "CHAARVARF42", ParamGamme)
                ' Ajoute un nouvel énuméré de gamme (68 cm) et une nouvelle référence CHAARVARC68 :
                ParamGamme = Nothing
                With ParamGamme
                    .Reference = "CHAARVARC68"
                    .CodeBarre = "38141030"
```

```
                .PrixAchat = 180
                .RefGammeEnum1 = "68 cm"
                .RefGammeEnum2 = "Classique"
            End With
            AjouteGammeArticle(Article, ParamGamme)
            ' Ajoute deux nouveaux énumérés de gamme (35 cm et Fine) et une nouvelle référence CHAARVARF35 :
            ParamGamme = Nothing
            With ParamGamme
                .Reference = "CHAARVARF35"
                .CodeBarre = "38141031"
                .PrixAchat = 120
                .RefGammeEnum1 = "35 cm"
                .RefGammeEnum2 = "Fine"
            End With
            AjouteGammeArticle(Article, ParamGamme)
            AfficheGammeArticle(Article)
        End If
        If BaseCial.FactoryArticle.ExistReference("BAOR01") Then
            Article = BaseCial.FactoryArticle.ReadReference("BAOR01") ' Article mono-gamme
            AfficheGammeArticle(Article)
            ' Ajoute un nouvel énumérés de gamme (Diamant) et une nouvelle référence BAOR01DI :
            ParamGamme = Nothing
            With ParamGamme
                .Reference = "BAOR01DI"
                .CodeBarre = "38141032"
                .PrixAchat = 166
                .RefGammeEnum1 = "Diamant"
            End With
            AjouteGammeArticle(Article, ParamGamme)
            AfficheGammeArticle(Article)
            Console.ReadLine()
        End If
    End If
```

```
    Catch ex As Exception
        Console.WriteLine("Erreur : " & ex.Message)
    Finally
        FermeBaseCial(BaseCial)
    End Try
End Sub

Sub AfficheGammeArticle(ByVal Article As IBOArticle3)
    Try
        Console.WriteLine("Article : " & Article.AR_Ref)
        If Not Article.Gamme1 Is Nothing Then
            Console.WriteLine(vbNewLine & "Enumérés gamme 1 (" & Article.Gamme1.G_Intitule & ") :")
            For Each GammeEnum1 As IBOArticleGammeEnum3 In Article.FactoryArticleGammeEnum1.List()
                Console.WriteLine(GammeEnum1.EG_Enumere)
                For Each GammeEnumRef As IBOArticleGammeEnumRef3 In GammeEnum1.FactoryArticleGammeEnumRef.List
                    With GammeEnumRef
                        If .ArticleGammeEnum2 Is Nothing Then
                            Console.WriteLine(.AE_Ref & " / " & .AE_CodeBarre & " / " & .AE_PrixAch)
                        Else
                            Console.WriteLine(.AE_Ref & " / " & .AE_CodeBarre & " / " & _
                                .ArticleGammeEnum2.EG_Enumere & " / " & .AE_PrixAch)
                        End If
                    End With
                Next
            Next
            If Not Article.Gamme2 Is Nothing Then
                Console.WriteLine(vbNewLine & "Enumérés gamme 2 (" & Article.Gamme2.G_Intitule & ") :")
                For Each GammeEnum2 As IBOArticleGammeEnum3 In Article.FactoryArticleGammeEnum2.List()
                    Console.WriteLine(GammeEnum2.EG_Enumere)
                    For Each GammeEnumRef As IBOArticleGammeEnumRef3 In GammeEnum2.FactoryArticleGammeEnumRef.List
                        With GammeEnumRef
                            Console.WriteLine(.AE_Ref & " / " & .AE_CodeBarre & " / " & _
                                .ArticleGammeEnum1.EG_Enumere & " / " & .AE_PrixAch)
```

```
                End With
            Next
        Next
    End If
  End If
 Catch ex As Exception
  Console.WriteLine("Erreur : " & ex.Message)
 End Try
End Sub

Sub ModifieGammeArticle(ByRef Article As IBOArticle3, ByVal RefGamme As String, ByVal ParamGamme As Gamme)
 Try
  If Article.Gamme1 Is Nothing Then
   Throw New Exception(Article.AR_Ref & " n'est pas un article à gamme !")
  Else
   For Each GammeEnum1 As IBOArticleGammeEnum3 In Article.FactoryArticleGammeEnum1.List()
    For Each GammeEnumRef As IBOArticleGammeEnumRef3 In GammeEnum1.FactoryArticleGammeEnumRef.List
     If GammeEnumRef.AE_Ref = RefGamme Then
      With GammeEnumRef
       .AE_CodeBarre = IIf(ParamGamme.CodeBarre = Nothing, .AE_CodeBarre, ParamGamme.CodeBarre)
       .AE_PrixAch = IIf(ParamGamme.PrixAchat = Nothing, .AE_PrixAch, ParamGamme.PrixAchat)
       .AE_Ref = IIf(ParamGamme.Reference = Nothing, .AE_Ref, ParamGamme.Reference)
       .Write()
       Exit Sub
      End With
     End If
    Next
   Next
  End If
 Catch ex As Exception
  Console.WriteLine("Erreur : " & ex.Message)
 End Try
End Sub
```

```
Sub AjouteGammeArticle(ByRef Article As IBOArticle3, ByVal ParamGamme As Gamme)
    Dim GammeEnum1 As IBOArticleGammeEnum3 = Nothing
    Dim GammeEnum2 As IBOArticleGammeEnum3 = Nothing
    Dim GammeEnumRef As IBOArticleGammeEnumRef3
    Try
        If Article.Gamme1 Is Nothing Then ' Test si l'article est à gamme
            Throw New Exception(Article.AR_Ref & " n'est pas un article à gamme !")
        Else
            If ParamGamme.RefGammeEnum1 Is Nothing Then ' Test si la nouvelle gamme possède un énuméré de gamme
                Throw New Exception(Article.AR_Ref & " est un article à gamme !")
            Else
                GammeEnum1 = LisGammeEnum(Article.FactoryArticleGammeEnum1.List, ParamGamme.RefGammeEnum1)
                If GammeEnum1 Is Nothing Then ' Si l'énuméré de la gamme n'existe pas, il est créé
                    GammeEnum1 = CreeGammeEnum(Article.FactoryArticleGammeEnum1.Create, ParamGamme.RefGammeEnum1)
                End If
            End If
            If Article.Gamme2 Is Nothing Then ' Test si l'article est à double gamme
                If Not ParamGamme.RefGammeEnum2 Is Nothing Then ' Test si la nouvelle gamme ne possède qu'un énuméré de
gamme 1
                    Throw New Exception(Article.AR_Ref & " n'est pas un article à double gamme !")
                End If
            Else
                If ParamGamme.RefGammeEnum2 Is Nothing Then ' Test si la nouvelle gamme possède bien un énuméré de gamme 2
                    Throw New Exception(Article.AR_Ref & " est un article à double gamme !")
                Else
                    GammeEnum2 = LisGammeEnum(Article.FactoryArticleGammeEnum2.List, ParamGamme.RefGammeEnum2)
                    If GammeEnum2 Is Nothing Then ' Si l'énuméré de la gamme n'existe pas, il est créé
                        GammeEnum2 = CreeGammeEnum(Article.FactoryArticleGammeEnum2.Create, ParamGamme.RefGammeEnum2)
                    End If
                End If
            End If
        End If
        ' Remarque : la nouvelle gamme est automatiquement créée .
```

```
        ' Elle correspond à l'intersection des deux énumérés de gamme.
        GammeEnumRef = LisGammeEnumRef(Article, GammeEnum1, GammeEnum2)
        If Not GammeEnumRef Is Nothing Then ' Modification de la nouvelle gamme
            With GammeEnumRef
                .AE_Ref = IIf(ParamGamme.Reference = Nothing, Nothing, ParamGamme.Reference)
                .AE_CodeBarre = IIf(ParamGamme.CodeBarre = Nothing, Nothing, ParamGamme.CodeBarre)
                .AE_PrixAch = IIf(ParamGamme.PrixAchat = Nothing, Nothing, ParamGamme.PrixAchat)
                .Write()
                Exit Sub
            End With
        End If
    Catch ex As Exception
        Console.WriteLine("Erreur : " & ex.Message)
    End Try
End Sub

Function LisGammeEnum(ByVal CollGammeEnum As IBICollection, ByVal RefGammeEnum As String) As IBOArticleGammeEnum3
    ' Retourne l'énuméré de gamme correspondant à la référence d'énuméré de gamme
    Try
        For Each GammeEnum As IBOArticleGammeEnum3 In CollGammeEnum
            If GammeEnum.EG_Enumere = RefGammeEnum Then
                Return GammeEnum
            End If
        Next
        Return Nothing
    Catch ex As Exception
        Console.WriteLine("Erreur : " & ex.Message)
        Return Nothing
    End Try
End Function

Function LisGammeEnumRef(ByVal Article As IBOArticle3, ByVal GammeEnum1 As IBOArticleGammeEnum3, _
    ByVal GammeEnum2 As IBOArticleGammeEnum3) As IBOArticleGammeEnumRef3
```

```
        ' Retourne la gamme correspondant aux énumérés de gamme
        Try
            For Each GammeEnum As IBOArticleGammeEnum3 In Article.FactoryArticleGammeEnum1.List
                For Each GammeEnumRef As IBOArticleGammeEnumRef3 In GammeEnum.FactoryArticleGammeEnumRef.List
                    If GammeEnumRef.ArticleGammeEnum1 Is GammeEnum1 And GammeEnumRef.ArticleGammeEnum2 Is GammeEnum2
Then
                        Return GammeEnumRef
                    End If
                Next
            Next
            Return Nothing
        Catch ex As Exception
            Console.WriteLine("Erreur : " & ex.Message)
            Return Nothing
        End Try
    End Function

    Function CreeGammeEnum(ByRef GammeEnum As IBOArticleGammeEnum3, ByVal RefGammeEnum As String) As
IBOArticleGammeEnum3
        ' Créé un nouvel énuméré de gamme
        Try
            GammeEnum.EG_Enumere = RefGammeEnum
            GammeEnum.Write()
            Return GammeEnum
        Catch ex As Exception
            Console.WriteLine("Erreur : " & ex.Message)
            Return Nothing
        End Try
    End Function

    Function OuvreBaseCial(ByRef BaseCial As BSCIALApplication100c, ByRef BaseCpta As BSCPTAApplication100c, _
        ByVal ParamBaseCial As ParamBase, ByVal ParamBaseCpta As ParamBase) As Boolean
        Dim LoginCpta, LoginCial As IBILoggable
        Try
```

```
        With ParamBaseCpta
            BaseCpta.Name = .NomBase
            LoginCpta = BaseCpta
            If Not .Utilisateur Is Nothing Then
                LoginCpta.UserName = .Utilisateur
                LoginCpta.UserPwd = .MotDePasse
            End If
        End With
        BaseCial.CptaApplication = BaseCpta
        With ParamBaseCial
            BaseCial.Name = .NomBase
            LoginCial = BaseCial
            If Not .Utilisateur Is Nothing Then
                LoginCial.UserName = .Utilisateur
                LoginCial.UserPwd = .MotDePasse
            End If
        End With
        BaseCial.Open()
        Return True
    Catch ex As Exception
        Console.WriteLine("Erreur en ouverture de base commerciale : {0}", ex.Message)
        Return False
    End Try
End Function

Function FermeBaseCial(ByRef BaseCial As BSCIALApplication100c) As Boolean
    Try
        BaseCial.Close()
        Return True
    Catch ex As Exception
        Console.WriteLine("Erreur en fermeture de base commerciale : {0}", ex.Message)
        Return False
    End Try
```

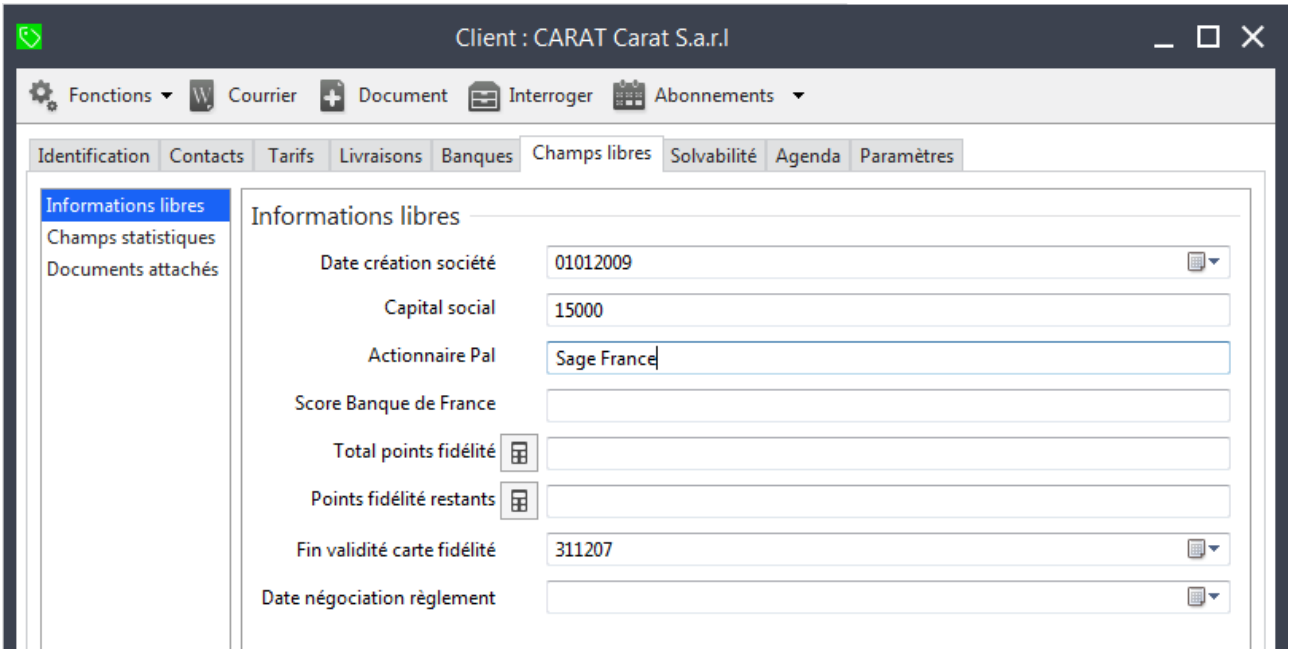
```
    End Function

End Module
```

Les informations libres

Les informations libres sont des champs personnalisables pouvant contenir des valeurs de différents types (texte, date, montant, etc…).

Sage 100cloud Objets Métiers permet la lecture et la modification des valeurs de ces informations libres.

L'affectation de valeurs aux propriétés Informations libres ne peut être réalisée que si l'objet est persistant. De plus, l'accès au paramétrage des informations libres n'est possible qu'en lecture seule.



Exemple :

Le paramétrage des informations libres tiers peut être consulté par l'intermédiaire des objets suivants :

Objet	Objet maître	Propriété	Description
IBIFields	IBOTiers3	InfoLibreFields	Collection des paramétrages des informations libres tiers.
IBIField	IBIFields	Item(ByVal *lIndex* As Integer)	Paramétrage information libre tiers.

La gestion des valeurs des informations libres tiers s'effectue par l'intermédiaire de l'objet suivant :

Objet	Objet maître	Propriété	Description
IBIValues	IBOTiers3	InfoLibre	Collection de valeurs d'informations libres.

Code source :

```
Option Strict Off

Imports System

Imports Objets100cLib


Module InfoLibres


  Sub Main()

    Dim BaseCpta As New BSCPTAApplication100c

    Dim Client As IBOClient3

    Dim RefClient As String = "CARAT"

    Try

      If OuvreBaseCpta(BaseCpta, "C:\Temp\Bijou.mae") Then

        If BaseCpta.FactoryClient.ExistNumero(RefClient) Then

          Client = BaseCpta.FactoryClient.ReadNumero(RefClient)

          AfficheParamInfoLibreTiers(BaseCpta)

          Console.ReadLine()

          Client.InfoLibre.Item(1) = CType("01/01/2009", Date)

          Client.InfoLibre.Item("Actionnaire_Pal") = "Sage France"

          Client.InfoLibre.Item("Capital_social") = 15000

          Client.Write()

        End If

      End If

    Catch ex As Exception

      Console.WriteLine("Erreur : " & ex.Message)

    Finally

      FermeBaseCpta(BaseCpta)

    End Try

  End Sub
```

```
Sub AfficheParamInfoLibreTiers(ByVal BaseCpta As BSCPTAApplication100c)
    Try
        For Each InfoLibre As IBIField In BaseCpta.FactoryTiers.InfoLibreFields
            Console.WriteLine(vbNewLine & "Intitulé : " & InfoLibre.Name)
            Console.WriteLine("Type : " & InfoLibre.Type.ToString)
            If InfoLibre.Type = FieldType.FieldTypeCStr Then
                Console.WriteLine("Longueur : " & InfoLibre.Size)
            End If
            Console.WriteLine("Valeur calculée : " & IIf(InfoLibre.IsCalculable, "Oui", "Non"))
            If InfoLibre.IsCalculable Then
                Console.WriteLine("Formule de calcul :")
                Console.WriteLine(InfoLibre.Formule)
            End If
        Next
    Catch ex As Exception
        Console.WriteLine("Erreur : " & ex.Message)
    End Try
End Sub

Function OuvreBaseCpta(ByRef BaseCpta As BSCPTAApplication100c, _
 ByVal NomBaseCpta As String, Optional ByVal Utilisateur As String = "<Administrateur>", _
 Optional ByVal MotDePasse As String = "") As Boolean
    Try
        BaseCpta.Name = NomBaseCpta
        BaseCpta.Loggable.UserName = Utilisateur
        BaseCpta.Loggable.UserPwd = MotDePasse
        BaseCpta.Open()
        Return True
    Catch ex As Exception
        Console.WriteLine("Erreur en ouverture de base comptable : {0}", ex.Message)
        Return False
    End Try
End Function
```

```
    Function FermeBaseCpta(ByRef BaseCpta As BSCPTAApplication100c) As Boolean
        Try
            BaseCpta.Close()
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur en fermeture de base comptable : {0}", ex.Message)
            Return False
        End Try
    End Function

End Module
```

Les écritures comptables

Cet exemple illustre l'insertion d'écritures comptables correspondant à la vente de produits soumis à TVA.

Le code source comprend plusieurs fonctions :

La principale, *InsereEcritureVenteProduit*, prend en paramètre (sous forme d'une structure) les différents éléments nécessaires à l'insertion des écritures comptables :

- Le code journal ;
- La date de l'écriture ;
- Le numéro de pièce ;
- Le libellé de l'écriture ;
- Le code tiers ;
- Le compte général HT ;
- Le compte général de TVA ;
- Le montant HT.

Cette fonction appelle d'autres fonctions générant les différentes lignes d'écritures comptables :

- Si le tiers dispose d'un mode de règlement sur une ou plusieurs échéances, autant de lignes d'écritures que de lignes d'échéances sont générées (appel de la fonction *InsereLigneTTCMultiEch*).

- Par contre, si aucun mode de règlement n'est affecté au tiers, une seule échéance est générée (appel de la fonction *InsereLigneTTCMonoEch*).

Les lignes de HT et de TVA sont ensuite générées (appel des fonctions *InsereLigneHT* et *InsereLigneTVA*).

Si une erreur se produit avant que la totalité des lignes d'écritures ait pu être écrite dans la base de données, le journal est déséquilibré. Il est donc nécessaire de supprimer ces lignes d'écritures. C'est pourquoi, à chaque insertion réussie d'une ligne d'écriture, l'identifiant de la ligne (IDLigne de type IBIObjectID) est stocké sur une pile (PileLignes de type Stack). En cas d'erreur, la pile est dépilée afin de récupérer chaque identifiant de ligne et de supprimer la ligne correspondante.

Sage 100cloud Objets Métiers simplifie la gestion des modes de règlement tiers par l'appel de la méthode Echeance des objets de type IBOTiersReglement3. Cette méthode prend en paramètre une date et retourne la date d'échéance calculée.

Exemple :

Code source :

```
Option Strict Off

Imports Objets100cLib

Imports System


Module EcrituresVenteProduit


    Structure StrEcriture

        Dim Journal, Tiers, CompteGTTC, CompteGHT, CompteGTVA, [Date], Piece, Libelle As String

        Dim MontantHT, MontantTTC As Double

    End Structure


    Sub Main()

        Dim BaseCpta As New BSCPTAApplication100c
```

```
    Dim Ecriture As StrEcriture = Nothing
    If OuvreBaseCpta(BaseCpta, "C:\temp\Bijou.mae") Then
      With Ecriture
        .Journal = "VTE"
        .Date = "22/08/09"
        .Piece = "AXJ48"
        .Libelle = "Facture 0508102"
        .Tiers = "CARAT"
        .CompteGHT = "701019"
        .CompteGTVA = "4457119"
        .MontantHT = 10000
      End With
      InsereEcritureVenteProduit(BaseCpta, Ecriture)
      With Ecriture
        .Journal = "VTE"
        .Date = "25/08/09"
        .Piece = "AXJ59"
        .Libelle = "Facture 0508103"
        .Tiers = "TOPAZ"
        .CompteGHT = "701005"
        .CompteGTVA = "4457105"
        .MontantHT = 2000
      End With
      InsereEcritureVenteProduit(BaseCpta, Ecriture)
      FermeBaseCpta(BaseCpta)
    End If
  End Sub

  Function InsereEcritureVenteProduit(ByRef BaseCpta As BSCPTAApplication100c, ByVal Ecriture As StrEcriture) As Boolean
    Dim ListeEcheances As IBICollection
    Dim PileLignes As New Stack
    Dim IDLigne As IBIObjectID
    Ecriture.MontantTTC = Ecriture.MontantHT * (1 + CType(BaseCpta.FactoryCompteG.ReadNumero(Ecriture.CompteGHT) _
```

```
        , IBOCompteG3).Taxe.TA_Taux / 100)
    Try
        ' Création de la ligne TTC
        ListeEcheances = CType(BaseCpta.FactoryTiers.ReadNumero(Ecriture.Tiers), IBOTiers3).FactoryTiersReglement.List
        If ListeEcheances.Count > 0 Then ' Règlement sur plusieurs échéances
            Dim MontRestEcheance As Double
            Dim i As Integer
            MontRestEcheance = Ecriture.MontantTTC
            For i = 1 To ListeEcheances.Count
                IDLigne = InsereLigneTTCMultiEch(BaseCpta, Ecriture, ListeEcheances.Item(i), MontRestEcheance)
                If Not IDLigne Is Nothing Then
                    PileLignes.Push(IDLigne)
                Else
                    Throw New Exception
                End If
            Next
            ListeEcheances = Nothing
        Else ' Règlement en une seule échéance
            IDLigne = InsereLigneTTCMonoEch(BaseCpta, Ecriture)
            If Not IDLigne Is Nothing Then
                PileLignes.Push(IDLigne)
            Else
                Throw New Exception
            End If
        End If
        ' Création de la ligne HT
        IDLigne = InsereLigneHT(BaseCpta, Ecriture)
        If Not IDLigne Is Nothing Then
            PileLignes.Push(IDLigne)
        Else
            Throw New Exception
        End If
        ' Création de la ligne de TVA
```

```
        IDLigne = InsereLigneTVA(BaseCpta, Ecriture)
        If IDLigne Is Nothing Then
            Throw New Exception
        End If
        Return True
    Catch ex As Exception ' Suppression des lignes déjà saisies pour éviter un déséquilibre du journal
        While PileLignes.Count > 0
            CType(BaseCpta.FactoryEcriture.Read(PileLignes.Pop), IBOEcriture3).Remove()
        End While
        Return False
    End Try
End Function

Function InsereLigneTTCMultiEch(ByRef BaseCpta As BSCPTAApplication100c, ByRef Ecriture As StrEcriture, _
    ByVal Echeance As IBOTiersReglement3, ByRef MontRestEcheance As Double) As IBIObjectID
    Dim LigneEcriture As IBOEcriture3
    Try
        LigneEcriture = BaseCpta.FactoryEcriture.Create
        With LigneEcriture
            .Tiers = BaseCpta.FactoryTiers.ReadNumero(Ecriture.Tiers)
            If Ecriture.CompteGTTC = "" Then
                Ecriture.CompteGTTC = .Tiers.CompteGPrinc.CG_Num
            End If
            .CompteG = BaseCpta.FactoryCompteG.ReadNumero(Ecriture.CompteGTTC)
            .CompteGContrepartie = BaseCpta.FactoryCompteG.ReadNumero(Ecriture.CompteGHT)
            .Journal = BaseCpta.FactoryJournal.ReadNumero(Ecriture.Journal)
            .Date = Ecriture.Date
            .DateSaisie = Now
            .EC_Piece = Ecriture.Piece
            .EC_Intitule = Ecriture.Libelle
            .EC_Sens = EcritureSensType.EcritureSensTypeDebit
            ' Calcul du montant de l'échéance en fonction du type de répartition
            Select Case Echeance.TRepart
```

```
            Case ReglementRepartitionType.ReglementRepartitionTypePourcent
                .EC_Montant = Ecriture.MontantTTC * Echeance.VRepart / 100
            Case ReglementRepartitionType.ReglementRepartitionTypeMontant
                If MontRestEcheance >= Echeance.VRepart Then
                    .EC_Montant = Echeance.VRepart
                Else
                    .EC_Montant = MontRestEcheance
                End If
            Case Else
                .EC_Montant = MontRestEcheance
        End Select
        MontRestEcheance -= .EC_Montant
        .Reglement = Echeance.Reglement
        ' Calcul de la date d'échéance en fonction de la date d'écriture
        .EC_Echeance = Echeance.Echeance(Ecriture.Date)
        .Write()
        Return .OID
    End With
  Catch ex As Exception
    Console.WriteLine("Erreur : {0}", ex.Message)
    Return Nothing
  End Try
End Function

Function InsereLigneTTCMonoEch(ByRef BaseCpta As BSCPTAApplication100c, ByVal Ecriture As StrEcriture) As IBIObjectID
  Try
    Dim LigneEcriture As IBOEcriture3
    LigneEcriture = BaseCpta.FactoryEcriture.Create
    With LigneEcriture
      .Tiers = BaseCpta.FactoryTiers.ReadNumero(Ecriture.Tiers)
      If Ecriture.CompteGTTC = "" Then
        Ecriture.CompteGTTC = .Tiers.CompteGPrinc.CG_Num
      End If
```

```
            .CompteG = BaseCpta.FactoryCompteG.ReadNumero(Ecriture.CompteGTTC)
            .CompteGContrepartie = BaseCpta.FactoryCompteG.ReadNumero(Ecriture.CompteGHT)
            .Journal = BaseCpta.FactoryJournal.ReadNumero(Ecriture.Journal)
            .Date = Ecriture.Date
            .DateSaisie = Now
            .EC_Piece = Ecriture.Piece
            .EC_Intitule = Ecriture.Libelle
            .EC_Sens = EcritureSensType.EcritureSensTypeDebit
            .EC_Montant = Ecriture.MontantTTC
            .Reglement = BaseCpta.FactoryReglement.List.Item(1)
            .EC_Echeance = Ecriture.Date
            .Write()
            Return .OID
        End With
    Catch ex As Exception
        Console.WriteLine("Erreur : {0}", ex.Message)
        Return Nothing
    End Try
End Function

Function InsereLigneHT(ByRef BaseCpta As BSCPTAApplication100c, ByVal Ecriture As StrEcriture) As IBIObjectID
    Try
        Dim LigneEcriture As IBOEcriture3
        LigneEcriture = BaseCpta.FactoryEcriture.Create
        With LigneEcriture
            .TiersContrepartie = BaseCpta.FactoryTiers.ReadNumero(Ecriture.Tiers)
            .Journal = BaseCpta.FactoryJournal.ReadNumero(Ecriture.Journal)
            .Date = Ecriture.Date
            .DateSaisie = Now
            .EC_Piece = Ecriture.Piece
            .EC_Intitule = Ecriture.Libelle
            .CompteG = BaseCpta.FactoryCompteG.ReadNumero(Ecriture.CompteGHT)
            .CompteGContrepartie = BaseCpta.FactoryCompteG.ReadNumero(Ecriture.CompteGTVA)
```

```
            .Taxe = .CompteG.Taxe
            .EC_Sens = EcritureSensType.EcritureSensTypeCredit
            .EC_Montant = Ecriture.MontantHT
            .WriteDefault() ' Si nécessaire génération de l'analytique
            Return .OID
        End With
    Catch ex As Exception
        Console.WriteLine("Erreur : {0}", ex.Message)
        Return Nothing
    End Try
End Function

Function InsereLigneTVA(ByRef BaseCpta As BSCPTAApplication100c, ByVal Ecriture As StrEcriture) As IBIObjectID
    Try
        Dim LigneEcriture As IBOEcriture3
        LigneEcriture = BaseCpta.FactoryEcriture.Create
        With LigneEcriture
            .TiersContrepartie = BaseCpta.FactoryTiers.ReadNumero(Ecriture.Tiers)
            .Journal = BaseCpta.FactoryJournal.ReadNumero(Ecriture.Journal)
            .Date = Ecriture.Date
            .DateSaisie = Now
            .EC_Piece = Ecriture.Piece
            .EC_Intitule = Ecriture.Libelle
            .CompteG = BaseCpta.FactoryCompteG.ReadNumero(Ecriture.CompteGTVA)
            .CompteGContrepartie = BaseCpta.FactoryCompteG.ReadNumero(Ecriture.CompteGTTC)
            .Taxe = CType(BaseCpta.FactoryCompteG.ReadNumero(Ecriture.CompteGHT), IBOCompteG3).Taxe
            .EC_Sens = EcritureSensType.EcritureSensTypeCredit
            .EC_Montant = Ecriture.MontantHT * (CType(BaseCpta.FactoryCompteG.ReadNumero(Ecriture.CompteGHT), _
                IBOCompteG3).Taxe.TA_Taux / 100)
            .Write()
            Return .OID
        End With
    Catch ex As Exception
```

```
            Console.WriteLine("Erreur : {0}", ex.Message)
            Return Nothing
        End Try
    End Function

    Function OuvreBaseCpta(ByRef BaseCpta As BSCPTAApplication100c, _
     ByVal NomBaseCpta As String, Optional ByVal Utilisateur As String = "<Administrateur>", _
     Optional ByVal MotDePasse As String = "") As Boolean
        Try
            BaseCpta.Name = NomBaseCpta
            BaseCpta.Loggable.UserName = Utilisateur
            BaseCpta.Loggable.UserPwd = MotDePasse
            BaseCpta.Open()
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur en ouverture de base comptable : {0}", ex.Message)
            Return False
        End Try
    End Function

    Function FermeBaseCpta(ByRef BaseCpta As BSCPTAApplication100c) As Boolean
        Try
            BaseCpta.Close()
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur en fermeture de base comptable : {0}", ex.Message)
            Return False
        End Try
    End Function

End Module
```

Les modèles de saisie

Les modèles de saisie permettent de simplifier et d'accélérer la saisie des écritures comptables.

Il est possible de lire, de modifier et de créer de nouveaux modèles de saisie par l'intermédiaire de Sage 100cloud Objets Métiers.

La gestion des modèles de saisie s'effectue par l'intermédiaire des objets suivants :

Objet	Description
IBOModeleEcriture3	Entête de modèle de saisie.
IBOModeleEcritureLigne3	Ligne de modèle de saisie (associée à un entête).
IBOModeleEcritureLigneA3	Ligne d'analytique (associée à une ligne de modèle de saisie).

➢ Voir Annexes – IBOModeleEcriture3, IBOModeleEcritureLigne3, IBOModeleEcritureLigneA3

Exemple :

Le programme ci-dessous permet de :

- *Modifier les différents champs de lignes d'analytique associées à la première ligne du modèle de saisie "Achats intracommunautaires" ;*
- *D'ajouter une nouvelle ligne d'analytique à la première ligne de ce modèle de saisie.*

Code source :

```
Option Strict Off

Imports System

Imports Objets100cLib


Module ModeleSaisie


  Structure LigneModeleSaisieA

    Dim PlanAna As IBPAnalytique3

    Dim Section As String

    Dim QteDevise As String

    Dim Montant As String
```

```
End Structure


Dim BaseCpta As BSCPTAApplication100c


Sub Main()
    Dim ModeleSaisie As IBOModeleEcriture3
    Dim LigneA As LigneModeleSaisieA
    Try
        BaseCpta = New BSCPTAApplication100c
        If OuvreBaseCpta(BaseCpta, "C:\Temp\Bijou.mae") Then
            ModeleSaisie = LisModeleSaisie("Achats intracommunautaires")
            AfficheModeleSaisie(ModeleSaisie)
            ' Modification de l'analytique de la 1ere ligne du modèle de saisie :
            ' Modification du montant (25%) de la 1ere ligne d'analytique du plan Activité :
            With LigneA
                .PlanAna = BaseCpta.FactoryAnalytique.ReadIntitule("Activité")
                .Section = Nothing
                .QteDevise = Nothing
                .Montant = "25%"
            End With
            ModifieLigneModeleSaisieA(ModeleSaisie, 1, 1, LigneA)
            ' Modification de la section (922ME4) de la 2eme ligne d'analytique du plan Activité :
            With LigneA
                .PlanAna = BaseCpta.FactoryAnalytique.ReadIntitule("Activité")
                .Section = "922ME4"
                .QteDevise = Nothing
                .Montant = Nothing
            End With
            ModifieLigneModeleSaisieA(ModeleSaisie, 1, 2, LigneA)
            ' Ajout d'une nouvelle ligne d'analytique sur le plan Activité :
            With LigneA
                .PlanAna = BaseCpta.FactoryAnalytique.ReadIntitule("Activité")
                .Section = "921SI3"
```

```
                .QteDevise = "=MacroSaisir()"
                .Montant = "=MacroEquilibrer()"
            End With
            AjouteLigneModeleSaisieA(ModeleSaisie, 1, LigneA)
            AfficheModeleSaisie(ModeleSaisie)
            Console.ReadLine()
        End If
    Catch ex As Exception
        Console.WriteLine("Erreur : " & ex.Message)
    Finally
        FermeBaseCpta(BaseCpta)
    End Try
End Sub

Sub AfficheModeleSaisie(ByVal ModeleSaisie As IBOModeleEcriture3)
    Try
        If Not ModeleSaisie Is Nothing Then
            Console.WriteLine("Modèle de saisie : " & ModeleSaisie.PI_Intitule)
            Console.WriteLine("Type : " & ModeleSaisie.JO_Type.ToString)
            Console.WriteLine("Raccourci : " & ModeleSaisie.PI_Raccourci)
            For Each ModeleSaisieLigne As IBOModeleEcritureLigne3 In ModeleSaisie.FactoryModeleEcritureLigne.List
                Dim i As Integer : i += 1
                Console.WriteLine(vbNewLine & "Ligne " & i & " :")
                With ModeleSaisieLigne
                    Console.WriteLine("Jour : " & .PG_Jour)
                    Console.WriteLine("N° pièce : " & .PG_Piece)
                    Console.WriteLine("Référence pièce : " & .PG_RefPiece)
                    Console.WriteLine("N° compte général : " & .CG_Num)
                    Console.WriteLine("N° compte tiers : " & .CT_Num)
                    Console.WriteLine("Code taxe : " & .TA_Code)
                    Console.Write("Provenance : ")
                    If .HasTA_Provenance Then
                        Console.Write(.TA_Provenance.ToString)
```

```
                Else
                    If .CalculTA_Provenance Then
                        Console.Write("Calcul")
                    Else
                        Console.Write("Aucune")
                    End If
                End If
                Console.WriteLine()
                Console.WriteLine("Libellé écriture : " & .PG_Intitule)
                Console.WriteLine("Echéance : " & .PG_Echeance)
                Console.WriteLine("Montant : " & .PG_Montant)
                Console.WriteLine("Sens : " & .PG_Sens.ToString)
            End With
            ' Affichage de la répartition analytique s'il en existe :
            If ModeleSaisieLigne.FactoryModeleEcritureLigneA.List.Count > 0 Then
                Console.WriteLine(vbNewLine & "Répartition analytique : ")
                For Each ModeleSaisieLigneA As IBOModeleEcritureLigneA3 In _
                    ModeleSaisieLigne.FactoryModeleEcritureLigneA.List
                    With ModeleSaisieLigneA
                        Console.WriteLine("Plan : " & .Analytique.A_Intitule)
                        Console.WriteLine("Section : " & .CA_Num)
                        Console.WriteLine("Qté/devise : " & .PA_Quantite)
                        Console.WriteLine("Montant : " & .PA_Montant)
                    End With
                Next
            End If
        Next
    End If
Catch ex As Exception
    Console.WriteLine("Erreur : " & ex.Message)
End Try
End Sub
```

```
Function LisModeleSaisie(ByVal IntituleModeleSaisie As String) As IBOModeleEcriture3
    Dim ModeleSaisie As IBOModeleEcriture3 = Nothing
    Try
        For Each ModeleSaisie In BaseCpta.FactoryModeleEcriture.List
            If ModeleSaisie.PI_Intitule = IntituleModeleSaisie Then
                Exit For
            Else
                ModeleSaisie = Nothing
            End If
        Next
        Return ModeleSaisie
    Catch ex As Exception
        Console.WriteLine("Erreur : " & ex.Message)
        Return Nothing
    End Try
End Function

Sub ModifieLigneModeleSaisieA(ByRef ModeleSaisie As IBOModeleEcriture3, ByVal NumLigneG As Integer, _
    ByVal NumLigneA As Integer, ByVal LigneA As LigneModeleSaisieA)
    Dim ModeleSaisieLigneG As IBOModeleEcritureLigne3
    Try
        ModeleSaisieLigneG = ModeleSaisie.FactoryModeleEcritureLigne.List(NumLigneG)
        For Each LigneACourante As IBOModeleEcritureLigneA3 In ModeleSaisieLigneG.FactoryModeleEcritureLigneA.List
            Dim NumLigneACourante As Integer
            If LigneACourante.Analytique Is LigneA.PlanAna Then
                NumLigneACourante += 1
                If NumLigneACourante = NumLigneA Then
                    With LigneACourante
                        .CA_Num = IIf(LigneA.Section Is Nothing, .CA_Num, LigneA.Section)
                        .PA_Quantite = IIf(LigneA.QteDevise Is Nothing, .PA_Quantite, LigneA.QteDevise)
                        .PA_Montant = IIf(LigneA.Montant Is Nothing, .PA_Montant, LigneA.Montant)
                        .Write()
                    End With
```

```
                Exit For
            End If
        End If
    Next
  Catch ex As Exception
    Console.WriteLine("Erreur : " & ex.Message)
  End Try
End Sub

Sub AjouteLigneModeleSaisieA(ByRef ModeleSaisie As IBOModeleEcriture3, ByVal NumLigneG As Integer, _
  ByVal LigneA As LigneModeleSaisieA)
  Dim NouvLigneA As IBOModeleEcritureLigneA3
  Try
    NouvLigneA = CType(ModeleSaisie.FactoryModeleEcritureLigne.List(NumLigneG), _
      IBOModeleEcritureLigne3).FactoryModeleEcritureLigneA.Create
    With NouvLigneA
      .Analytique = IIf(LigneA.PlanAna Is Nothing, .Analytique, LigneA.PlanAna)
      .CA_Num = IIf(LigneA.Section Is Nothing, .CA_Num, LigneA.Section)
      .PA_Quantite = IIf(LigneA.QteDevise Is Nothing, .PA_Quantite, LigneA.QteDevise)
      .PA_Montant = IIf(LigneA.Montant Is Nothing, .PA_Montant, LigneA.Montant)
      .Write()
    End With
  Catch ex As Exception
    Console.WriteLine("Erreur : " & ex.Message)
  End Try
End Sub

Function OuvreBaseCpta(ByRef BaseCpta As BSCPTAApplication100c, _
 ByVal NomBaseCpta As String, Optional ByVal Utilisateur As String = "<Administrateur>", _
 Optional ByVal MotDePasse As String = "") As Boolean
  Try
    BaseCpta.Name = NomBaseCpta
    BaseCpta.Loggable.UserName = Utilisateur
```

```
            BaseCpta.Loggable.UserPwd = MotDePasse
            BaseCpta.Open()
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur en ouverture de base comptable : {0}", ex.Message)
            Return False
        End Try
    End Function

    Function FermeBaseCpta(ByRef BaseCpta As BSCPTAApplication100c) As Boolean
        Try
            BaseCpta.Close()
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur en fermeture de base comptable : {0}", ex.Message)
            Return False
        End Try
    End Function

End Module
```

Les documents commerciaux

Les documents

Cet exemple illustre la création d'un document de vente de type Devis pour un article nomenclaturé. Le devis est crée pour le client « CARAT » en utilisant l'article « ENSHF ». Le type de la nomenclature de cet article est « commerciale / composé ». Le code source contient une fonction principale « CreationDevis ». C'est à partir de celle-ci que l'entête et les lignes de documents vont être générés.

Exemple :

La gestion des documents s'effectue par l'intermédiaire des objets suivants :

Objet	Description
IBODocumentVente3	Entête des documents des ventes

Objet	Description
IBODocumentAchat3	Entête des documents des achats
IBODocumentStock3	Entête des documents des stocks
IBODocumentInterne3	Entête des documents internes
IBODocumentVenteLigne3	Ligne des documents des ventes
IBODocumentAchatLigne3	Ligne des documents des achats
IBODocumentStockLigne3	Ligne des documents des stocks
IBODocumentInterneLigne3	Ligne des documents internes

Les objets correspondants aux entêtes de documents offrent plusieurs méthodes *SetDefaultxxx()* permettant d'initialiser automatiquement les valeurs. Dans l'exemple, la méthode *SetDefaultClient()* affecte automatiquement à l'entête de document, la plupart des informations relatives au client : Dépôt client, mode d'expédition, condition de livraison, catégorie comptable, catégorie tarifaire…

Pour les lignes de documents, il existe également des méthodes *SetDefaultxxx()* qui permettent certains automatismes. Par exemple, *SetDefaultArticle()* retourne automatiquement le prix de l'article.

Code source :

```
Option Strict Off

Imports System

Imports Objets100cLib


Module CreationDevis

    Dim bCpta As New BSCPTAApplication100c

    Dim bCial As New BSCIALApplication100c


    Public Sub Main()

        Try

            If OpenBase(bCpta, bCial, "C:\Temp\Bijou.mae", "C:\Temp\Bijou.gcm") Then

                If CreateDevis(bCial, bCpta) Then

                    Console.WriteLine("Document créé")

                End If

            End If

        Catch ex As Exception
```

```
        Console.WriteLine("Erreur dans la création du document")
    Finally
        CloseBase(bCial)
    End Try
End Sub

Public Function CreateDevis(ByRef bCial As BSCIALApplication100c, ByRef bCpta As BSCPTAApplication100c) As Boolean
    Try
        Dim DocEntete As IBODocumentVente3
        Dim mIboArt As IBOArticleNomenclature3
        'Création de l'entête
        DocEntete = bCial.FactoryDocumentVente.CreateType(DocumentType.DocumentTypeVenteDevis)
        With DocEntete
            .SetDefaultClient(bCpta.FactoryClient.ReadNumero("CARAT"))
            .DO_Date = Now
            .SetDefaultDO_Piece()
            .Write()
        End With
        'Création de la ligne pour l'article composé
        Dim DocLigneCompose As IBODocumentVenteLigne3 = DocEntete.FactoryDocumentLigne.Create
        With DocLigneCompose
            .SetDefaultArticle(bCial.FactoryArticle.ReadReference("ENSHF"), 1)
            .ArticleCompose = bCial.FactoryArticle.ReadReference("ENSHF")
            .Write()
        End With
        ' Recherche du taux de remise pour la catégorie tarifaire de CARAT
        Dim dRemise = bCpta.FactoryClient.ReadNumero("CARAT").TauxRemise
        'Création des lignes pour les articles composants
        For Each mIboArt In bCial.FactoryArticle.ReadReference("ENSHF").FactoryArticleNomenclature.List
            Dim DocLigneComposant As IBODocumentVenteLigne3 = DocEntete.FactoryDocumentLigne.Create
            With DocLigneComposant
                .SetDefaultArticle(mIboArt.ArticleComposant, mIboArt.NO_Qte)
                .ArticleCompose = mIboArt.Article
```

```
                .Remise.Remise(1).REM_Type = RemiseType.RemiseTypePourcent
                .Remise.Remise(1).REM_Valeur = dRemise
                .Write()
            End With
        Next
        Return True
    Catch ex As Exception
        Console.WriteLine("Erreur lors de la création du devis : " & ex.Message)
        Return False
    End Try
End Function

Public Function OpenBase(ByRef BaseCpta As BSCPTAApplication100c, ByRef BaseCial As BSCIALApplication100c, _
    ByVal sMae As String, ByVal sGcm As String, Optional ByVal sUid As String = "<Administrateur>", _
    Optional ByVal sPwd As String = "") As Boolean
    Try
        bCpta.Name = sMae
        BaseCial.CptaApplication = BaseCpta
        BaseCial.Name = sGcm
        BaseCial.Loggable.UserName = sUid
        BaseCial.Loggable.UserPwd = sPwd
        BaseCial.Open()
        Return True
    Catch ex As Exception
        Console.WriteLine("Erreur lors de l'ouverture du fichier gcm : " & ex.Message)
        Return False
    End Try
End Function

Private Function CloseBase(ByRef bCial As BSCIALApplication100c) As Boolean
    Try
        If bCial.IsOpen Then bCial.Close()
        Return True
```

```
        Catch ex As Exception
            Console.WriteLine("Erreur lors de la fermeture de la base : " & ex.Message)
            Return False
        End Try
    End Function

End Module
```

Les réglements en Gestion commerciale

Il est possible de lire, de modifier et de créer de nouveaux réglements par l'intermédiaire de Sage 100cloud Objets Métiers.

Description complète Voir Objets Réglements

La gestion des réglements s'effectue par l'intermédiaire des objets suivants :

Objet	Description
IBODocumentReglement	Objet Règlement en Gestion commerciale.
IBODocumentReglementFactory	Méthodes de lecture et écriture des réglements
IBODocumentReglementEcheance	Objet Imputations entre réglements et échéances
IBODocumentReglementEcheanceReglementFactory	Méthodes d'Imputations entre échéances et réglements
IBODocumentReglementEcheanceEcheanceFactory	Méthodes d'Imputations entre réglements et échéances
IPMReglerEcheances	Processus d'Imputations entre réglements et échéances

Exemple :

Le programme ci-dessous crée le règlement comptant d'une facture client :

- *Création d'un règlement pour le reste à payer de la facture;*
- *Création du processus avec le règlement et les échéances de la facture en paramètres.*

Code source :

```
    private void TesProcessReglementComptant(ref BSCPTAApplication100c BaseCpta, ref BSCIALApplication100c
BaseCial)
    {
        try
        {
            string piece = "FA00028";

// Recherche du document et du client associé
```

```
            IBODocumentVente3 doc =
BaseCial.FactoryDocumentVente.ReadPiece(DocumentType.DocumentTypeVenteFacture, piece);
            IBOTiersPart3 client = doc.TiersPayeur;

// création du règlement

            IBODocumentReglement iReglt = (IBODocumentReglement)BaseCial.FactoryDocumentReglement.Create();
           iReglt.TiersPayeur = client;
           iReglt.RG_Date = DateTime.Now; ;
           iReglt.RG_Reference = "Référence";
           iReglt.RG_Libelle = "Libellé";
           iReglt.RG_Montant = doc.DO_NetAPayer- doc.DO_MontantRegle;
           iReglt.Journal = BaseCial.CptaApplication.FactoryJournal.ReadNumero("BRD");
           iReglt.CompteG = client.CompteGPrinc;
           iReglt.WriteDefault();

// création du Processus régler les échéances

            IPMReglerEcheances pRegler = BaseCial.CreateProcess_ReglerEcheances();

          pRegler.Reglement = iReglt;
          foreach (IBODocumentEcheance3 iEcheance in doc.FactoryDocumentEcheance.List)
               pRegler.AddDocumentEcheance(iEcheance);
          pRegler.Process();

// Lecture du résultat du Processus

          string txt = "";

          foreach (IBODocumentReglementEcheance iECh in pRegler.ListLignesResult)
            {
               txt = iECh.Echeance.DR_Date.ToString() + " " + iECh.RC_Montant.ToString();
               MessageBox.Show(txt, "Echéance");
            }
         }
         catch (Exception ex)
         {
            MessageBox.Show(ex.Message, "Resultat Errors");
         }
      }
```

Les Informations complémentaires en Gestion commerciale

Il est possible de lire, de modifier des informations complémentaires par l'intermédiaire de Sage 100cloud Objets Métiers.

La gestion des informations complémentaires s'effectue par l'intermédiaire des Factory suivants :

IBOClient3.FactoryInfoComplement pour accéder aux informations clients

IBODocumentVente3.FactoryInfoComplement pour accéder aux informations Document Vente

IBODocumentVenteLigne3.FactoryInfoComplement pour accéder aux informations Lignes de documents

Objet	Description
IBOInfoComplementClient	Objet Information Client en Gestion commerciale.
IBOInfoComplementClientFactory	Méthodes d'accés aux Objet Information Client
IBOInfoComplementEntete	Objet Information Document en Gestion commerciale.

IBOInfoComplementEnteteFactory	Méthodes d'accés aux Objet Information Document
IBOInfoComplementDocligne	Objet Information Ligne en Gestion commerciale.
IBOInfoComplementDocligneFactory	Méthodes d'accés aux Objet Information Ligne

Exemple :

Le programme ci-dessous accède et modifie des informations complémentaires :

- *Lecture de l'information complémentaire du client "DIAMA" qui a le code "CODE2" , doublement de sa valeur et réécriture de l'information comlémentaire du client.*
- *Lecture de l'information complémentaire du document "FA00005" qui a le code "CODEFA00005" , Ajout d'un jour à sa valeur et réécriture de l'information comlémentaire.*
- *Pour toutes les lignes du document "FA00005", lecture de toutes les informations complémentaires.*

Code source :

```
private void testInfosComplementaire(ref BSCPTAApplication100c BaseCpta, ref BSCIALApplication100c BaseCial)
    {
        try
        {
          // Lecture et modification d'une information complémentaire Client
            IBOClient3 client = BaseCpta.FactoryClient.ReadNumero("DIAMA");
            IBOInfoComplementClientFactory pInfoClifact = client.FactoryInfoComplement;

            if (pInfoClifact.ExistCode("CODE2"))
            {
                IBOInfoComplementClient InfoCli = pInfoClifact.ReadCode("CODE2");
                InfoType iInfoT = InfoCli.CI_Type;
                double dVal = 0.0;

                if (iInfoT == InfoType.eCDT_MONTANT || iInfoT == InfoType.eCDT_VALEUR)
                {
                    dVal = Convert.ToDouble(InfoCli.CI_Valeur) * 2;
                    InfoCli.CI_Valeur = dVal.ToString();
                    InfoCli.Write();
                }
           // Lecture et modification d'une information complémentaire Document de vente
                IBODocumentVente3 DocVente =
BaseCial.FactoryDocumentVente.ReadPiece(DocumentType.DocumentTypeVenteFacture, "FA00005");
                IBOInfoComplementEnteteFactory InfoFactDoc = DocVente.FactoryInfoComplement;
                if (InfoFactDoc.ExistCode("CODEFA00005"))
                {
                    IBOInfoComplementEntete InfoDoc = InfoFactDoc.ReadCode("CODEFA00005");

                    InfoType iInfoD = InfoDoc.DI_Type;
                    DateTime dDate = new DateTime(2020, 1, 1);

                    if (iInfoT == InfoType.eCDT_DATE)
                    {
                        dDate = Convert.ToDateTime(InfoDoc.DI_Valeur);
                        dDate.AddDays(1);
                        InfoCli.CI_Valeur = dDate.ToString();
                        InfoCli.Write();
                    }
```

```
            }
        // Lecture de toutes les  informations complémentaires Lignes du document de vente
            foreach (IBODocumentVenteLigne3 DocLigne in DocVente.FactoryDocumentLigne.List)
            {
                IBOInfoComplementDocligneFactory InfoFactLigne = DocLigne.FactoryInfoComplement;
                foreach (IBOInfoComplementDocligne iInfo in DocLigne.FactoryInfoComplement.List)
                    MessageBox.Show(iInfo.DC_Intitule + " " + iInfo.DC_Code + " " + iInfo.DC_Valeur,
DocLigne.Article.AR_Ref);
            }
          }

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, "Resultat Errors");
        }
    }
```

Développement .Net avec Sage 100cloud Objets Métiers

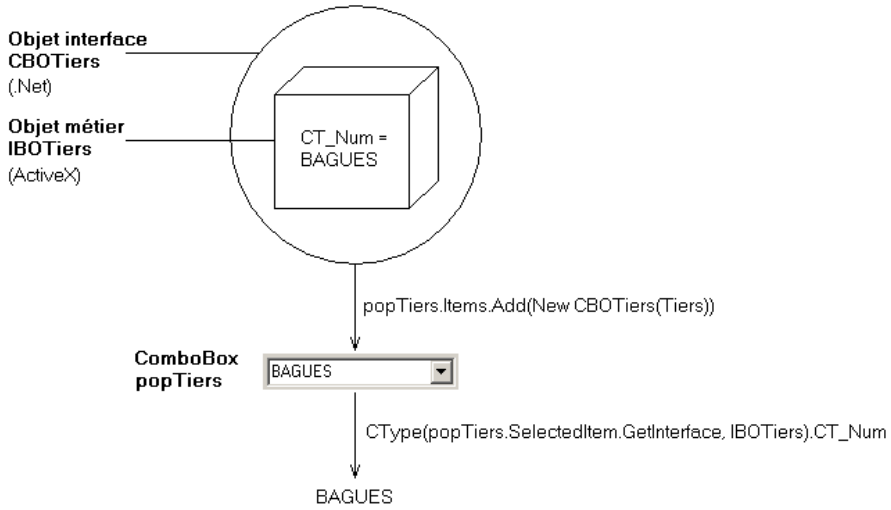
Sage 100cloud Objets Métiers se présente sous la forme d'un composant ActiveX (technologie COM), tandis que les différents composants de l'environnement de développement Visual Studio .Net sont des composants .Net (ex : contrôles Windows).

Les différents exemples de ce manuel s'exécutent en mode console et n' interagissent pas avec les composants Net.

Pour utiliser Sage 100cloud Objets Métiers avec des composants .Net, il est possible de les encapsuler dans des objets .Net afin de pouvoir les manipuler comme des objets .Net.

Exemple :

*En encapsulant un objet métier tiers (*IBOTiers3*) dans un objet interface .Net (*CBOTiers*), il est possible de l'utiliser avec d'autres composants .Net (ex : une ComboBox).*
Il est ensuite possible d'accéder aux propriétés de l'objet métier tiers par l'intermédiaire de la méthode GetInterface *de l'objet interface .Net :*



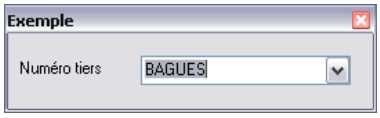
Code source des classes d'interface :

```
Imports Objets100cLib

Module Adapter

    ' Classe d'interface commune à tous les Objets Métiers 100

    Public Class ControlObjet100


        Private mPersistObject As IBIPersistObject


        Public Sub New(ByVal obj As IBIPersistObject)

            mPersistObject = obj

        End Sub


        Public Overridable Function GetObject() As IBIPersistObject

            Return mPersistObject

        End Function


        Public Overrides Function ToString() As String

            Return ""

        End Function

    End Class


    ' Classe d'interface objet tiers (IBOTiers)

    Public Class CBOTiers

        Inherits ControlObjet100


        Public Sub New(ByVal obj As IBIPersistObject)

            MyBase.New(obj)

        End Sub


        Public Overrides Function ToString() As String

            Return GetInterface().CT_Num

        End Function
```

```
    Public Function GetInterface() As IBOTiers3
        Return GetObject()
    End Function

  End Class

End Module
```

Exemple :

Utilisation d'Objets Métiers avec des composants .Net



L'application ci-dessous réalise les opérations suivantes :

- Remplissage d'une ComboBox avec le numéro de compte des tiers d'une base de données comptable.
- Lorsqu'un tiers est sélectionné dans cette ComboBox, une boîte de dialogue s'ouvre et affichage l'intitulé du tiers.

Code source de l'exemple :

```
Imports Objets100cLib
Public Class FrmExemple
  Inherits System.Windows.Forms.Form

#Region " Windows Form Designer generated code "

  Public Sub New()
    MyBase.New()

    'This call is required by the Windows Form Designer.
    InitializeComponent()
```

```
    'Add any initialization after the InitializeComponent() call


  End Sub


  'Form overrides dispose to clean up the component list.
  Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
    If disposing Then
      If Not (components Is Nothing) Then
        components.Dispose()
      End If
    End If
    MyBase.Dispose(disposing)
  End Sub


  'Required by the Windows Form Designer
  Private components As System.ComponentModel.IContainer


  'NOTE: The following procedure is required by the Windows Form Designer
  'It can be modified using the Windows Form Designer.
  'Do not modify it using the code editor.
  Friend WithEvents Label1 As System.Windows.Forms.Label
  Friend WithEvents CbTiers As System.Windows.Forms.ComboBox
  <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
    Me.Label1 = New System.Windows.Forms.Label
    Me.CbTiers = New System.Windows.Forms.ComboBox
    Me.SuspendLayout()
    '
    'Label1
    '
    Me.Label1.FlatStyle = System.Windows.Forms.FlatStyle.System
    Me.Label1.Location = New System.Drawing.Point(8, 18)
    Me.Label1.Name = "Label1"
    Me.Label1.Size = New System.Drawing.Size(80, 16)
```

```
        Me.Label1.TabIndex = 0
        Me.Label1.Text = "Numéro tiers"
        '
        'CbTiers
        '
        Me.CbTiers.Location = New System.Drawing.Point(96, 16)
        Me.CbTiers.Name = "CbTiers"
        Me.CbTiers.Size = New System.Drawing.Size(152, 21)
        Me.CbTiers.TabIndex = 1
        Me.CbTiers.Text = "ComboBox1"
        '
        'FrmExemple
        '
        Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
        Me.ClientSize = New System.Drawing.Size(264, 54)
        Me.Controls.Add(Me.CbTiers)
        Me.Controls.Add(Me.Label1)
        Me.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedToolWindow
        Me.Name = "FrmExemple"
        Me.Text = "Exemple"
        Me.ResumeLayout(False)

    End Sub

#End Region
    Private BaseCpta As BSCPTAApplication100c
    Private Tiers As IBOTiers3

    Private Sub frmExemple_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        BaseCpta = New BSCPTAApplication100c
        BaseCpta.Name = "C:\temp\Bijou.mae"
        BaseCpta.Open()
        For Each Tiers As IBOTiers3 In BaseCpta.FactoryTiers.ListOrderNumero
```

```
            CbTiers.Items.Add(New CBOTiers(Tiers))
        Next
    End Sub

    Private Sub frmExemple_Leave(ByVal sender As System.Object, ByVal e As System.ComponentModel.CancelEventArgs) Handles MyBase.Closing
        If Not BaseCpta Is Nothing Then
            If BaseCpta.IsOpen Then
                BaseCpta.Close()
            End If
        End If
    End Sub

    Private Sub popTiers_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles CbTiers.SelectedIndexChanged
        MessageBox.Show("Intitulé du tiers : " & CType(CbTiers.SelectedItem.GetInterface, IBOTiers3).CT_Intitule)
    End Sub

End Class
```

Processus Encoder

Création d'une facture client

L'exemple suivant permet, comme l'exemple présenté sous le paragraphe ***Les écritures comptables***, de générer une pièce dans la base de comptabilité. Cependant, pour cet exemple, l'insertion de la pièce comptable est réalisée par implémentation du processus de saisie de pièce comptable : *IPMEncoder*.

Ainsi, l'exemple suivant permet de créer une pièce comptable (facture client) en générant automatiquement les écritures d'échéances du compte tiers, les écritures analytiques, ainsi que les écritures TTC, HT et TVA. Cet exemple met également en œuvre une méthode permettant de récupérer la collection des éventuelles erreurs (*RecupErrors()*).

Pour le compte tiers *TOPAZ*, le mode de règlement est défini de la manière suivante :

Valeur	Jour	Condition	Mode de règlement
50%	60	Jour(s) net(s)	Chèque

Equilibre	90	Fin mois	LCR Acceptée

Sur le compte général *701019*, l'option *Saisie analytique* est également cochée.

Code source

```
Imports Objets100cLib

Module ModEncoder

  'Objet base comptable

  Dim oCpta As BSCPTAApplication100c

  'emplacement du fichier comptable

  Dim sPathMae As String = "C:\Temp\Bijou.mae"


  Sub Main()

    Try

      'Instanciation de l'objet base comptable

      oCpta = New BSCPTAApplication100c

      'Ouverture de la base

      If OpenBase(oCpta, sPathMae) Then

        'Création d'un objet processus Saisie de pièce comptable

        Dim mProcessEncoder As IPMEncoder = oCpta.CreateProcess_Encoder

        'Affectation des propriétes globales du processus

        EntetePiece(mProcessEncoder)

        'Génération des écritures de la pièce

        TiersPart(mProcessEncoder)

        'Si les écritures du processus ne sont pas correctes

        If Not mProcessEncoder.CanProcess Then

          'Récupération des erreurs

          RecupError(mProcessEncoder)

        Else

          'Gérération de la pièce dans la base

          mProcessEncoder.Process()

        End If

      End If

    Catch ex As Exception
```

```
            Console.WriteLine("Erreur : " & ex.Message)
        End Try
        CloseBase(oCpta)
    End Sub

    Public Sub EntetePiece(ByRef mP As IPMEncoder)
        Try
            'Affectation de la date
            mP.Date = #3/22/2009#
            'Test sur l'existence du journal
            If oCpta.FactoryJournal.ExistNumero("VTE") Then
                'Affectation du journal
                mP.Journal = oCpta.FactoryJournal.ReadNumero("VTE")
                'Affectation du numéro de pièce
                mP.EC_Piece = mP.Journal.NextEC_Piece(#3/22/2009#)
            End If
            'Affectation de l'intitulé
            mP.EC_Intitule = "Facture TOPAZ"
            'Affectation de la référence pièce
            mP.EC_RefPiece = "A005"
            'Affectation du numéro de facture
            mP.EC_Reference = "A005"
            'Génération automatique des écritures analytiques
            mP.bAnalytiqueAuto = True
            'Génération automatique des écritures d'échéances
            mP.bMultiEcheanceAuto = True
        Catch ex As Exception
            Console.WriteLine("Erreur : " & ex.Message)
        End Try
    End Sub

    Public Sub TiersPart(ByRef mP As IPMEncoder)
        Dim mTiers As IBOTiers3
```

```
    Dim mCpt, mCptCharge, mCptTVA As IBOCompteG3
    Dim dMnt As Double = 40000
    Try
      mTiers = Nothing
      mCpt = Nothing
      mCptCharge = Nothing
      mCptTVA = Nothing
      'Test sur l'existence du numéro de compte tiers
      If oCpta.FactoryTiers.ExistNumero("TOPAZ") Then
        'Affectation du compte tiers
        mTiers = oCpta.FactoryTiers.ReadNumero("TOPAZ")
        'Affectation du compte général principal
        mCpt = mTiers.CompteGPrinc
      End If
      'Affectation du compte de charge
      If oCpta.FactoryCompteG.ExistNumero("701019") Then mCptCharge = oCpta.FactoryCompteG.ReadNumero("701019")
      'Affectation du compte de TVA
      If oCpta.FactoryCompteG.ExistNumero("4457119") Then mCptTVA = oCpta.FactoryCompteG.ReadNumero("4457119")
      'Génération des écritures d'échéances, HT et TVA
      mP.AddTiersPart(mTiers, dMnt, mCptCharge, mCptTVA, mCpt)
    Catch ex As Exception
      Console.WriteLine("Erreur : " & ex.Message)
    End Try
  End Sub

  Public Sub RecupError(ByRef mP As IPMEncoder)
    Try
      'Boucle sur les erreurs contenues dans la collection
      For i As Integer = 1 To mP.Errors.Count
        'Récupération des éléments erreurs
        Dim iFail As IFailInfo = mP.Errors.Item(i)
        'récupération du numéro d'erreur, de l'indice et de la description de l'erreur
        Console.WriteLine("Code Erreur : " & iFail.ErrorCode & " Indice : " & iFail.Indice & _
```

```
            " Description : " & iFail.Text)
        Next
    Catch ex As Exception
        Console.WriteLine("Erreur : " & ex.Message)
    End Try
End Sub

Public Function OpenBase(ByRef BaseCpta As BSCPTAApplication100c, ByVal sMae As String, _
Optional ByVal sUid As String = "<Administrateur>", _
    Optional ByVal sPwd As String = "") As Boolean
    Try
        'Affectation de l'emplacement du fichier comptable
        BaseCpta.Name = sMae
        'Affectation du code utilisateur
        BaseCpta.Loggable.UserName = sUid
        'Affectation du mot de passe
        BaseCpta.Loggable.UserPwd = sPwd
        'Ouverture de la base comptable
        BaseCpta.Open()
        Return True
    Catch ex As Exception
        Console.WriteLine("Erreur lors de l'ouverture du fichier mae : " & ex.Message)
        Return False
    End Try
End Function

Private Function CloseBase(ByRef BaseCpta As BSCPTAApplication100c) As Boolean
    Try
        'Si la base est ouverte, alors fermeture de la base
        If BaseCpta.IsOpen Then BaseCpta.Close()
        Return True
    Catch ex As Exception
        Console.WriteLine("Erreur lors de la fermeture de la base : " & ex.Message)
```

```
        Return False
    End Try
  End Function
End Module
```

Processus Création de document

Création d'une facture client

L'exemple suivant permet de créer une facture de vente sur un article nomenclaturé. A la différence de la création d'un document en passant par les objets « standards » (IBODocumentVente3 et IBODocumentVenteLigne3), l'ajout d'une ligne de document sur un article nomenclaturé en passant par le processus, ajoute automatiquement les lignes d'articles composant cette nomenclature au document.

Dans cet exemple, l'article utilisé « ENSHF » est de nomenclature « commerciale/composé » dont la composition est la suivante :

Référence	Quantité	Coût Std	Prix vente
MODIV01	1	155,00	310,00
MOOR001	1	245,00	539,00
STYPLOR	2	124,36	250,00

Ainsi, dans le processus de création de document, lors de l'ajout de la ligne d'article sur ENSHF, trois lignes de document seront automatiquement ajoutées pour les articles MODIV01, MOOR001 et STYPLOR.

Code source

```
Option Strict Off

Imports Objets100cLib

Imports System


Module ProcessCreateDocument

  'Objet base commerciale

  Dim oCial As BSCIALApplication100c

  'emplacement du fichier commercial

  Dim sPathGcm As String = "C:\Temp\Bijou.gcm"
```

```
Sub Main()
    Try
        'Instanciation de l'objet base commercial
        oCial = New BSCIALApplication100c
        'Ouverture de la base
        If OpenBase(oCial, sPathGcm) Then
            'Création d'un objet processus "Création de document"
            Dim mProcessDoc As IPMDocument = oCial.CreateProcess_Document(DocumentType.DocumentTypeVenteFacture)
            'Conversion du document du processus (IBODocument3) dans le type du document de destination : Facture de vente (IBODocumentVente3)
            Dim mDoc As IBODocumentVente3 = CType(mProcessDoc.Document, IBODocumentVente3)
        'Indique au document qu'il ne doit pas recalculer les totaus automatiquement à chaque modification ou ajout de lignes
        mDoc.SetAutoRecalculTotaux(false);
            'Affectation du client au document
            mDoc.SetDefaultClient(oCial.CptaApplication.FactoryClient.ReadNumero("CARAT"))
            'Ajout d'une ligne sur l'article ENSHF de nomenclature commerciale/composé et
            'conversion dans le type de ligne de document de destination (IBODocumentVenteLigne3).
            'Lors de l'ajout de cette ligne, les autres lignes composant la nomenclature sont également ajoutées
            Dim mLig As IBODocumentVenteLigne3 = CType(mProcessDoc.AddArticle(oCial.FactoryArticle.ReadReference("ENSHF"), 1) _
                , IBODocumentVenteLigne3)
            'Parcours de toutes les lignes du document
            For Each mLig In mDoc.FactoryDocumentLigne.List
                'Application de la remise par défaut pour chacune des lignes
                mLig.SetDefaultRemise()
            Next
            'Si le document est cohérent et peut être écrit en base
            If Not mProcessDoc.CanProcess Then
                'Récupération des erreurs
                RecupError(mProcessDoc)
            Else
                'Gérération de document dans la base
                mProcessDoc.Process()
```

```
            End If
        End If
    Catch ex As Exception
        Console.WriteLine("Erreur : " & ex.Message)
    Finally
        'Fermeture de la connexion
        CloseBase(oCial)
    End Try
End Sub
Public Sub RecupError(ByRef mP As IPMDocument)
    Try
        'Boucle sur les erreurs contenues dans la collection
        For i As Integer = 1 To mP.Errors.Count
            'Récupération des éléments erreurs
            Dim iFail As IFailInfo = mP.Errors.Item(i)
            'récupération du numéro d'erreur, de l'indice et de la description de l'erreur
            Console.WriteLine("Code Erreur : " & iFail.ErrorCode & " Indice : " & iFail.Indice & _
            " Description : " & iFail.Text)
        Next
    Catch ex As Exception
        Console.WriteLine("Erreur : " & ex.Message)
    End Try
End Sub

Public Function OpenBase(ByRef BaseCial As BSCIALApplication100c, ByVal sGcm As String, _
    Optional ByVal sUid As String = "<Administrateur>", _
    Optional ByVal sPwd As String = "") As Boolean
    Try
        'Affectation de l'emplacement du fichier commercial
        BaseCial.Name = sGcm
        'Affectation du code utilisateur
        BaseCial.Loggable.UserName = sUid
        'Affectation du mot de passe
```

```
            BaseCial.Loggable.UserPwd = sPwd
            'Ouverture de la base commerciale
            BaseCial.Open()
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur lors de l'ouverture du fichier gcm : " & ex.Message)
            Return False
        End Try
    End Function

    Private Function CloseBase(ByRef BaseCial As BSCIALApplication100c) As Boolean
        Try
            'Si la base est ouverte, alors fermeture de la base
            If BaseCial.IsOpen Then BaseCial.Close()
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur lors de la fermeture de la base : " & ex.Message)
            Return False
        End Try
    End Function
End Module
```

Processus de contrôle qualité

L'exemple suivant permet de créer un contrôle qualité sur les achats, pour l'article BAAR01 pour lequel une quantité de 50 est placée sur un emplacement de contrôle.

Le contrôle qualité mis en place dans cet exemple permet de valider une quantité de 42 BAAR01 et de retourner les 8 restants. Ainsi, la validation génèrera un mouvement de transfert, et le retour génèrera une facture de retour fournisseur.

Code source

```
Option Strict Off

Imports Objets100cLib

Imports System
```

```
Module ProcessusControleQualite

    'Objet base commerciale
    Dim oCial As BSCIALApplication100c
    'emplacement du fichier commercial
    Dim sPathGcm As String = "C:\Temp\Bijou.gcm"

    Sub Main()
        Try
            'Instanciation de l'objet base commercial
            oCial = New BSCIALApplication100c
            'Ouverture de la base
            If OpenBase(oCial, sPathGcm) Then
                'Création d'un objet processus "Contrôle qualité"
                Dim pQualite As IPMControlerQualite = oCial.CreateProcess_ControlerQualite
                'Initialisation des paramètres de sélection des lignes à contrôler
                pQualite.Domaine = DomaineType.DomaineTypeAchat
                pQualite.PeriodDO_Date.Start = #1/1/2011#
                pQualite.PeriodDO_Date.End = #12/31/2011#
                'récupération de la collection des lignes correspondant aux paramètres de sélection
                For Each mLigEmpl As IBODocumentLigneEmplacement In pQualite.QueryLigneEmplacement
                    'Test pour ne prendre que la ligne correspondant à l'article BAAR01
                    If mLigEmpl.DocumentLigne.Article.AR_Ref = "BAAR01" Then
                        'Définit le type de document pour le retour fournisseur : Facture
                        'Cette propriété doit être renseignée avant de générer les documents
                        'de sortie (appel de Valider, Retourner ou MettreRebut)
                        pQualite.TypeDocumentRetour = DocumentType.DocumentTypeAchatFacture
                        'Valide l'emplacement pour une quantité de 42
                        pQualite.Valider(mLigEmpl, 42)
                        'Retourne la quantité de 8
                        pQualite.Retourner(mLigEmpl, 8, "Erreur de commande")
                    End If
                Next
```

```
            'Si le processus ne peut pas être exécuté
            If Not pQualite.CanProcess Then
                'Récupération des erreurs
                RecupError(pQualite)
            Else
                'Gérération d'un mouvement de transfert pour une quantité de 42 et une facture de retour fournisseur pour une quantité de
8
                pQualite.Process()
            End If
        End If
    Catch ex As Exception
        Console.WriteLine("Erreur : " & ex.Message)
    Finally
        'Fermeture de la connexion
        CloseBase(oCial)
    End Try
End Sub
Public Sub RecupError(ByRef mP As IPMControlerQualite)
    Try
        'Boucle sur les erreurs contenues dans la collection
        For i As Integer = 1 To mP.Errors.Count
            'Récupération des éléments erreurs
            Dim iFail As IFailInfo = mP.Errors.Item(i)
            'récupération du numéro d'erreur, de l'indice et de la description de l'erreur
            Console.WriteLine("Code Erreur : " & iFail.ErrorCode & " Indice : " & iFail.Indice & _
            " Description : " & iFail.Text)
        Next
    Catch ex As Exception
        Console.WriteLine("Erreur : " & ex.Message)
    End Try
End Sub

Public Function OpenBase(ByRef BaseCial As BSCIALApplication100c, ByVal sGcm As String, _
    Optional ByVal sUid As String = "<Administrateur>", _
```

```
        Optional ByVal sPwd As String = "") As Boolean
        Try
            'Affectation de l'emplacement du fichier commercial
            BaseCial.Name = sGcm
            'Affectation du code utilisateur
            BaseCial.Loggable.UserName = sUid
            'Affectation du mot de passe
            BaseCial.Loggable.UserPwd = sPwd
            'Ouverture de la base commerciale
            BaseCial.Open()
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur lors de l'ouverture du fichier gcm : " & ex.Message)
            Return False
        End Try
    End Function

    Private Function CloseBase(ByRef BaseCial As BSCIALApplication100c) As Boolean
        Try
            'Si la base est ouverte, alors fermeture de la base
            If BaseCial.IsOpen Then BaseCial.Close()
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur lors de la fermeture de la base : " & ex.Message)
            Return False
        End Try
    End Function
End Module
```

Processus d’application des barèmes

L’exemple suivant permet d’appliquer les barèmes sur un bon de commande de vente, afin que deux lignes de remises soient automatiquement ajoutées au document.

Dans le fichier commercial, deux barèmes sont créés :

- Un barème nommé « Promotion Avril » pour une promotion de 5% sur tous les articles sur la période du mois d'avril 2011.
- Un barème nommé « Promotion montre » pour une promotion sur des articles associés aux familles *MONTREDIV* et *MONTREOR*, défini avec une gamme de remise :

Jusqu'à	Remise
5000,00	10%
10000,00	12%
20000,00	15%
100000,00	15%+1000F

Après application des ces barèmes sur un bon de commande de vente créé le 05/04/2011, et contenant des lignes pour des articles associés aux familles *MONTREDIV* et *MONTREOR*. Deux lignes de remise auront automatiquement été ajoutées dans le document :

- 1ère remise de 5% correspondant au barème « Promotion Avril » et s'appliquant sur le total des lignes du document,
- 2ième remise de 10% correspondant au barème « Promotion montres » et s'appliquant sur le montant total des articles montre (*MOOR001* et *MOOR002*).

Code source

```
Option Strict Off

Imports Objets100cLib

Imports System


Module ProcessusAppliquerBareme

  'Objet base commerciale

  Dim oCial As BSCIALApplication100c

  'emplacement du fichier commercial

  Dim sPathGcm As String = "C:\Temp\Bijou.gcm"


  Sub Main()

    Try

      'Instanciation de l'objet base commercial

      oCial = New BSCIALApplication100c
```

```
    'Ouverture de la base
    If OpenBase(oCial, sPathGcm) Then
      'Teste l'existence du bon de commande BC00061
      If oCial.FactoryDocumentVente.ExistPiece( _
        DocumentType.DocumentTypeVenteCommande, "BC00197") Then
        'Récupération de l'objet document répondant aux critères
        Dim mDoc As IBODocumentVente3 = _
          oCial.FactoryDocumentVente.ReadPiece( _
            DocumentType.DocumentTypeVenteCommande, "BC00197")
        'Création d'un objet processus "Appliquer les barèmes"
        'en lui passant en paramètre le document précédemment récupéré
        Dim pBareme As IPMAppliquerBareme = _
          oCial.CreateProcess_AppliquerBareme(mDoc)
        'Si le processus peut être exécuté
        If pBareme.CanProcess Then
          'Validation du processus
          '=> Ajout des lignes de remise dans le document
          pBareme.Process()
        Else
          'Récupération de erreurs
          RecupError(CType(pBareme, IPMProcess))
        End If
      End If
    End If
  Catch ex As Exception
    Console.WriteLine("Erreur : " & ex.Message)
  Finally
    'Fermeture de la connexion
    CloseBase(oCial)
  End Try
End Sub


Public Sub RecupError(ByRef mP As IPMProcess)
```

```
    Try
      'Boucle sur les erreurs contenues dans la collection
      For i As Integer = 1 To mP.Errors.Count
        'Récupération des éléments erreurs
        Dim iFail As IFailInfo = mP.Errors.Item(i)
        'récupération du numéro d'erreur,
        'de l'indice et de la description de l'erreur
        Console.WriteLine("Code Erreur : " & _
          iFail.ErrorCode & " Indice : " & iFail.Indice & _
            " Description : " & iFail.Text)
      Next
    Catch ex As Exception
      Console.WriteLine("Erreur : " & ex.Message)
    End Try
  End Sub

  Public Function OpenBase(ByRef BaseCial As BSCIALApplication100c, _
    ByVal sGcm As String, Optional ByVal sUid As String = "<Administrateur>", _
    Optional ByVal sPwd As String = "") As Boolean
    Try
      'Affectation de l'emplacement du fichier commercial
      BaseCial.Name = sGcm
      'Affectation du code utilisateur
      BaseCial.Loggable.UserName = sUid
      'Affectation du mot de passe
      BaseCial.Loggable.UserPwd = sPwd
      'Ouverture de la base commerciale
      BaseCial.Open()
      Return True
    Catch ex As Exception
      Console.WriteLine("Erreur lors de l'ouverture du fichier gcm : " & ex.Message)
      Return False
    End Try
```

```
 End Function


 Private Function CloseBase(ByRef BaseCial As BSCIALApplication100c) As Boolean
  Try
   'Si la base est ouverte, alors fermeture de la base
   If BaseCial.IsOpen Then BaseCial.Close()
   Return True
  Catch ex As Exception
   Console.WriteLine("Erreur lors de la fermeture de la base : " & ex.Message)
   Return False
  End Try
 End Function
End Module
```

Processus de colisage

L'exemple suivant permet de décomposer une ligne de préparation de livraison, en générant plusieurs colis pour lesquels la quantité de chaque colis sera définie à 1.

La ligne de document à traiter appartient à la préparation de livraison ***PL00014***, pour l'article ***BAAR01*** avec une quantité colisée de 10.

Après exécution du processus de colisage, la ligne de document sera décomposée en 10 lignes portant chacune un numéro de colis différent.

Code source

```
Option Strict Off

Imports Objets100cLib

Imports System


Module ProcessusColisage
 'Objet base commerciale
 Dim oCial As BSCIALApplication100c
 'emplacement du fichier commercial
 Dim sPathGcm As String = "C:\Temp\Bijou.gcm"
```

```
Sub Main()
  Try
    'Instanciation de l'objet base commercial
    oCial = New BSCIALApplication100c
    'Ouverture de la base
    If OpenBase(oCial, sPathGcm) Then
      'Création du processus Coliser
      Dim pColiser As IPMColiser = oCial.CreateProcess_Coliser
      'Initialisation d'un compteur utilisé
      'pour l'attribution d'un numéro de colis
      Dim iCpt As Integer = 1
      'Récupération du document
      Dim pDoc As IBODocumentVente3 = _
       oCial.FactoryDocumentVente.ReadPiece( _
       DocumentType.DocumentTypeVentePrepaLivraison, "PL00014")
      'Parcours de chacune des lignes du document
      For Each pLig As IBODocumentVenteLigne3 In _
       pDoc.FactoryDocumentLigne.List
        'Affectation de la ligne au processus
        pColiser.LigneOrigine = pLig
        'Tant qu'il reste une quantité à coliser
        While pColiser.QteRestantAColiser > 0
          'Affectation d'un numéro de colis avec une quantité de 1
          CType(pColiser.UserColis.AddNew, IUserColis).Set( _
           "Colis" & iCpt, 1)
          'L'ajout d'un colis aurait également pu être
          'écrit de la manière suivante :
          'Dim iColis As IUserColis = _
          ' pColiser.UserColis.AddNew => Création d'un objet colis
          'iColis.NumColis = _
          ' pLig.Article.AR_Ref & iCpt => Affectation du numéro colis
          'iColis.Qte = 1 => Affectation de la quantité
          iCpt += 1
```

```
        End While
        'Si le processus peut être validé
        If pColiser.CanProcess Then
         'Validation du processus
         pColiser.Process()
        Else
         'Récupération des erreurs
         RecupError(CType(pColiser, IPMProcess))
        End If
       Next
      End If
    Catch ex As Exception
      Console.WriteLine("Erreur : " & ex.Message)
    Finally
      'Fermeture de la connexion
      CloseBase(oCial)
    End Try
  End Sub

  Public Sub RecupError(ByRef mP As IPMProcess)
    Try
      'Boucle sur les erreurs contenues dans la collection
      For i As Integer = 1 To mP.Errors.Count
        'Récupération des éléments erreurs
        Dim iFail As IFailInfo = mP.Errors.Item(i)
        'récupération du numéro d'erreur,
        'de l'indice et de la description de l'erreur
        Console.WriteLine("Code Erreur : " & _
          iFail.ErrorCode & " Indice : " & iFail.Indice & _
            " Description : " & iFail.Text)
      Next
    Catch ex As Exception
      Console.WriteLine("Erreur : " & ex.Message)
```

```
    End Try
  End Sub

  Public Function OpenBase(ByRef BaseCial As BSCIALApplication100c, _
    ByVal sGcm As String, Optional ByVal sUid As String = "<Administrateur>", _
    Optional ByVal sPwd As String = "") As Boolean
    Try
      'Affectation de l'emplacement du fichier commercial
      BaseCial.Name = sGcm
      'Affectation du code utilisateur
      BaseCial.Loggable.UserName = sUid
      'Affectation du mot de passe
      BaseCial.Loggable.UserPwd = sPwd
      'Ouverture de la base commerciale
      BaseCial.Open()
      Return True
    Catch ex As Exception
      Console.WriteLine("Erreur lors de l'ouverture du fichier gcm : " & ex.Message)
      Return False
    End Try
  End Function

  Private Function CloseBase(ByRef BaseCial As BSCIALApplication100c) As Boolean
    Try
      'Si la base est ouverte, alors fermeture de la base
      If BaseCial.IsOpen Then BaseCial.Close()
      Return True
    Catch ex As Exception
      Console.WriteLine("Erreur lors de la fermeture de la base : " & ex.Message)
      Return False
    End Try
  End Function
End Module
```

Processus de prélèvement Série/Lot

Les exemples suivants permettent de décomposer une ligne de préparation de livraison sur des articles suivis en Série/Lot, en générant plusieurs lignes pour lesquelles les numéros Série/Lot seront attribués.

Deux méthodes vont être employées. La première consistant à laisser le processus de prélèvement Série/Lot à affecter automatiquement les Série/Lot (équivalent à la fonction *Automatique* de la Gestion commerciale). La deuxième méthode, quant à elle, consistera à affecter manuellement les Série/Lot.

Ces deux exemples s'appuient sur une préparation de livraison (PL00015) contenant deux lignes sur des articles suivis en série et en lot (MOBWAC01 et LINGOR18). Pour l'article LINGOR18, des lots avec une quantité de 5 ont été entrées en stock. Détail des lignes du document :

Référence	P.U. HT	Quantité	Montant HT
MOBWAC01	980,00	3,00	2352,00
LINGOR18	80700,00	6,00	387360,00

Après exécution du processus de prélèvement, la ligne sur l'article suivi en série (MOBWAC01) de quantité de 3, sera décomposée en 3 lignes. La ligne sur l'article suivi en lot (LINGOR18) de quantité de 6, sera quant à elle décomposée en deux lignes : une ligne avec une quantité de 5 et une autre avec une quantité de 1.

Référence	Numéro Série/Lot	P.U. HT	Quantité	Montant HT
MOBWAC01	MOB0001	980,00	1,00	784,00
MOBWAC01	MOB0002	980,00	1,00	784,00
MOBWAC01	MOB0003	980,00	1,00	784,00
LINGOR18	LINGO0001	80700,00	5,00	322800,00
LINGOR18	LINGO0002	80700,00	1,00	64560,00

Code source traitement automatique des série/lot

```
Option Strict Off

Imports Objets100cLib

Imports System


Module ProcessusPreleverLotAutomatique
```

```
'Objet base commerciale
Dim oCial As BSCIALApplication100c
'emplacement du fichier commercial
Dim sPathGcm As String = "C:\Temp\Bijou.gcm"

Sub Main()
 Try
  'Instanciation de l'objet base commercial
  oCial = New BSCIALApplication100c
  'Ouverture de la base
  If OpenBase(oCial, sPathGcm) Then
   'Création de la variable processus Prelever
   Dim pPrelever As IPMPreleverLot = Nothing
   'Lecture de la préparation de livraison devant être traitée
   Dim pDoc As IBODocumentVente3 = _
    oCial.FactoryDocumentVente.ReadPiece( _
    DocumentType.DocumentTypeVentePrepaLivraison, "PL00015")
   'Parcours de chacune des lignes du document
   For Each pLig As IBODocumentVenteLigne3 In _
    pDoc.FactoryDocumentLigne.List
    'Si les lignes sont des lignes sur des articles
    'suivis en série/lot et que le numéro de série/lot
    'n'est pas déjà associé à la ligne
    If pLig.Article IsNot Nothing AndAlso _
     ((pLig.Article.AR_SuiviStock = _
     SuiviStockType.SuiviStockTypeLot _
     OrElse pLig.Article.AR_SuiviStock = _
     SuiviStockType.SuiviStockTypeSerie) _
     AndAlso pLig.LS_NoSerie = "") Then
     'Création du processus IPMPreleverLot
     pPrelever = oCial.CreateProcess_PreleverLot
     'Affectation de la ligne au processus
     pPrelever.LigneOrigine = pLig
```

```
            'Affichage des série/lot et quantités
            'proposés automatiquement
            For Each mUserLot As IUserLot In _
             pPrelever.UserLots
              Console.WriteLine(mUserLot.Lot.NoSerie & _
               " : " & mUserLot.QteToUse)
            Next
            'Si le processus peut être validé
            If pPrelever.CanProcess Then
              'Validation du processus
              pPrelever.Process()
            Else
              'Traitement des erreurs
              RecupError(CType(pPrelever, IPMProcess))
            End If
          End If
        Next
      End If
    Catch ex As Exception
      Console.WriteLine("Erreur : " & ex.Message)
    Finally
      'Fermeture de la connexion
      CloseBase(oCial)
    End Try
  End Sub

  Public Sub RecupError(ByRef mP As IPMProcess)
    Try
      'Boucle sur les erreurs contenues dans la collection
      For i As Integer = 1 To mP.Errors.Count
        'Récupération des éléments erreurs
        Dim iFail As IFailInfo = mP.Errors.Item(i)
        'récupération du numéro d'erreur,
```

```
        'de l'indice et de la description de l'erreur
        Console.WriteLine("Code Erreur : " & _
          iFail.ErrorCode & " Indice : " & iFail.Indice & _
            " Description : " & iFail.Text)
      Next
    Catch ex As Exception
      Console.WriteLine("Erreur : " & ex.Message)
    End Try
  End Sub

  Public Function OpenBase(ByRef BaseCial As BSCIALApplication100c, _
      ByVal sGcm As String, Optional ByVal sUid As String = "<Administrateur>", _
      Optional ByVal sPwd As String = "") As Boolean
    Try
      'Affectation de l'emplacement du fichier commercial
      BaseCial.Name = sGcm
      'Affectation du code utilisateur
      BaseCial.Loggable.UserName = sUid
      'Affectation du mot de passe
      BaseCial.Loggable.UserPwd = sPwd
      'Ouverture de la base commerciale
      BaseCial.Open()
      Return True
    Catch ex As Exception
      Console.WriteLine("Erreur lors de l'ouverture du fichier gcm : " & ex.Message)
      Return False
    End Try
  End Function

  Private Function CloseBase(ByRef BaseCial As BSCIALApplication100c) As Boolean
    Try
      'Si la base est ouverte, alors fermeture de la base
      If BaseCial.IsOpen Then BaseCial.Close()
```

```
      Return True
    Catch ex As Exception
      Console.WriteLine("Erreur lors de la fermeture de la base : " & ex.Message)
      Return False
    End Try
  End Function
End Module
```

Code source traitement manuel des série/lot

```
Option Strict Off
Imports Objets100cLib
Imports System

Module ProcessusPreleverLotManuel
  'Objet base commerciale
  Dim oCial As BSCIALApplication100c
  'emplacement du fichier commercial
  Dim sPathGcm As String = "C:\Temp\Bijou.gcm"

  Sub Main()
    Try
      'Instanciation de l'objet base commercial
      oCial = New BSCIALApplication100c
      'Ouverture de la base
      If OpenBase(oCial, sPathGcm) Then
        Dim pPrelever As IPMPreleverLot = Nothing
        'Lecture de la préparation de livraison devant être découpée
        Dim pDoc As IBODocumentVente3 = _
          oCial.FactoryDocumentVente.ReadPiece( _
          DocumentType.DocumentTypeVentePrepaLivraison, "PL00015")
        'Parcours de chacune des lignes du document
        For Each pLig As IBODocumentVenteLigne3 In _
```

```
pDoc.FactoryDocumentLigne.List
 'Si les lignes sont des lignes sur des articles
 'suivis en série/lot et que le numéro de série/lot
 'n'est pas déjà associé à la ligne
 If pLig.Article IsNot Nothing AndAlso _
  ((pLig.Article.AR_SuiviStock = _
  SuiviStockType.SuiviStockTypeLot _
  OrElse pLig.Article.AR_SuiviStock = _
  SuiviStockType.SuiviStockTypeSerie) _
  AndAlso pLig.LS_NoSerie = "") Then
   'Création du processus IPMPreleverLot
   pPrelever = oCial.CreateProcess_PreleverLot
   'Affectation de la ligne au processus
   pPrelever.LigneOrigine = pLig
   'Suppression des série/lot proposés par défaut
   pPrelever.UserLots.RemoveAll()
   'Création d'un objet dépôt
   Dim pDepot As IBODepot3 = pLig.Depot
   'Création d'un objet article dépôt
   Dim pArtDepot As IBOArticleDepot3 = _
    pLig.Article.FactoryArticleDepot.ReadDepot(pDepot)
   'Parcours de la collecttion des série/lot non
   'épuisés pour un article
   For Each pArtDepotLot As IBOArticleDepotLot In _
    pArtDepot.FactoryArticleDepotLot.QueryNonEpuise
     'Sortie de la boucle s'il n'y a plus de quantité à répartir
     If pPrelever.QteRestantARepartir = 0 Then Exit For
     'Récupération de la quantité disponible du série/lot
     Dim dQte As Double = pArtDepotLot.StockATerme
     'S'il y a une quantité disponible sur le série/lot
     If dQte > 0 Then
       'Création d'un série/lot pour le processus
       Dim mUserLot As IUserLot = _
```

```
            pPrelever.UserLots.AddNew
           'Si la quantité du série/lot est inférieure
           'à la quantité restante
           If pPrelever.QteRestantARepartir >= dQte Then
            'Affectation de l'intégralité de
            'la quantité du série/lot
            mUserLot.Set(pArtDepotLot, dQte, _
             pArtDepotLot.Complement)
           Else
            'Affectation de la quantité ajustée
            'à la quantité restante à répartir
            mUserLot.Set(pArtDepotLot, _
             pPrelever.QteRestantARepartir, _
             pArtDepotLot.Complement)
           End If
          End If
         Next
         'Affichage des série/lot et quantités
         'viennent d'être affectées
         For Each mUserLot As IUserLot In _
          pPrelever.UserLots
          Console.WriteLine(mUserLot.Lot.NoSerie & _
           " : " & mUserLot.QteToUse)
         Next
         'Si le processus peut être validé
         If pPrelever.CanProcess Then
          'Validation du processus
          pPrelever.Process()
         Else
          'Traitement des erreurs
          RecupError(CType(pPrelever, IPMProcess))
         End If
        End If
```

```
      Next
    End If
  Catch ex As Exception
    Console.WriteLine("Erreur : " & ex.Message)
  Finally
    'Fermeture de la connexion
    CloseBase(oCial)
  End Try
End Sub

Public Sub RecupError(ByRef mP As IPMProcess)
  Try
    'Boucle sur les erreurs contenues dans la collection
    For i As Integer = 1 To mP.Errors.Count
      'Récupération des éléments erreurs
      Dim iFail As IFailInfo = mP.Errors.Item(i)
      'récupération du numéro d'erreur,
      'de l'indice et de la description de l'erreur
      Console.WriteLine("Code Erreur : " & _
        iFail.ErrorCode & " Indice : " & iFail.Indice & _
          " Description : " & iFail.Text)
    Next
  Catch ex As Exception
    Console.WriteLine("Erreur : " & ex.Message)
  End Try
End Sub

Public Function OpenBase(ByRef BaseCial As BSCIALApplication100c, _
    ByVal sGcm As String, Optional ByVal sUid As String = "<Administrateur>", _
    Optional ByVal sPwd As String = "") As Boolean
  Try
    'Affectation de l'emplacement du fichier commercial
    BaseCial.Name = sGcm
```

```
        'Affectation du code utilisateur
        BaseCial.Loggable.UserName = sUid
        'Affectation du mot de passe
        BaseCial.Loggable.UserPwd = sPwd
        'Ouverture de la base commerciale
        BaseCial.Open()
        Return True
      Catch ex As Exception
        Console.WriteLine("Erreur lors de l'ouverture du fichier gcm : " & ex.Message)
        Return False
      End Try
    End Function

    Private Function CloseBase(ByRef BaseCial As BSCIALApplication100c) As Boolean
      Try
        'Si la base est ouverte, alors fermeture de la base
        If BaseCial.IsOpen Then BaseCial.Close()
        Return True
      Catch ex As Exception
        Console.WriteLine("Erreur lors de la fermeture de la base : " & ex.Message)
        Return False
      End Try
    End Function
End Module
```

Processus Transfert d’article

Les trois exemples présentés ci-après permettent la mise en œuvre de différents types de transfert :

- Transfert d’un dépôt vers un autre pour un article suivi au CMUP en utilisant un mouvement de transfert existant, et en laissant les emplacements affectés automatiquement par le processus de transfert,
- Transfert d’un emplacement vers un autre pour un article suivi au CMUP en créant un nouveau mouvement de transfert (utilisation du processus de création de document ***IPMDocument***), et en affectant manuellement les emplacements,

- Transfert d'un emplacement vers un autre pour un article suivi par lot en créant un nouveau mouvement de transfert (utilisation du processus de création de document ***IPMDocument***), et en affectant l'emplacement sur le dépôt de destination.

Transfert d'un dépôt vers un autre pour un article suivi au CMUP en utilisant un document existant

L'exemple suivant permet de créer un transfert d'article dans un document existant, pour transférer 10 quantités de l'article ***BAAR01*** suivi au CMUP, du dépôt ***Bijou SA*** vers le dépôt ***Annexe Bijou SA***. Les emplacements seront affectés automatiquement par le processus en utilisant la même règle que celle utilisée par la Gestion commerciale (utilisation de l'emplacement par défaut du dépôt).

Code source

```
Option Strict Off

Imports Objets100cLib

Imports System


Module ProcessusTransfererDocumentExistant
 'Objet base commerciale
 Dim oCial As BSCIALApplication100c
 'emplacement du fichier commercial
 Dim sPathGcm As String = "C:\Temp\Bijou.gcm"


 Sub Main()
  Try
   'Instanciation de l'objet base commercial
   oCial = New BSCIALApplication100c
   'Ouverture de la base
   If OpenBase(oCial, sPathGcm) Then
    'Récupération du mouvement de transfert MT00021
    Dim mDoc As IBODocumentStock3 = _
       oCial.FactoryDocumentStock.ReadPiece( _
       DocumentType.DocumentTypeStockVirement, "MT00021")
    'Création du processus transférer
    Dim pTransfert As IPMDocTransferer = _
       oCial.CreateProcess_DocTransferer
    'Affectation du document au processus
```

```
      pTransfert.Document = mDoc
      'Affectation de l'article au processus
      pTransfert.SetDefaultArticle( _
        oCial.FactoryArticle.ReadReference("BAAR01"), 10)
      'Si le processus peut être validé
      If pTransfert.CanProcess Then
        'Validation du processus
        pTransfert.Process()
        '=> Ajout de la ligne de sortie et de la ligne d'entrée
        '   pour l'article BAAR01 dans le document MT00021
      Else
        'Si CanProcess() a échoué, traitement
        'des erreurs
        RecupError(CType(pTransfert, IPMProcess))
      End If
    End If
  Catch ex As Exception
    Console.WriteLine("Erreur : " & ex.Message)
  Finally
    'Fermeture de la connexion
    CloseBase(oCial)
  End Try
End Sub

Public Sub RecupError(ByRef mP As IPMProcess)
  Try
    'Boucle sur les erreurs contenues dans la collection
    For i As Integer = 1 To mP.Errors.Count
      'Récupération des éléments erreurs
      Dim iFail As IFailInfo = mP.Errors.Item(i)
      'récupération du numéro d'erreur,
      'de l'indice et de la description de l'erreur
      Console.WriteLine("Code Erreur : " & _
```

```
            iFail.ErrorCode & " Indice : " & iFail.Indice & _
            " Description : " & iFail.Text)
        Next
    Catch ex As Exception
        Console.WriteLine("Erreur : " & ex.Message)
    End Try
End Sub

Public Function OpenBase(ByRef BaseCial As BSCIALApplication100c, _
    ByVal sGcm As String, Optional ByVal sUid As String = "<Administrateur>", _
    Optional ByVal sPwd As String = "") As Boolean
    Try
        'Affectation de l'emplacement du fichier commercial
        BaseCial.Name = sGcm
        'Affectation du code utilisateur
        BaseCial.Loggable.UserName = sUid
        'Affectation du mot de passe
        BaseCial.Loggable.UserPwd = sPwd
        'Ouverture de la base commerciale
        BaseCial.Open()
        Return True
    Catch ex As Exception
        Console.WriteLine("Erreur lors de l'ouverture du fichier gcm : " & ex.Message)
        Return False
    End Try
End Function

Private Function CloseBase(ByRef BaseCial As BSCIALApplication100c) As Boolean
    Try
        'Si la base est ouverte, alors fermeture de la base
        If BaseCial.IsOpen Then BaseCial.Close()
        Return True
    Catch ex As Exception
```

```
    Console.WriteLine("Erreur lors de la fermeture de la base : " & ex.Message)
    Return False
   End Try
  End Function
End Module
```

Transfert d'un emplacement vers un autre pour un article suivi au CMUP avec creation d'un nouveau document

L'exemple suivant permet de créer un transfert d'article dans un nouveau document, pour transférer 10 quantités de l'article ***BAAR01*** suivi au CMUP, de l'emplacement ***A1T2N1P2*** vers l'emplacement ***A3T1N2P2*** pour le dépôt ***Bijou SA***. Pour ce faire, cet exemple mettra en œuvre deux processus :

- ***IPMDocument*** pour la création du document mouvement de transfert,
- ***IPMDocTransferer*** pour le transfert de l'article.

Code source

```
Option Strict Off
Imports Objets100cLib
Imports System

Module ProcessusTransfererNouveauDocument
 'Objet base commerciale
 Dim oCial As BSCIALApplication100c
 'emplacement du fichier commercial
 Dim sPathGcm As String = "C:\Temp\Bijou.gcm"

 Sub Main()
  Try
   'Instanciation de l'objet base commercial
   oCial = New BSCIALApplication100c
   'Ouverture de la base
   If OpenBase(oCial, sPathGcm) Then
    'Récupération du dépôt Bijou SA
    Dim mDepot As IBODepot3 = _
      oCial.FactoryDepot.ReadIntitule("Bijou SA")
```

```
'Création d'un processus document
'de type Mouvement de transfert
Dim pDoc As IPMDocument = _
    oCial.CreateProcess_Document _
     (DocumentType.DocumentTypeStockVirement)
'Initialisation du document
Dim mDoc As IBODocumentStock3 = pDoc.Document
'Affectation du dépôt d'origine
mDoc.DepotOrigine = mDepot
'Affectation du dépôt de destination
mDoc.DepotDestination = mDepot
'Initialisation des valeurs par défaut
mDoc.SetDefaultDO_Piece()
'Ecriture mémoire du document
mDoc.WriteDefault()
'Création du processus transférer
Dim pTransfert As IPMDocTransferer = _
    oCial.CreateProcess_DocTransferer
'Affectation du document mémoire au processus
pTransfert.Document = mDoc
'Affectation de l'article au processus
pTransfert.SetDefaultArticle _
  (oCial.FactoryArticle.ReadReference("BAAR01"), 10)
'Affectation de l'emplacement pour la ligne de sortie
pTransfert.DepotEmplOrigine = _
  mDepot.FactoryDepotEmplacement.ReadCode("A1T2N1P2")
'Affectation de l'emplacement pour la ligne d'entrée
pTransfert.DepotEmplDest = _
  mDepot.FactoryDepotEmplacement.ReadCode("A3T1N2P2")
'Si le processus de transfert peut être validé
If pTransfert.CanProcess Then
  'Validation du processus
  pTransfert.Process()
```

```
        '=> Ajout de la ligne de sortie et de la ligne d'entrée
        '   pour l'article BAAR01 dans le document mémoire

        'Si le processus de création de document peut être validé
        If pDoc.CanProcess Then
          'Validation du processus
          pDoc.Process()
          'Ecriture du document dans la base de données
        Else
          'Si CanProcess() sur le processus document
          'a échoué, traitement des erreurs
          RecupError(CType(pDoc, IPMProcess))
        End If
      Else
        'Si CanProcess() sur le processus de transfert
        'a échoué, traitement des erreurs
        RecupError(CType(pTransfert, IPMProcess))
      End If
    End If
  Catch ex As Exception
    Console.WriteLine("Erreur : " & ex.Message)
  Finally
    'Fermeture de la connexion
    CloseBase(oCial)
  End Try
End Sub

Public Sub RecupError(ByRef mP As IPMProcess)
  Try
    'Boucle sur les erreurs contenues dans la collection
    For i As Integer = 1 To mP.Errors.Count
      'Récupération des éléments erreurs
      Dim iFail As IFailInfo = mP.Errors.Item(i)
```

```
        'récupération du numéro d'erreur,
        'de l'indice et de la description de l'erreur
        Console.WriteLine("Code Erreur : " & _
          iFail.ErrorCode & " Indice : " & iFail.Indice & _
            " Description : " & iFail.Text)
      Next
    Catch ex As Exception
      Console.WriteLine("Erreur : " & ex.Message)
    End Try
  End Sub

  Public Function OpenBase(ByRef BaseCial As BSCIALApplication100c, _
      ByVal sGcm As String, Optional ByVal sUid As String = "<Administrateur>", _
      Optional ByVal sPwd As String = "") As Boolean
    Try
      'Affectation de l'emplacement du fichier commercial
      BaseCial.Name = sGcm
      'Affectation du code utilisateur
      BaseCial.Loggable.UserName = sUid
      'Affectation du mot de passe
      BaseCial.Loggable.UserPwd = sPwd
      'Ouverture de la base commerciale
      BaseCial.Open()
      Return True
    Catch ex As Exception
      Console.WriteLine("Erreur lors de l'ouverture du fichier gcm : " & ex.Message)
      Return False
    End Try
  End Function

  Private Function CloseBase(ByRef BaseCial As BSCIALApplication100c) As Boolean
    Try
      'Si la base est ouverte, alors fermeture de la base
```

```
    If BaseCial.IsOpen Then BaseCial.Close()
    Return True
   Catch ex As Exception
    Console.WriteLine("Erreur lors de la fermeture de la base : " & ex.Message)
    Return False
   End Try
  End Function
End Module
```

Transfert d'un emplacement vers un autre pour un article géré par lot avec création d'un nouveau document

L'exemple suivant permet de créer un transfert d'article dans un nouveau document, pour transférer l'article ***LINGOR18*** suivi par lot, de l'emplacement d'origine du lot vers l'emplacement ***A3T1N2P2*** pour le dépôt ***Bijou SA***. Pour ce faire, comme dans l'exemple précédent, deux processus seront mis en œuvre : ***IPMDocument*** pour la création de document et ***IPMDocTransferer*** pour le transfert d'article.

Code source

```
Option Strict Off
Imports Objets100cLib
Imports System

Module ProcessusTransfererNouveauDocumentAvecLot
 'Objet base commerciale
 Dim oCial As BSCIALApplication100c
 'emplacement du fichier commercial
 Dim sPathGcm As String = "C:\Temp\Bijou.gcm"

 Sub Main()
  Try
   'Instanciation de l'objet base commercial
   oCial = New BSCIALApplication100c
   'Ouverture de la base
   If OpenBase(oCial, sPathGcm) Then
    'Récupération du dépôt Bijou SA
```

```
Dim mDepot As IBODepot3 = _
    oCial.FactoryDepot.ReadIntitule("Bijou SA")
'Création d'un processus document
'de type Mouvement de transfert
Dim pDoc As IPMDocument = _
    oCial.CreateProcess_Document _
     (DocumentType.DocumentTypeStockVirement)
'Initialisation du document
Dim mDoc As IBODocumentStock3 = pDoc.Document
'Affectation du dépôt d'origine
mDoc.DepotOrigine = mDepot
'Affectation du dépôt de destination
mDoc.DepotDestination = mDepot
'Initialisation des valeurs par défaut
mDoc.SetDefaultDO_Piece()
'Ecriture mémoire du document
mDoc.WriteDefault()
'Création du processus transférer
Dim pTransfert As IPMDocTransferer = _
    oCial.CreateProcess_DocTransferer
'Affectation du document mémoire au processus
pTransfert.Document = mDoc
'Initialisation d'un objet article pour LINGOR18
Dim mArt As IBOArticle3 = _
    oCial.FactoryArticle.ReadReference("LINGOR18")
'Affectation de l'article au processus
pTransfert.SetDefaultArticle(mArt, 2)
'Suppression des lots affectés automatiquement
pTransfert.UserLotsToUse.RemoveAll()
'Création d'un objet IUserLot
Dim mUserLot As IUserLot = _
pTransfert.UserLotsToUse.AddNew()
'Intitialisation du lot article
```

```
Dim mArtDepotLot As IBOArticleDepotLot = _
mArt.FactoryArticleDepot.ReadDepot(mDepot). _
FactoryArticleDepotLot.ReadNoSerie("LINGOR001")
'Affectation du lot au processus
mUserLot.Lot = mArtDepotLot
'Affectation de la quantité du lot
mUserLot.QteToUse = 2
'Affectation de l'emplacement pour la ligne d'entrée
pTransfert.DepotEmplDest = _
  mDepot.FactoryDepotEmplacement.ReadCode("A3T1N2P2")
'=> L'emplacement d'origine ne doit pas être
'renseigné car c'est le lot ajouté au processus
'qui détermine l'emplacement de sortie.
'Si toute la quantité nécessaire pour
'le transfert a été fournie
If pTransfert.UserLotsQteRestantAFournir = 0 Then
  'Si le processus de transfert peut être validé
  If pTransfert.CanProcess Then
    'Validation du processus
    pTransfert.Process()
    '=> Ajout de la ligne de sortie et de la ligne d'entrée
    '   pour l'article LINGOR18 dans le document mémoire

    'Si le processus de création de document peut être validé
    If pDoc.CanProcess Then
      'Validation du processus
      pDoc.Process()
      'Ecriture du document dans la base de données
    Else
      'Si CanProcess() sur le processus document
      'a échoué, traitement des erreurs
      RecupError(CType(pDoc, IPMProcess))
    End If
```

```
        Else
          'Si CanProcess() sur le processus de transfert
          'a échoué, traitement des erreurs
          RecupError(CType(pTransfert, IPMProcess))
        End If
      End If

    End If
  Catch ex As Exception
    Console.WriteLine("Erreur : " & ex.Message)
  Finally
    'Fermeture de la connexion
    CloseBase(oCial)
  End Try
End Sub

Public Sub RecupError(ByRef mP As IPMProcess)
  Try
    'Boucle sur les erreurs contenues dans la collection
    For i As Integer = 1 To mP.Errors.Count
      'Récupération des éléments erreurs
      Dim iFail As IFailInfo = mP.Errors.Item(i)
      'récupération du numéro d'erreur,
      'de l'indice et de la description de l'erreur
      Console.WriteLine("Code Erreur : " & _
        iFail.ErrorCode & " Indice : " & iFail.Indice & _
          " Description : " & iFail.Text)
    Next
  Catch ex As Exception
    Console.WriteLine("Erreur : " & ex.Message)
  End Try
End Sub
```

```
 Public Function OpenBase(ByRef BaseCial As BSCIALApplication100c, _
    ByVal sGcm As String, Optional ByVal sUid As String = "<Administrateur>", _
    Optional ByVal sPwd As String = "") As Boolean
  Try
    'Affectation de l'emplacement du fichier commercial
    BaseCial.Name = sGcm
    'Affectation du code utilisateur
    BaseCial.Loggable.UserName = sUid
    'Affectation du mot de passe
    BaseCial.Loggable.UserPwd = sPwd
    'Ouverture de la base commerciale
    BaseCial.Open()
    Return True
  Catch ex As Exception
    Console.WriteLine("Erreur lors de l'ouverture du fichier gcm : " & ex.Message)
    Return False
  End Try
 End Function

 Private Function CloseBase(ByRef BaseCial As BSCIALApplication100c) As Boolean
  Try
    'Si la base est ouverte, alors fermeture de la base
    If BaseCial.IsOpen Then BaseCial.Close()
    Return True
  Catch ex As Exception
    Console.WriteLine("Erreur lors de la fermeture de la base : " & ex.Message)
    Return False
  End Try
 End Function
End Module
```

Processus de transformation de documents

Les exemples présentés ci-après permettent la mise en œuvre de différents processus de transformation :

- Transformation de documents de vente :
  - ◊ Commander : Transformation d'une ligne d'un devis dans un nouveau bon de commande,
  - ◊ Commander : Transformation d'un devis dans un nouveau bon de commande,
  - ◊ Livrer : Transformation d'un devis et un bon de commande dans deux bons de livraison existant,
  - ◊ Livrer : Transformation d'un devis dans un nouveau bon de livraison avec affectation manuelle des numéros série/lot.
- Transformation de documents d'achat :
  - ◊ Commander : Transformation de deux préparations de commande dans un nouveau bon de commande,
  - ◊ Receptionner : Transformation d'un bon de commande dans un nouveau bon de livraison avec affectation manuelle des numéros série/lot.

Transformation d'une ligne d'un document de vente dans un nouveau Bon de commande

Cet exemple permet de transformer la première ligne de document du devis ***DE00036*** dans un bon de commande. Le document de destination n'étant pas précisé, le processus de transformation le créera automatiquement en s'appuyant sur les informations d'entête du devis d'origine.

Code source

```
Option Strict Off

Imports Objets100cLib

Imports System


Module ProcessusVenteCommander_1LigneDeToNewBc

  'Objet base commerciale

  Dim oCial As BSCIALApplication100c

  'emplacement du fichier commercial

  Dim sPathGcm As String = "C:\Temp\Bijou.gcm"


  Sub Main()
```

```
Try
 'Instanciation de l'objet base commercial
 oCial = New BSCIALApplication100c
 'Ouverture de la base
 If OpenBase(oCial, sPathGcm) Then
  'Création du processus Commander
  Dim pTransfo As IPMDocTransformer = _
  oCial.Transformation.Vente.CreateProcess_Commander
  'Si le devis DE00036 existe
  If oCial.FactoryDocumentVente.ExistPiece( _
   DocumentType.DocumentTypeVenteDevis, "DE00036") Then
   'Sélection du devis DE00036
   Dim pDoc As IBODocumentVente3 = _
    oCial.FactoryDocumentVente.ReadPiece( _
    DocumentType.DocumentTypeVenteDevis, "DE00036")
   'Si le document contient au moins une ligne
   If pDoc.FactoryDocumentLigne.List.Count > 0 Then
    'Sélection de la première ligne du devis
    Dim pLig As IBODocumentVenteLigne3 = _
     pDoc.FactoryDocumentLigne.List(1)
    'Ajout de la ligne au process
    pTransfo.AddDocumentLigne(pLig)
    'Test pour savoir si le processus peut être validé
    If pTransfo.CanProcess Then
     'Validation du processus
     pTransfo.Process()
     'Affichage du numéro de pièce du document créé
     'par le processus de transformation
     Console.WriteLine("Ligne transformée dans le document " & _
      CType(pTransfo.ListDocumentsResult(1),  _
      IBODocumentVente3).DO_Piece)
    Else
     'Traitement de récupération des erreurs
```

```
            RecupError(CType(pTransfo, IPMProcess))
          End If
        End If
      End If
    End If
  Catch ex As Exception
    Console.WriteLine("Erreur : " & ex.Message)
  Finally
    'Fermeture de la connexion
    CloseBase(oCial)
  End Try
End Sub

Public Sub RecupError(ByRef mP As IPMProcess)
  Try
    'Boucle sur les erreurs contenues dans la collection
    For i As Integer = 1 To mP.Errors.Count
      'Récupération des éléments erreurs
      Dim iFail As IFailInfo = mP.Errors.Item(i)
      'récupération du numéro d'erreur,
      'de l'indice et de la description de l'erreur
      Console.WriteLine("Code Erreur : " & _
        iFail.ErrorCode & " Indice : " & iFail.Indice & _
          " Description : " & iFail.Text)
    Next
  Catch ex As Exception
    Console.WriteLine("Erreur : " & ex.Message)
  End Try
End Sub

Public Function OpenBase(ByRef BaseCial As BSCIALApplication100c, _
    ByVal sGcm As String, Optional ByVal sUid As String = "<Administrateur>", _
    Optional ByVal sPwd As String = "") As Boolean
```

```
    Try
      'Affectation de l'emplacement du fichier commercial
      BaseCial.Name = sGcm
      'Affectation du code utilisateur
      BaseCial.Loggable.UserName = sUid
      'Affectation du mot de passe
      BaseCial.Loggable.UserPwd = sPwd
      'Ouverture de la base commerciale
      BaseCial.Open()
      Return True
    Catch ex As Exception
      Console.WriteLine("Erreur lors de l'ouverture du fichier gcm : " & ex.Message)
      Return False
    End Try
  End Function

  Private Function CloseBase(ByRef BaseCial As BSCIALApplication100c) As Boolean
    Try
      'Si la base est ouverte, alors fermeture de la base
      If BaseCial.IsOpen Then BaseCial.Close()
      Return True
    Catch ex As Exception
      Console.WriteLine("Erreur lors de la fermeture de la base : " & ex.Message)
      Return False
    End Try
  End Function
End Module
```

Transformation d’un document de vente dans un nouveau Bon de commande

L’exemple suivant permet de transformer l’intégralité du devis ***DE00037*** dans un nouveau bon de commande. Le document de destination n’étant pas précisé, le processus de transformation le créera automatiquement en s’appuyant sur les informations d’entête du devis d’origine. Après transformation, le devis d’origine n’existera plus car l’intégralité des lignes aura été transformée dans le bon de commande.

Code source

```
Option Strict Off

Imports Objets100cLib

Imports System


Module ProcessusVenteCommander_1DeToNewBc
 'Objet base commerciale
 Dim oCial As BSCIALApplication100c
 'emplacement du fichier commercial
 Dim sPathGcm As String = "C:\Temp\Bijou.gcm"


 Sub Main()
  Try
   'Instanciation de l'objet base commercial
   oCial = New BSCIALApplication100c
   'Ouverture de la base
   If OpenBase(oCial, sPathGcm) Then
    'Création du processus Commander
    Dim pTransfo As IPMDocTransformer = _
     oCial.Transformation.Vente.CreateProcess_Commander
    'Si le devis DE00037 existe
    If oCial.FactoryDocumentVente.ExistPiece( _
     DocumentType.DocumentTypeVenteDevis, "DE00037") Then
     'Sélection du devis DE00037
     Dim pDoc As IBODocumentVente3 = _
      oCial.FactoryDocumentVente.ReadPiece( _
       DocumentType.DocumentTypeVenteDevis, "DE00037")
     'Ajout du document au process
     pTransfo.AddDocument(pDoc)
     'Test pour savoir si le processus peut être validé
     If pTransfo.CanProcess Then
      'Validation du processus
      pTransfo.Process()
```

```
            'Affichage du numéro de pièce du document créé
            'par le processus de transformation
            Console.WriteLine("Document transformé dans le document " & _
             CType(pTransfo.ListDocumentsResult(1),  _
              IBODocumentVente3).DO_Piece)
          Else
            'Traitement de récupération des erreurs
            RecupError(CType(pTransfo, IPMProcess))
          End If
        End If
      End If
    Catch ex As Exception
      Console.WriteLine("Erreur : " & ex.Message)
    Finally
      'Fermeture de la connexion
      CloseBase(oCial)
    End Try
  End Sub

  Public Sub RecupError(ByRef mP As IPMProcess)
    Try
      'Boucle sur les erreurs contenues dans la collection
      For i As Integer = 1 To mP.Errors.Count
        'Récupération des éléments erreurs
        Dim iFail As IFailInfo = mP.Errors.Item(i)
        'récupération du numéro d'erreur,
        'de l'indice et de la description de l'erreur
        Console.WriteLine("Code Erreur : " & _
          iFail.ErrorCode & " Indice : " & iFail.Indice & _
            " Description : " & iFail.Text)
      Next
    Catch ex As Exception
      Console.WriteLine("Erreur : " & ex.Message)
```

```
    End Try
  End Sub

  Public Function OpenBase(ByRef BaseCial As BSCIALApplication100c, _
    ByVal sGcm As String, Optional ByVal sUid As String = "<Administrateur>", _
    Optional ByVal sPwd As String = "") As Boolean
    Try
      'Affectation de l'emplacement du fichier commercial
      BaseCial.Name = sGcm
      'Affectation du code utilisateur
      BaseCial.Loggable.UserName = sUid
      'Affectation du mot de passe
      BaseCial.Loggable.UserPwd = sPwd
      'Ouverture de la base commerciale
      BaseCial.Open()
      Return True
    Catch ex As Exception
      Console.WriteLine("Erreur lors de l'ouverture du fichier gcm : " & ex.Message)
      Return False
    End Try
  End Function

  Private Function CloseBase(ByRef BaseCial As BSCIALApplication100c) As Boolean
    Try
      'Si la base est ouverte, alors fermeture de la base
      If BaseCial.IsOpen Then BaseCial.Close()
      Return True
    Catch ex As Exception
      Console.WriteLine("Erreur lors de la fermeture de la base : " & ex.Message)
      Return False
    End Try
  End Function
End Module
```

Transformation de deux documents de vente dans deux bons de livraison existants

Cet exemple permet de transformer l'intégralité du devis ***DE00038*** et du bon de commande ***BC00201*** dans deux bons de livraisons existants (***BL00015*** et ***BL00016***). Pour que le processus de transformation utilise ces documents de destination, il est nécessaire que les informations d'entête des bons de livraison correspondent avec les documents à transformer (même tiers, devise, souche…). Après transformation, le devis et bon de commande d'origine n'existeront plus car l'intégralité des lignes aura été transformée dans les deux bons de commande.

Code source

```
Option Strict Off

Imports Objets100cLib

Imports System


Module ProcessusVenteLivrer_1De1BcTo2Bl
 'Objet base commerciale
 Dim oCial As BSCIALApplication100c
 'emplacement du fichier commercial
 Dim sPathGcm As String = "C:\Temp\Bijou.gcm"


 Sub Main()
  Try
   'Instanciation de l'objet base commercial
   oCial = New BSCIALApplication100c
   'Ouverture de la base
   If OpenBase(oCial, sPathGcm) Then
    'Création du processus Livrer
    Dim pTransfo As IPMDocTransformer = _
     oCial.Transformation.Vente.CreateProcess_Livrer
    'Si les documents à utiliser dans le processus existent
    If oCial.FactoryDocumentVente.ExistPiece( _
     DocumentType.DocumentTypeVenteDevis, "DE00038") AndAlso _
      oCial.FactoryDocumentVente.ExistPiece( _
       DocumentType.DocumentTypeVenteCommande, "BC00201") AndAlso _
        oCial.FactoryDocumentVente.ExistPiece( _
         DocumentType.DocumentTypeVenteLivraison, "BL00015") AndAlso _
```

```
            oCial.FactoryDocumentVente.ExistPiece( _
             DocumentType.DocumentTypeVenteLivraison, "BL00016") Then
        'Sélection du devis DE00038 du tiers CARAT
        Dim pDoc As IBODocumentVente3 = _
         oCial.FactoryDocumentVente.ReadPiece( _
          DocumentType.DocumentTypeVenteDevis, "DE00038")
        'Ajout du document au processus
        pTransfo.AddDocument(pDoc)
        'Sélection du bon de commande BC00201 du tiers CISEL
        pDoc = oCial.FactoryDocumentVente.ReadPiece( _
         DocumentType.DocumentTypeVenteCommande, "BC00201")
        'Ajout du document au processus
        pTransfo.AddDocument(pDoc)
        'Sélection du bon de commande BL00015 du tiers CARAT
        pDoc = oCial.FactoryDocumentVente.ReadPiece( _
         DocumentType.DocumentTypeVenteLivraison, "BL00015")
        'Ajout du document en tant que document de destination
        pTransfo.AddDocumentDestination(pDoc)
        'Sélection du bon de commande BL00016 du tiers CISEL
        pDoc = oCial.FactoryDocumentVente.ReadPiece( _
         DocumentType.DocumentTypeVenteLivraison, "BL00016")
        'Ajout du document en tant que document de destination
        pTransfo.AddDocumentDestination(pDoc)
        'Test pour savoir si le processus peut être validé
        If pTransfo.CanProcess Then
          'Validation du processus
          pTransfo.Process()
        Else
          'Traitement de récupération des erreurs
          RecupError(CType(pTransfo, IPMProcess))
        End If
      End If
    End If
```

```
  Catch ex As Exception
    Console.WriteLine("Erreur : " & ex.Message)
  Finally
    'Fermeture de la connexion
    CloseBase(oCial)
  End Try
End Sub

Public Sub RecupError(ByRef mP As IPMProcess)
  Try
    'Boucle sur les erreurs contenues dans la collection
    For i As Integer = 1 To mP.Errors.Count
      'Récupération des éléments erreurs
      Dim iFail As IFailInfo = mP.Errors.Item(i)
      'récupération du numéro d'erreur,
      'de l'indice et de la description de l'erreur
      Console.WriteLine("Code Erreur : " & _
        iFail.ErrorCode & " Indice : " & iFail.Indice & _
          " Description : " & iFail.Text)
    Next
  Catch ex As Exception
    Console.WriteLine("Erreur : " & ex.Message)
  End Try
End Sub

Public Function OpenBase(ByRef BaseCial As BSCIALApplication100c, _
    ByVal sGcm As String, Optional ByVal sUid As String = "<Administrateur>", _
    Optional ByVal sPwd As String = "") As Boolean
  Try
    'Affectation de l'emplacement du fichier commercial
    BaseCial.Name = sGcm
    'Affectation du code utilisateur
    BaseCial.Loggable.UserName = sUid
```

```
        'Affectation du mot de passe
        BaseCial.Loggable.UserPwd = sPwd
        'Ouverture de la base commerciale
        BaseCial.Open()
        Return True
      Catch ex As Exception
        Console.WriteLine("Erreur lors de l'ouverture du fichier gcm : " & ex.Message)
        Return False
      End Try
    End Function

    Private Function CloseBase(ByRef BaseCial As BSCIALApplication100c) As Boolean
      Try
        'Si la base est ouverte, alors fermeture de la base
        If BaseCial.IsOpen Then BaseCial.Close()
        Return True
      Catch ex As Exception
        Console.WriteLine("Erreur lors de la fermeture de la base : " & ex.Message)
        Return False
      End Try
    End Function
End Module
```

Transformation d'un document de vente dans un nouveau bon de livraison en affectant manuellement les numéros de série/lot

Cet exemple permet de transformer l'intégralité du devis ***DE00002*** (contenant des articles suivis en série et en lot) dans un nouveu bon de livraison, en affectant manuellement les numéros série/lot. Après transformation, le devis d'origine n'existera plus car l'intégralité des lignes aura été transformée dans le nouveau bon de livraison.

> L'affectation manuelle des numéros série/lot n'est pas obligatoire. Le processus de transformation les attribuera automatiquement s'ils ne sont pas renseignés.

Code source

```
Option Strict Off
Imports Objets100cLib
Imports System

Module ProcessusVenteLivrer_1DeTo1BlAvecSerieLot
 'Objet base commerciale
 Dim oCial As BSCIALApplication100c
 'emplacement du fichier commercial
 Dim sPathGcm As String = "C:\Temp\Bijou.gcm"
 Sub Main()
  Try
   'Instanciation de l'objet base commercial
   oCial = New BSCIALApplication100c
   'Ouverture de la base
   If OpenBase(oCial, sPathGcm) Then
    'Création du processus Livrer
    Dim pTransfo As IPMDocTransformer = _
     oCial.Transformation.Vente.CreateProcess_Livrer
    'Si le devis DE00002 existe
    If oCial.FactoryDocumentVente.ExistPiece( _
     DocumentType.DocumentTypeVenteDevis, "DE00002") Then
     'Sélection du devis DE00002
     Dim pDoc As IBODocumentVente3 = _
      oCial.FactoryDocumentVente.ReadPiece( _
       DocumentType.DocumentTypeVenteDevis, "DE00002")
     'Ajout du document au processus
     pTransfo.AddDocument(pDoc)
     'Création d'une table permettant de stocker
     'les lots affectés au processus
     Dim hashTb As New Hashtable
     'Si le document contient des lignes
     If pTransfo.ListLignesATransformer.Count > 0 Then
      'Parcours de lignes de document présentes dans le processus
      For Each pLig As IBODocumentVenteLigne3 _
       In pTransfo.ListLignesATransformer
       'Test sur le suivi de stock de l'article
       If pLig.Article.AR_SuiviStock = _
        SuiviStockType.SuiviStockTypeSerie Then
        'Affectation des numéros de série
        SetSerie(pTransfo, pLig)
       ElseIf pLig.Article.AR_SuiviStock = _
        SuiviStockType.SuiviStockTypeLot Then
        'Affectation des numéros de lot
```

```
            SetLot(pTransfo, pLig, hashTb)
          End If
        Next
      End If
      'Test pour savoir si le processus peut être validé
      If pTransfo.CanProcess Then
        'Validation du processus
        pTransfo.Process()
      Else
        'Traitement de récupération des erreurs
        RecupError(CType(pTransfo, IPMProcess))
      End If
    End If
  End If
 Catch ex As Exception
  Console.WriteLine("Erreur : " & ex.Message)
 Finally
  'Fermeture de la connexion
  CloseBase(oCial)
 End Try
End Sub

'Fonction permettant de tester si le numéro de série a déjà été affectée
'à la ligne de document
Private Function SerieAlreadyUse(ByRef pTransfo As IPMDocTransformer, ByRef pLig _
 As IBODocumentVenteLigne3, ByRef pLot As IBOArticleDepotLot) As Boolean
 Dim bRes As Boolean = False
 Try
  'Parcours de toutes les lignes du processus
  For Each pL As IBODocumentVenteLigne3 In pTransfo.ListLignesATransformer
    'Si l'article est identique et que des numéros de
    If Not pL.Article Is Nothing AndAlso pL.Article.Equals(pLig.Article) _
     AndAlso pTransfo.UserLotsToUse(pL).Count > 0 Then
      'Parcours des numéros de série affectés à la ligne
      For i As Integer = 1 To pTransfo.UserLotsToUse(pL).Count
        Dim pTmpUserLot As IUserLot = _
         pTransfo.UserLotsToUse(pL).Item(i)
        'Si le numéro de série que l'on souhaite
        'affecter est déjà associé à une ligne
        If pTmpUserLot.Lot.Equals(pLot) Then
          Return True
        End If
      Next
    End If
  Next
```

```
  Catch ex As Exception
    Console.WriteLine(ex.Message)
  End Try
  Return bRes
End Function

'Procédure permettant d'affecter des numéros de série aux lignes du processus
Private Sub SetSerie(ByRef pTransfo As IPMDocTransformer, _
                     ByRef pLig As IBODocumentVenteLigne3)
  Try
    Dim bReadAllSerie As Boolean = False
    'Tant qu'on a pas parcouru tous les série/lot et
    'qu'il y a une quantité série à fournir
    While Not bReadAllSerie AndAlso pTransfo.UserLotsQteRestantAFournir(pLig) > 0
      Dim pDepot As IBODepot3 = pLig.Depot
      Dim pArtDepot As IBOArticleDepot3 = _
       pLig.Article.FactoryArticleDepot.ReadDepot(pDepot)
      'Parcours des numéros de série pour l'article
      'et pour le dépôt de la ligne
      For Each pArtDepotLot As IBOArticleDepotLot _
       In pArtDepot.FactoryArticleDepotLot.List
        'Si le numéro n'est pas épuisé et s'il n'a pas déjà été affecté
        If Not pArtDepotLot.IsEpuised And pArtDepotLot.StockATerme > 0 And _
         Not SerieAlreadyUse(pTransfo, pLig, pArtDepotLot) Then
          'Création d'un objet série
          Dim pUserLot As IUserLot = pTransfo.UserLotsToUse(pLig).AddNew
          'Affectation du numéro de série
          pUserLot.Set(pArtDepotLot, 1, pArtDepotLot.Complement)
          bReadAllSerie = False
          Exit For
        End If
        bReadAllSerie = True
      Next
    End While
  Catch ex As Exception
    Console.WriteLine(ex.Message)
  End Try
End Sub

'Procédure permettant d'affecter des numéros de lot aux lignes du processus
Private Sub SetLot(ByRef pTransfo As IPMDocTransformer, _
  ByRef pLig As IBODocumentVenteLigne3, ByVal hashTb As Hashtable)
  Try
    Dim bReadAllLot As Boolean = False
    'Tant qu'on a pas parcouru tous les série/lot et
```

```
'qu'il y a une quantité lot à fournir
While Not bReadAllLot AndAlso pTransfo.UserLotsQteRestantAFournir(pLig) > 0
 Dim pDepot As IBODepot3 = pLig.Depot
 Dim pArtDepot As IBOArticleDepot3 = _
  pLig.Article.FactoryArticleDepot.ReadDepot(pDepot)
 Dim i As Integer = 0
 'Parcours des numéros de lot pour l'article
 For Each pArtDepotLot As IBOArticleDepotLot _
  In pArtDepot.FactoryArticleDepotLot.List
  Dim dQteTb As Double = 0
  Dim dQteFournir As Double = _
   pTransfo.UserLotsQteRestantAFournir(pLig)
  'Si le lot n'est pas épuisé
  If Not pArtDepotLot.IsEpuised AndAlso pArtDepotLot.StockATerme > 0 Then
   'Création d'un objet lot
   Dim pUserLot As IUserLot = _
    pTransfo.UserLotsToUse(pLig).AddNew
   'Si le lot a déjà été affecté mais qu'il
   'lui reste une quantité disponible
   If hashTb.ContainsKey(pArtDepotLot) _
    AndAlso hashTb(pArtDepotLot) > 0 Then
    'Récupération de la quantité disponible du lot
    dQteTb = hashTb(pArtDepotLot)
    'Si la quantité à fournir est inférieur
    'à la quantité disponible du lot
    If dQteFournir <= dQteTb Then
     'Affectation du lot
     pUserLot.Set(pArtDepotLot, _
      dQteFournir, pArtDepotLot.Complement)
     'Décrémentation de la quantité
     'disponible du lot dans la table de hash
     hashTb(pArtDepotLot) = dQteTb - dQteFournir
     'Sortie de la boucle car tous les
     'lots ont été affectés à la ligne
     Exit For
    Else
     'Affectation de la quantité restante du lot
     pUserLot.Set(pArtDepotLot, _
      dQteTb, pArtDepotLot.Complement)
     'Le lot passe à indisponible
     hashTb(pArtDepotLot) = 0
     'On ne sort pas de la boucle car il faut peut être
     'trouver d'autres lots pour la ligne
    End If
   Else
```

```
            'Si la quantité à fournir est inférieur à
            'la quantité disponible du lot
            If dQteFournir <= pArtDepotLot.StockATerme Then
              'Affectation du lot
              pUserLot.Set(pArtDepotLot, _
               dQteFournir, pArtDepotLot.Complement)
              'Ajout du lot dans la table de hash
              'avec décrémentation de la quantité disponible
              hashTb.Add(pArtDepotLot, _
               pArtDepotLot.StockATerme - dQteFournir)
              'Sortie de la boucle car tous les lots
              'ont été affectés à la ligne
              Exit For
            Else
              'Affectation de la quantité restante du lot
              pUserLot.Set(pArtDepotLot, _
               pArtDepotLot.StockATerme, pArtDepotLot.Complement)
              'Le lot passe à indisponible et est
              'ajouté dans la table de hash
              hashTb.Add(pArtDepotLot, 0)
              'On ne sort pas de la boucle car il faut peut être
              'trouver d'autres lots pour la ligne
            End If
          End If
        End If
        i += 1
        'Si tous les lots de l'article ont été lus
        If i = pArtDepot.FactoryArticleDepotLot.List.Count Then
          bReadAllLot = True
        End If
      Next
    End While
  Catch ex As Exception
    Console.WriteLine(ex.Message)
  End Try
End Sub

Public Sub RecupError(ByRef mP As IPMProcess)
  Try
    'Boucle sur les erreurs contenues dans la collection
    For i As Integer = 1 To mP.Errors.Count
      'Récupération des éléments erreurs
      Dim iFail As IFailInfo = mP.Errors.Item(i)
      'récupération du numéro d'erreur,
      'de l'indice et de la description de l'erreur
```

```
        Console.WriteLine("Code Erreur : " & _
          iFail.ErrorCode & " Indice : " & iFail.Indice & _
            " Description : " & iFail.Text)
      Next
    Catch ex As Exception
      Console.WriteLine("Erreur : " & ex.Message)
    End Try
  End Sub

  Public Function OpenBase(ByRef BaseCial As BSCIALApplication100c, _
      ByVal sGcm As String, Optional ByVal sUid As String = "<Administrateur>", _
      Optional ByVal sPwd As String = "") As Boolean
    Try
      'Affectation de l'emplacement du fichier commercial
      BaseCial.Name = sGcm
      'Affectation du code utilisateur
      BaseCial.Loggable.UserName = sUid
      'Affectation du mot de passe
      BaseCial.Loggable.UserPwd = sPwd
      'Ouverture de la base commerciale
      BaseCial.Open()
      Return True
    Catch ex As Exception
      Console.WriteLine("Erreur lors de l'ouverture du fichier gcm : " & ex.Message)
      Return False
    End Try
  End Function

  Private Function CloseBase(ByRef BaseCial As BSCIALApplication100c) As Boolean
    Try
      'Si la base est ouverte, alors fermeture de la base
      If BaseCial.IsOpen Then BaseCial.Close()
      Return True
    Catch ex As Exception
      Console.WriteLine("Erreur lors de la fermeture de la base : " & ex.Message)
      Return False
    End Try
  End Function
End Module
```

Transformation d’un document d’achat dans un bon de commande existant avec regroupement de lignes

Cet exemple permet, de transformer une préparation de commande d’achat contenant deux lignes pour le même article, dans un bon de commande existant. Ce bon de commande contient déjà une

ligne pour le même article que celui utilisé dans la préparation de commande. Après transformation, les lignes de la préparation de commande seront automatiquement regroupées. La ligne contenue dans le bon de commande de destination ne sera quant à elle pas impactée.

Code source

```
Option Strict Off

Imports Objets100cLib

Imports System


Module ProcessusAchatCommander_1PcTo1BC
 'Objet base commerciale
 Dim oCial As BSCIALApplication100c
 'emplacement du fichier commercial
 Dim sPathGcm As String = "C:\Temp\Bijou.gcm"


 Sub Main()
  Try
   'Instanciation de l'objet base commercial
   oCial = New BSCIALApplication100c
   'Ouverture de la base
   If OpenBase(oCial, sPathGcm) Then
    'Création du processus Commander
    Dim pTransfo As IPMDocTransformer = _
     oCial.Transformation.Achat.CreateProcess_Commander
    'Si le document à utiliser dans le processus existe
    If oCial.FactoryDocumentAchat.ExistPiece( _
     DocumentType.DocumentTypeAchatCommande, "PC00008") Then
     'Sélection de la préparation de commande PC00008
     Dim pDoc As IBODocumentAchat3 = _
      oCial.FactoryDocumentAchat.ReadPiece( _
     DocumentType.DocumentTypeAchatCommande, "PC00008")
     'Ajout du document au processus
     pTransfo.AddDocument(pDoc)
     'Si le document de destination à utiliser dans le processus existe
```

```
        If oCial.FactoryDocumentAchat.ExistPiece( _
                DocumentType.DocumentTypeAchatCommandeConf, "FBC00005") Then
          'Sélection du bon de commande FBC00005
          pDoc = oCial.FactoryDocumentAchat.ReadPiece( _
           DocumentType.DocumentTypeAchatCommandeConf, "FBC00005")
          'Ajout du document en tant que document de destination
          pTransfo.AddDocumentDestination(pDoc)
        End If
        'Test pour savoir si le processus peut être validé
        If pTransfo.CanProcess Then
          'Validation du processus
          pTransfo.Process()
        Else
          'Traitement de récupération des erreurs
          RecupError(CType(pTransfo, IPMProcess))
        End If
      End If
    End If
  Catch ex As Exception
    Console.WriteLine("Erreur : " & ex.Message)
  Finally
    'Fermeture de la connexion
    CloseBase(oCial)
  End Try
End Sub


Public Sub RecupError(ByRef mP As IPMProcess)
  Try
    'Boucle sur les erreurs contenues dans la collection
    For i As Integer = 1 To mP.Errors.Count
      'Récupération des éléments erreurs
      Dim iFail As IFailInfo = mP.Errors.Item(i)
      'récupération du numéro d'erreur,
```

```
        'de l'indice et de la description de l'erreur
        Console.WriteLine("Code Erreur : " & _
          iFail.ErrorCode & " Indice : " & iFail.Indice & _
            " Description : " & iFail.Text)
      Next
    Catch ex As Exception
      Console.WriteLine("Erreur : " & ex.Message)
    End Try
  End Sub

  Public Function OpenBase(ByRef BaseCial As BSCIALApplication100c, _
      ByVal sGcm As String, Optional ByVal sUid As String = "<Administrateur>", _
      Optional ByVal sPwd As String = "") As Boolean
    Try
      'Affectation de l'emplacement du fichier commercial
      BaseCial.Name = sGcm
      'Affectation du code utilisateur
      BaseCial.Loggable.UserName = sUid
      'Affectation du mot de passe
      BaseCial.Loggable.UserPwd = sPwd
      'Ouverture de la base commerciale
      BaseCial.Open()
      Return True
    Catch ex As Exception
      Console.WriteLine("Erreur lors de l'ouverture du fichier gcm : " & ex.Message)
      Return False
    End Try
  End Function

  Private Function CloseBase(ByRef BaseCial As BSCIALApplication100c) As Boolean
    Try
      'Si la base est ouverte, alors fermeture de la base
      If BaseCial.IsOpen Then BaseCial.Close()
```

```
    Return True
   Catch ex As Exception
    Console.WriteLine("Erreur lors de la fermeture de la base : " & ex.Message)
    Return False
   End Try
  End Function
End Module
```

Transformation d'un document d'achat dans un nouveau bon de livraison avec affectation manuelle des numéros Série/Lot

L'exemple suivant permet de transformer un bon de commande d'achat, dans un nouveau bon de livraison avec gestion manuelle des série/lot. Cet exemple fait de plus appel à la fonction ***NextNoSerie*** (cf. ***IBOArticleDepotLotFactory***), permettant d'incrémenter une chaîne de caractères, afin de calculer les numéros de série/lot.

Code source

```
Option Strict Off
Imports Objets100cLib
Imports System

Module ProcessusAchatReceptionner_1BcTo1BLAvecLot
 'Objet base commerciale
 Dim oCial As BSCIALApplication100c
 'emplacement du fichier commercial
 Dim sPathGcm As String = "C:\Temp\Bijou.gcm"

 Sub Main()
  Try
   'Instanciation de l'objet base commercial
   oCial = New BSCIALApplication100c
   'Ouverture de la base
   If OpenBase(oCial, sPathGcm) Then
    'Création du processus Réceptionner
    Dim pTransfo As IPMDocTransformer = _
     oCial.Transformation.Achat.CreateProcess_Receptionner
```

```
'Si le bon de commande FBC00006 existe
If oCial.FactoryDocumentAchat.ExistPiece( _
 DocumentType.DocumentTypeAchatCommandeConf, "FBC00006") Then
 'Sélection du bon de commande FBC00006
 Dim pDoc As IBODocumentAchat3 = _
 oCial.FactoryDocumentAchat.ReadPiece( _
  DocumentType.DocumentTypeAchatCommandeConf, "FBC00006")
 'Ajout du document au processus
 pTransfo.AddDocument(pDoc)
 'Initialisation des numéros série/lot à créer
 Dim sNumLot As String = "LOT0000"
 Dim sNumSerie As String = "SERIE0000"
 'Parcours des lignes de document présentes dans le processus
 For Each pLig As IBODocumentAchatLigne3 _
  In pTransfo.ListLignesATransformer
  'Si le suivi de l'article est Série/Lot
  If pLig.Article.AR_SuiviStock = _
   SuiviStockType.SuiviStockTypeLot OrElse _
    pLig.Article.AR_SuiviStock = _
     SuiviStockType.SuiviStockTypeSerie Then
    'Récupération du dépôt de la ligne
    Dim pDepot As IBODepot3 = _
    pLig.Depot
    'Récupération du dépôt de stockage de l'article
    Dim pArtDepot As IBOArticleDepot3 = _
     pLig.Article.FactoryArticleDepot.ReadDepot(pDepot)
    'Tant que des série/lot doivent être fournis
    While pTransfo.UserLotsQteRestantAFournir(pLig) > 0
     'Création d'un Lot/Série pour l'article
     Dim pArtDepotLot As IBOArticleDepotLot = _
      pArtDepot.FactoryArticleDepotLot.Create
     'Création d'un lot/série pour la ligne
     Dim pUserLot As IUserLot = _
```

```
            pTransfo.UserLotsToUse(pLig).AddNew
           If pLig.Article.AR_SuiviStock = _
            SuiviStockType.SuiviStockTypeLot Then
             'Calcul d'un numéro de lot inexistant
             While pArtDepotLot.FactoryArticleDepotLot.ExistNoSerie(sNumLot)
               sNumLot = _
                 pArtDepotLot.FactoryArticleDepotLot.NextNoSerie(sNumLot)
             End While
             'Affectation du numéro de lot
             pArtDepotLot.NoSerie = sNumLot
             'Ajout du lot au processus pour toute la quantité nécessaire
             pUserLot.Set(pArtDepotLot, _
              pTransfo.UserLotsQteRestantAFournir(pLig), _
               pArtDepotLot.Complement)
             'Incrémentation du numéro de lot
             sNumLot = _
               pArtDepotLot.FactoryArticleDepotLot.NextNoSerie(sNumLot)
           Else
             'Calcul d'un numéro de série inexistant
             While pArtDepotLot.FactoryArticleDepotLot.ExistNoSerie(sNumSerie)
               sNumSerie = _
                 pArtDepotLot.FactoryArticleDepotLot.NextNoSerie(sNumSerie)
             End While
             'Affectation du numéro de série
             pArtDepotLot.NoSerie = sNumSerie
             'Ajout du numéro de série au processus
             pUserLot.Set(pArtDepotLot, 1, pArtDepotLot.Complement)
             'Incrémentation du numéro de série
             sNumSerie = _
                 pArtDepotLot.FactoryArticleDepotLot.NextNoSerie(sNumSerie)
           End If
         End While
       End If
```

```
        Next
        'Test pour savoir si le processus peut être validé
        If pTransfo.CanProcess Then
          'Validation du processus
          pTransfo.Process()
        Else
          'Traitement de récupération des erreurs
          RecupError(CType(pTransfo, IPMProcess))
        End If
      End If
    End If
  Catch ex As Exception
    Console.WriteLine("Erreur : " & ex.Message)
  Finally
    'Fermeture de la connexion
    CloseBase(oCial)
  End Try
End Sub

Public Sub RecupError(ByRef mP As IPMProcess)
  Try
    'Boucle sur les erreurs contenues dans la collection
    For i As Integer = 1 To mP.Errors.Count
      'Récupération des éléments erreurs
      Dim iFail As IFailInfo = mP.Errors.Item(i)
      'récupération du numéro d'erreur,
      'de l'indice et de la description de l'erreur
      Console.WriteLine("Code Erreur : " & _
        iFail.ErrorCode & " Indice : " & iFail.Indice & _
          " Description : " & iFail.Text)
    Next
  Catch ex As Exception
    Console.WriteLine("Erreur : " & ex.Message)
```

```
    End Try
  End Sub

  Public Function OpenBase(ByRef BaseCial As BSCIALApplication100c, _
    ByVal sGcm As String, Optional ByVal sUid As String = "<Administrateur>", _
    Optional ByVal sPwd As String = "") As Boolean
    Try
      'Affectation de l'emplacement du fichier commercial
      BaseCial.Name = sGcm
      'Affectation du code utilisateur
      BaseCial.Loggable.UserName = sUid
      'Affectation du mot de passe
      BaseCial.Loggable.UserPwd = sPwd
      'Ouverture de la base commerciale
      BaseCial.Open()
      Return True
    Catch ex As Exception
      Console.WriteLine("Erreur lors de l'ouverture du fichier gcm : " & ex.Message)
      Return False
    End Try
  End Function

  Private Function CloseBase(ByRef BaseCial As BSCIALApplication100c) As Boolean
    Try
      'Si la base est ouverte, alors fermeture de la base
      If BaseCial.IsOpen Then BaseCial.Close()
      Return True
    Catch ex As Exception
      Console.WriteLine("Erreur lors de la fermeture de la base : " & ex.Message)
      Return False
    End Try
  End Function
End Module
```

Processus de Lettrage

L'exemple suivant permet de mettre en œuvre le processus **IPMLettrer**, afin de lettrer une collection d'écritures comptables saisies sur le tiers CISEL.

Dans cet exemple, le lettrage sera réalisé en spécifiant le type de lettrage sur **Lettrage montant**. Ainsi, le solde des écritures ajoutées au processus devra être à 0. Les écritures sélectionnées seront les suivantes :

N° Ligne	Journal	Date	Compte général	Compte tiers	Débit	Crédit
479	VTE	050309	4110000	CISEL	55146,07	
146	BEU	140309	4110000	CISEL		35000,00
485	VTE	230309	4110000	CISEL	70000,00	
1194	BEU	310309	4110000	CISEL		90146,07

Pour lettrer ces écritures, il sera nécessaire de sélectionner chacune d'entre elles, et de les ajouter au processus. Dans cet exemple, les accès et ajouts des écritures au processus seront réalisés par l'appel de la fonction ***ReadNumero*** de l'interface ***IBOEcritureFactory3***.

Enfin, le code lettrage à affecter aux écritures sera automatiquement calculé après appel de la méthode ***SetLettreDefault()***.

Code source

```
Option Strict Off

Imports Objets100cLib

Imports System

Module Process_IPMLettrer

    'Objet base comptable

    Dim oCpta As BSCPTAApplication100c

    'emplacement du fichier comptable

    Dim sPathMae As String = "C:\Tmp\Bijou.mae"


    Sub Main()

        Try

            'Instanciation de l'objet comptable

            oCpta = New BSCPTAApplication100c

            'Ouverture de la base

            If OpenBase(oCpta, sPathMae) Then
```

```
'Création du processus de lettrage
Dim pLettre As IPMLettrer = _
  oCpta.CreateProcess_Lettrer()
'Test de l'existence de l'écriture portant le numéro 479
'=> Ecriture de montant 55176.07 au débit
If oCpta.FactoryEcriture.ExistNumero(479) Then
    'Ajout de l'écriture au processus
    pLettre.AddEcriture(oCpta.FactoryEcriture.ReadNumero(479))
End If
'Test de l'existence de l'écriture portant le numéro 146
'=> Ecriture de montant 35000.00 au crédit
If oCpta.FactoryEcriture.ExistNumero(146) Then
    'Ajout de l'écriture au processus
    pLettre.AddEcriture(oCpta.FactoryEcriture.ReadNumero(146))
End If
'Test de l'existence de l'écriture portant le numéro 485
'=> Ecriture de montant 70000.00 au débit
If oCpta.FactoryEcriture.ExistNumero(485) Then
    'Ajout de l'écriture au processus
    pLettre.AddEcriture(oCpta.FactoryEcriture.ReadNumero(485))
End If
'Test de l'existence de l'écriture portant le numéro 1194
'=> Ecriture de montant 90146.07 au crédit
If oCpta.FactoryEcriture.ExistNumero(1194) Then
    'Ajout de l'écriture au processus
    pLettre.AddEcriture(oCpta.FactoryEcriture.ReadNumero(1194))
End If
'Affectation du type de lettrage
pLettre.Type = LettrageType.LettrageTypeLettrageMontant
'Initialisation automatique du code lettrage à utiliser
pLettre.SetLettreDefault()
'Si le processus peut être validé
If pLettre.CanProcess Then
```

```
            'Validation du processus
            pLettre.Process()
        Else
            'Traitement de récupération des erreurs
            'si CanProcess() échoue
            RecupError(CType(pLettre, IPMProcess))
        End If
    End If
  Catch ex As Exception
    Console.WriteLine(ex.Message)
  Finally
    CloseBase(oCpta)
  End Try
End Sub

Public Sub RecupError(ByRef mP As IPMProcess)
  Try
    'Boucle sur les erreurs contenues dans la collection
    For i As Integer = 1 To mP.Errors.Count
      'Récupération des éléments erreurs
      Dim iFail As IFailInfo = mP.Errors.Item(i)
      'récupération du numéro d'erreur,
      'de l'indice et de la description de l'erreur
      Console.WriteLine("Code Erreur : " & _
        iFail.ErrorCode & " Indice : " & iFail.Indice & _
          " Description : " & iFail.Text)
    Next
  Catch ex As Exception
    Console.WriteLine("Erreur : " & ex.Message)
  End Try
End Sub

Public Function OpenBase(ByRef BaseCpta As BSCPTAApplication100c, _
```

```
        ByVal sMae As String, Optional ByVal sUid As String = "<Administrateur>", _
        Optional ByVal sPwd As String = "") As Boolean
        Try
            'Affectation de l'emplacement du fichier comptable
            BaseCpta.Name = sMae
            'Affectation du code utilisateur
            BaseCpta.Loggable.UserName = sUid
            'Affectation du mot de passe
            BaseCpta.Loggable.UserPwd = sPwd
            'Ouverture de la base comptable
            BaseCpta.Open()
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur lors de l'ouverture du fichier mae : " & _
                              ex.Message)
            Return False
        End Try
    End Function

    Private Function CloseBase(ByRef BaseCpta As BSCPTAApplication100c) As Boolean
        Try
            'Si la base est ouverte, alors fermeture de la base
            If BaseCpta.IsOpen Then BaseCpta.Close()
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur lors de la fermeture de la base : " & _
                              ex.Message)
            Return False
        End Try
    End Function
End Module
```

Processus d'insertion de ligne de sous-total

L'exemple présenté ci-dessous permet d'insérer une ligne de sous total dans un document mémoire (utilisation du processus ***IPMDocument***). A la validation des deux processus (***IPMDocInsererSousTotal*** et ***IPMDocument***), le document sera créé en base et contiendra une ligne de sous-total totalisant les deux lignes d'article présentes dans le document.

Code source

```
Option Strict Off

Imports Objets100cLib

Imports System

Module ProcessusCreationSousTotal

   'Objet base commerciale

   Dim oCial As BSCIALApplication100c

   'emplacement du fichier commercial

   Dim sPathGcm As String = "C:\Temp\Bijou.gcm"

   Sub Main()

      Try

         'Instanciation de l'objet base commercial

         oCial = New BSCIALApplication100c

         'Ouverture de la base

         If OpenBase(oCial, sPathGcm) Then

            'Création d'un processus de création de document

            Dim pProc As IPMDocument = _

               oCial.CreateProcess_Document( _

                  DocumentType.DocumentTypeVenteCommande)

            'Affectation du document processus à un objet document

            Dim mDoc As IBODocumentVente3 = pProc.Document

            'Affection du client au document

            mDoc.SetDefaultClient( _

               oCial.CptaApplication.FactoryClient.ReadNumero("CARAT"))

            'Ecriture mémoire du document

            mDoc.WriteDefault()

            'Création d'un processus d'insertion de

            'ligne de sous-total sur le document mémoire
```

```
Dim pSousTotal As IPMDocInsererSousTotal = _
    oCial.CreateProcess_SousTotal(mDoc)
'Création d'une ligne dans le document mémoire
Dim mLig As IBODocumentVenteLigne3 = _
    mDoc.FactoryDocumentLigne.Create
'Affectation d'un article à la ligne
mLig.SetDefaultArticle( _
    oCial.FactoryArticle.ReadReference("GRAVURE"), 1)
'Ecriture mémoire de la ligne
mLig.WriteDefault()
'Ajout de la ligne dans le processus
'd'insertion de ligne de sous-total
pSousTotal.AddDocumentLigne(mLig)
'Création d'une deuxième ligne mémoire
mLig = mDoc.FactoryDocumentLigne.Create
'Affectation d'un article à la ligne
mLig.SetDefaultArticle( _
    oCial.FactoryArticle.ReadReference("BAAR01"), 1)
'Ecriture mémoire de la ligne
mLig.WriteDefault()
'Ajout de la ligne dans le processus
'd'insertion de ligne de sous-total
pSousTotal.AddDocumentLigne(mLig)
'Si le processus d'insertion de ligne
'de sous-total peut être validé
If pSousTotal.CanProcess Then
    'Validation du processus
    '=> Ajout d'une ligne de sous-total dans le document mémoire
    pSousTotal.Process()
    'Si le processus de création de document peut être validé
    If pProc.CanProcess Then
        'Validation du processus
        '=> création du bon de commande. le document contient 3 lignes :
```

```
                '2 lignes d'article et une ligne de sous-total
                pProc.Process()
            Else
                'Récupération des erreurs du
                'processus création de document
                RecupError(CType(pProc, IPMProcess))
            End If
        Else
            'Récupération des erreurs du processus
            'd'insertion de ligne de sous-total
            RecupError(CType(pSousTotal, IPMProcess))
        End If
    End If
  Catch ex As Exception
    Console.WriteLine("Erreur : " & ex.Message)
  Finally
    'Fermeture de la connexion
    CloseBase(oCial)
  End Try
End Sub
Public Sub RecupError(ByRef mP As IPMProcess)
  Try
    'Boucle sur les erreurs contenues dans la collection
    For i As Integer = 1 To mP.Errors.Count
      'Récupération des éléments erreurs
      Dim iFail As IFailInfo = mP.Errors.Item(i)
      'récupération du numéro d'erreur,
      'de l'indice et de la description de l'erreur
      Console.WriteLine("Code Erreur : " & _
        iFail.ErrorCode & " Indice : " & iFail.Indice & _
        " Description : " & iFail.Text)
    Next
  Catch ex As Exception
```

```
        Console.WriteLine("Erreur : " & ex.Message)
    End Try
End Sub

Public Function OpenBase(ByRef BaseCial As BSCIALApplication100c, _
    ByVal sGcm As String, Optional ByVal sUid As String = "<Administrateur>", _
    Optional ByVal sPwd As String = "") As Boolean
    Try
        'Affectation de l'emplacement du fichier commercial
        BaseCial.Name = sGcm
        'Affectation du code utilisateur
        BaseCial.Loggable.UserName = sUid
        'Affectation du mot de passe
        BaseCial.Loggable.UserPwd = sPwd
        'Ouverture de la base commerciale
        BaseCial.Open()
        Return True
    Catch ex As Exception
        Console.WriteLine("Erreur lors de l'ouverture du fichier gcm : " _
                & ex.Message)
        Return False
    End Try
End Function

Private Function CloseBase(ByRef BaseCial As BSCIALApplication100c) As Boolean
    Try
        'Si la base est ouverte, alors fermeture de la base
        If BaseCial.IsOpen Then BaseCial.Close()
        Return True
    Catch ex As Exception
        Console.WriteLine("Erreur lors de la fermeture de la base : " _
                & ex.Message)
        Return False
```

```
        End Try
    End Function
End Module
```

Processus de recalcul de prix de revient

L'exemple présenté ci-dessous permet, dans un bon de fabrication, de modifier la quantité des composants *SVFORMAPP* de la nomenclature *CHORFA* (nomenclature fabrication), et ensuite de lancer un recalcul du prix de revient pour remettre à jour le prix du composé *CHORFA*.

Code source

```
Imports Objets100cLib

Module ProcessRecalculPrixCompose
    'Objet base commerciale
    Dim bCial As BSCIALApplication100c


    Sub Main()
        Try
            'Instanciation de l'objet
            bCial = New BSCIALApplication100c
            'Ouverture de la connexion à la base
            If OpenBase(bCial, "C:\Temp\Bijou.gcm", "<Administrateur>", "") Then
                'Appel de la méthode modifiant les lignes et recalculant le prix de revient
                Call RecalculPrixCompose()
            End If
        Catch ex As Exception
            Console.WriteLine(ex.Message)
        Finally
            'Fermeture de la connexion
            Call CloseBase(bCial)
        End Try
        Console.ReadLine()
    End Sub
```

```
Private Sub RecalculPrixCompose()
    Try
        'Objet Bon de fabrication
        Dim pDoc As IBODocumentStock3 = bCial.FactoryDocumentStock.ReadPiece(DocumentType.DocumentTypeStockFabrication,
"BF00008")
        'Initialisation du processus de recalcul du prix de revient sur le document
        Dim pRecalc As Objets100cLib.IPMDocRecalculPrixCompose = bCial.CreateProcess_RecalculPrixCompose(pDoc)
        'Parcours des lignes du document
        For Each pLig As IBODocumentStockLigne3 In pDoc.FactoryDocumentLigne.List
            'Si la ligne appartient à la nomenclature CHORFA
            If pLig.ArticleCompose.Equals(bCial.FactoryArticle.ReadReference("CHORFA")) Then
                'Si la ligne est un composant SVFORMAPP
                If pLig.Article.Equals(bCial.FactoryArticle.ReadReference("SVFORMAPP")) Then
                    'Majoration de la quantité
                    pLig.DL_Qte = pLig.DL_Qte - 0.5
                    pLig.Write()
                End If
                'Ajout de la ligne au processus
                pRecalc.AddDocumentLigne(pLig)
            End If
        Next
        'Si le processus peut être validé
        If pRecalc.CanProcess Then
            'Recalcul du prix de revient sur le composé CHORFA
            pRecalc.Process()
        Else
            'Traitement des erreurs du processus
            RecupError(CType(pRecalc, IPMProcess))
        End If
    Catch ex As Exception
        Throw ex
    End Try
End Sub
```

```
'Méthode permettant de traiter les erreurs du processus
Public Sub RecupError(ByRef mP As IPMProcess)
    Try
        'Boucle sur les erreurs contenues dans la collection
        For i As Integer = 1 To mP.Errors.Count
            'Récupération des éléments erreurs
            Dim iFail As IFailInfo = mP.Errors.Item(i)
            'Récupération du numéro d'erreur,
            'de l'indice et de la description de l'erreur
            Console.WriteLine("Code Erreur : " & _
            iFail.ErrorCode & " Indice : " & iFail.Indice & _
            " Description : " & iFail.Text)
        Next
    Catch ex As Exception
        Console.WriteLine("Erreur : " & ex.Message)
    End Try
End Sub

'Méthode permettant d'ouvrir la connexion à une base commerciale
Private Function OpenBase(ByRef BaseCial As BSCIALApplication100c, ByVal sGcm As String, _
                          ByVal sUid As String, ByVal sPwd As String) As Boolean
    Try
        'Emplacement du fichier gcm
        BaseCial.Name = sGcm
        'Utilisateur
        BaseCial.Loggable.UserName = sUid
        'Mot de passe
        BaseCial.Loggable.UserPwd = sPwd
        'Ouverture de la connexion
        BaseCial.Open()
        Return True
    Catch ex As Exception
        Console.WriteLine("Erreur lors de l'ouverture du fichier gcm : " & ex.Message)
```

```
            Return False
        End Try
    End Function

    'Méthode permettant de fermer la connexion à la base commerciale
    Private Function CloseBase(ByRef bCial As BSCIALApplication100c) As Boolean
        Try
            'Si la base est ouverte alors fermeture de la connexion
            If bCial.IsOpen Then bCial.Close()
            bCial = Nothing
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur lors de la fermeture de la base : " & ex.Message)
            Return False
        End Try
    End Function
End Module
```

Processus de sortie d'article géré par lot

Exemple 1

Pour un article géré en lot, plusieurs entrées en stocks pour différents numéros de lots et sur différents emplacements ont été réalisés :

Dépôt	Article	Numéro de lot	Quantité	Emplacement
Bijou SA	ARTICLE_LOT	LOT1	7	A1T1N1P1
		LOT2	15	A1T1N1P2
		LOT3	6	A1T1N1P1
		LOT4	10	A1T1N1P2

Une livraison de l'article ARTICLE_LOT pour le client CARAT et une quantité de 10 doit être réalisée. L'emplacement A1T1N1P1 doit être déstocké en priorité sans se soucier des numéros de lot qu'il contient.

Code source

```
Imports Objets100cLib

Module ProcessSortirLots

  Dim bCial As BSCIALApplication100c

  Sub Main()

    Try

      'Instanciation de l'objet

      bCial = New BSCIALApplication100c

      'Ouverture de la connexion à la base

      If OpenBase(bCial, "C:\Users\Public\Documents\Sage\iGestion commerciale\Gescom Bijou.gcm", "<Administrateur>", "") Then

        'Appel de la méthode permettant de créer les lignes de lot

        Call SortirLotEmplacement()

      End If

    Catch ex As Exception

      Console.WriteLine(ex.Message)

    Finally

      'Fermeture de la connexion

      Call CloseBase(bCial)

    End Try

    Console.ReadLine()

  End Sub


  Private Sub SortirLotEmplacement()

    'Objet Bon de livraison

    Dim pDoc As IBODocumentVente3 = bCial.FactoryDocumentVente.CreateType(DocumentType.DocumentTypeVenteLivraison)

    'Affectation du client CARAT

    pDoc.SetDefaultClient(bCial.CptaApplication.FactoryClient.ReadNumero("CARAT"))

    'Validation de l'entête de document

    pDoc.WriteDefault() '=> Le document est persistant dans la base

    'Ajout d'une ligne au document
```

```
    Dim pLig As IBODocumentVenteLigne3 = pDoc.FactoryDocumentLigne.Create
    'Affection de l'article suivi en lot pour une quantité de 10
    pLig.SetDefaultArticle(bCial.FactoryArticle.ReadReference("ARTICLE_LOT"), 10)
    'Recherche de l'emplacement A1T1N1P1 sur le dépôt du document (Bijou SA)
    Dim pDepotEmpl As IBODepotEmplacement = pDoc.DepotStockage.FactoryDepotEmplacement.ReadCode("A1T1N1P1")
    'Création d'un processus SortirLots()
    Dim pProc As IPMSortirLots = bCial.CreateProcess_SortirLots()
    'Initialisation du processus avec la ligne de document (non persistante), en indiquant l'emplacement mais pas de numéro de lot
    pProc.SetLigneDefaultLot(pLig, Nothing, pDepotEmpl)
    'Si le processus peut être validé
    If (pProc.CanProcess) Then
      'Validation du processus
      pProc.Process() '=> Ajout de 2 lignes dans le document pour LOT1 et LOT3
    Else
      'Traitement des erreurs du processus
      getErrorProcess(CType(pProc, IPMProcess))
    End If
  End Sub

  'Méthode permettant de traiter les erreurs du processus
  Public Sub getErrorProcess(ByRef mP As IPMProcess)
    Try
      'Boucle sur les erreurs contenues dans la collection
      For i As Integer = 1 To mP.Errors.Count
        'Récupération des éléments erreurs
        Dim iFail As IFailInfo = mP.Errors.Item(i)
        'Récupération du numéro d'erreur,
        'de l'indice et de la description de l'erreur
        Console.WriteLine("Code Erreur : " &
        iFail.ErrorCode & " Indice : " & iFail.Indice &
        " Description : " & iFail.Text)
      Next
    Catch ex As Exception
```

```
        Console.WriteLine("Erreur : " & ex.Message)
    End Try
End Sub

'Méthode permettant d'ouvrir la connexion à une base commerciale
Private Function OpenBase(ByRef BaseCial As BSCIALApplication100c, ByVal sGcm As String,
                          ByVal sUid As String, ByVal sPwd As String) As Boolean
    Try
        'Emplacement du fichier gcm
        BaseCial.Name = sGcm
        'Utilisateur
        BaseCial.Loggable.UserName = sUid
        'Mot de passe
        BaseCial.Loggable.UserPwd = sPwd
        'Ouverture de la connexion
        BaseCial.Open()
        Return True
    Catch ex As Exception
        Console.WriteLine("Erreur lors de l'ouverture du fichier gcm : " & ex.Message)
        Return False
    End Try
End Function

'Méthode permettant de fermer la connexion à la base commerciale
Private Function CloseBase(ByRef bCial As BSCIALApplication100c) As Boolean
    Try
        'Si la base est ouverte alors fermeture de la connexion
        If bCial.IsOpen Then bCial.Close()
        bCial = Nothing
        Return True
    Catch ex As Exception
        Console.WriteLine("Erreur lors de la fermeture de la base : " & ex.Message)
        Return False
```

```
    End Try
  End Function
End Module
```

Exemple 2

Pour un dossier commercial sur lequel l'option « Contrôler l'unicité des lots » n'est pas activée, un article suivi en lot (référence ARTICLE_LOT), est entré en stock en plusieurs fois sur le même dépôt, mais sur des emplacements différents :

Dépôt	Article	Numéro de lot	Quantité	Emplacement
Bijou SA	ARTICLE_LOT	LOT1	12	A1T1N1P1
			15	A1T1N1P2
			6	A1T1N1P3

Une livraison du LOT1 pour le client CARAT et pour une quantité de 30 doit être réalisée. Les emplacements d'origine du lot n'ont pas besoin d'être spécifiées.

Code source

```
Imports Objets100cLib

Module ProcessSortirLots

  Dim bCial As BSCIALApplication100c

  Sub Main()

    Try

      'Instanciation de l'objet

      bCial = New BSCIALApplication100c

      'Ouverture de la connexion à la base

      If OpenBase(bCial, "C:\Users\Public\Documents\Sage\iGestion commerciale\Gescom Bijou.gcm", "<Administrateur>", "") Then

        'Appel de la méthode permettant de créer les lignes de lot

        Call SortirLot()

      End If

    Catch ex As Exception

      Console.WriteLine(ex.Message)

    Finally
```

```
        'Fermeture de la connexion
        Call CloseBase(bCial)
    End Try
    Console.ReadLine()
End Sub

Private Sub SortirLot()
    'Objet Bon de livraison
    Dim pDoc As IBODocumentVente3 = bCial.FactoryDocumentVente.CreateType(DocumentType.DocumentTypeVenteLivraison)
    'Affectation du client CARAT
    pDoc.SetDefaultClient(bCial.CptaApplication.FactoryClient.ReadNumero("CARAT"))
    'Validation de l'entête de document
    pDoc.WriteDefault() '=> Le document est persistant dans la base
    'Ajout d'une ligne au document
    Dim pLig As IBODocumentVenteLigne3 = pDoc.FactoryDocumentLigne.Create
    'Affection de l'article suivi en lot pour une quantité de 30
    pLig.SetDefaultArticle(bCial.FactoryArticle.ReadReference("ARTICLE_LOT"), 30)
    'Création d'un processus SortirLots()
    Dim pProc As IPMSortirLots = bCial.CreateProcess_SortirLots()
    'Initialisation du processus avec la ligne de document (non persistante), le numéro de lot et sans préciser d'emplacement
    pProc.SetLigneDefaultLot(pLig, "LOT1", Nothing)
    'Si le processus peut être validé
    If (pProc.CanProcess) Then
        'Validation du processus
        pProc.Process() '=> Ajout de 3 lignes dans le documents pour une quantité totale de 30
    Else
        'Traitement des erreurs du processus
        getErrorProcess(CType(pProc, IPMProcess))
    End If
End Sub

'Méthode permettant de traiter les erreurs du processus
Public Sub getErrorProcess(ByRef mP As IPMProcess)
```

```
    Try
        'Boucle sur les erreurs contenues dans la collection
        For i As Integer = 1 To mP.Errors.Count
            'Récupération des éléments erreurs
            Dim iFail As IFailInfo = mP.Errors.Item(i)
            'Récupération du numéro d'erreur,
            'de l'indice et de la description de l'erreur
            Console.WriteLine("Code Erreur : " &
            iFail.ErrorCode & " Indice : " & iFail.Indice &
            " Description : " & iFail.Text)
        Next
    Catch ex As Exception
        Console.WriteLine("Erreur : " & ex.Message)
    End Try
End Sub

'Méthode permettant d'ouvrir la connexion à une base commerciale
Private Function OpenBase(ByRef BaseCial As BSCIALApplication100c, ByVal sGcm As String,
                          ByVal sUid As String, ByVal sPwd As String) As Boolean
    Try
        'Emplacement du fichier gcm
        BaseCial.Name = sGcm
        'Utilisateur
        BaseCial.Loggable.UserName = sUid
        'Mot de passe
        BaseCial.Loggable.UserPwd = sPwd
        'Ouverture de la connexion
        BaseCial.Open()
        Return True
    Catch ex As Exception
        Console.WriteLine("Erreur lors de l'ouverture du fichier gcm : " & ex.Message)
        Return False
    End Try
```

```
    End Function


    'Méthode permettant de fermer la connexion à la base commerciale
    Private Function CloseBase(ByRef bCial As BSCIALApplication100c) As Boolean
        Try
            'Si la base est ouverte alors fermeture de la connexion
            If bCial.IsOpen Then bCial.Close()
            bCial = Nothing
            Return True
        Catch ex As Exception
            Console.WriteLine("Erreur lors de la fermeture de la base : " & ex.Message)
            Return False
        End Try
    End Function
End Module
```

Processus de Règlement des échéances

L'exemple présenté ci-dessous permet de régler toutes les échéances d'un document de vente .

Code source

```
Private Sub ReglementComptant(ByRef bCial As BSCIALApplication100c)
    Try

        'Objet Document vente et Client

        Dim Doc As IBODocumentVente3 =
bCial.FactoryDocumentStock.ReadPiece(DocumentType.DocumentTypeVenteFacture, "FA00028")
        Dim client As IBOTiersPart3 = Doc.TiersPayeur

        ' Création du règlement

        Dim iReglt As IBODocumentReglement = bCial.FactoryDocumentReglement.Create()

        iReglt.TiersPayeur = client
        iReglt.RG_Date = DateTime.Now
        iReglt.RG_Reference = "Référence"
        iReglt.RG_Libelle = "Libellé"
        iReglt.RG_Montant = Doc.DO_NetAPayer - Doc.DO_MontantRegle
        iReglt.Journal = bCial.CptaApplication.FactoryJournal.ReadNumero("BRD")
        iReglt.CompteG = client.CompteGPrinc
        iReglt.WriteDefault()

        'Création du Processus régler les échéances

        Dim pRegler As IPMReglerEcheances = bCial.CreateProcess_ReglerEcheances()
```

```
        pRegler.Reglement = iReglt
        For Each iEcheance As IBODocumentEcheance3 In Doc.FactoryDocumentEcheance.List
            pRegler.AddDocumentEcheance(iEcheance)
        'Possibilité de régler l'échéance avec un montant particulier
            pRegler.AddDocumentEcheanceMontant(iEcheance, 100)
        Next

        pRegler.Process()

        ' Lecture du résultat du Processus

        For Each iECh As IBODocumentReglementEcheance In pRegler.ListLignesResult
            Console.WriteLine(iECh.Echeance.DR_Date.ToString() & " " & iECh.RC_Montant.ToString())
        Next

    Catch ex As Exception
        Console.WriteLine("Erreur : " & ex.Message)
    End Try

End Sub
```

Processus de l'Inventaire

L'exemple présenté ci-dessous fait un Inventaire sur le dépôt "Bijou SA" à la date du 31 décembre avec correction :

- Pour l'article "BAAR01" sur l'emplacement "SECTIONBC7" : nouvelle quantité 7, nouveau prix : 233
- Pour l'article " CHAAR/VAR", "42 cm", "Forçat" : sur l'emplacement par défaut de l'énuméré nouvelle quantité 3, nouveau prix : 111
- Pour l'article " LINGOR18", "LOT-BDF111111" sur l'emplacement "SECTIONBC7" : nouvelle quantité 5, nouveau prix : 66666

Code source

```
private void TestInventaire(ref BSCIALApplication100c BaseCial)
{
    try
    {
        IPMInventaire Inventaire = BaseCial.CreateProcess_Inventaire();
        Inventaire.Depot = BaseCial.FactoryDepot.ReadIntitule("Bijou SA");
        Inventaire.DateInventaire = new DateTime(2021, 12, 31);
        IBODepotEmplacement pDepotEmpl = Inventaire.Depot.FactoryDepotEmplacement.ReadCode("SECTIONBC7");

// Article CMUP : Nouvelle quantité 7, prix 233, sur emplacement « SECTIONBC7 »
        IBOArticle3 cBOArticle = BaseCial.FactoryArticle.ReadReference("BAAR01");
        Inventaire.AddArticle(cBOArticle, 7, 233, pDepotEmpl) ;

// Article double gamme : Nouvelle quantité 3, prix 111, sur emplacement par défaut
        cBOArticle = BaseCial.FactoryArticle.ReadReference("CHAAR/VAR");
        Inventaire.AddArticleDoubleGamme(cBOArticle, 3, 111,
                                         cBOArticle.FactoryArticleGammeEnum1.ReadEnumere("42 cm"),
                                         cBOArticle.FactoryArticleGammeEnum2.ReadEnumere("Forçat"),
                                          null );
// Article Série/lot : Nouvelle quantité 5, prix 66666, sur emplacement « SECTIONBC7 »
        cBOArticle = BaseCial.FactoryArticle.ReadReference("LINGOR18");
```

```
        IBOArticleDepot3 pArtDepot = cBOArticle.FactoryArticleDepot.ReadDepot(Inventaire.Depot);
        IBOArticleDepotLot  pArtDepotLot = (IBOArticleDepotLot)pArtDepot.FactoryArticleDepotLot.ReadNoSerie("LOT-BDF111111");
        Inventaire.AddArticleSerie(cBOArticle, 5, 66666, pArtDepotLot, pDepotEmpl) ;

// Processus d'Inventaire
        Inventaire.Process();

// Résultat : 2 documents : un mouvement de sortie et un mouvement d'entrée
        String sResult = "";
        foreach (IBODocumentStock3 DocStock in Inventaire.ListDocuments)
        {
            sResult = sResult + DocStock.DO_Type.ToString()+" "+DocStock.DO_Piece +"\n";
        }
        MessageBox.Show(sResult, "Liste des Documents de stock");
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Resultat Errors");
    }
}
```

Gestion des identifiants

Vous trouverez ci-dessous, un exemple complet permettant de mettre en oeuvre la sérialisation et désérialisation des identifiants Sage 100c. Cet exemple contient 4 méthodes :

- **SerializeTiersToString** : Méthode permettant de retourner la clé d'un tiers sous forme de chaîne de caractères
- **SerializeTiersToFileStream** : Méthode permettant de stocker la clé d'un tiers dans un fichier
- **UnSerializeFileStream** : Méthode permettant d'accéder à un objet tiers depuis sa clé stockée dans un fichier
- **UnSerializeString** : Méthode permettant d'accéder à un objet contact tiers depuis sa clé passée sous forme de chaîne de caractères

Cet exemple contient également une classe implémentant l'interface **IStream** (Classe *SerializeOM*), nécessaire pour gérer les identifiants des bases Sage 100c.

Code source

```
using System;

using System.Text;

using System.Runtime.InteropServices;

using Objets100cLib;

using System.IO;
```

```
namespace SerializeOM
{
    public class Program
    {
        static void Main(string[] args)
        {
            //Instanciation d'un objet implémentant la classe IStream
            SerializeOM mObj = new SerializeOM();
            try
            {
                //Ouverture de la connexion à la base comptable
                if (mObj.OpenCpta("<Administrateur>", "", @"C:\Users\Public\Documents\Sage\iComptabilité\Compta Bijou.mae"))
                {
                    //Sérialisation du tiers CARAT dans une chaîne de caractères
                    string sKey = mObj.SerializeTiersToString("CARAT");
                    Console.WriteLine(sKey);
                    //Sérialisation du tiers CISEL dans un fichier
                    sKey = mObj.SerializeTiersToFileStream("CISEL");
                    Console.WriteLine(sKey);
                    //Désérialisation d'un fichier pour retourner le numéro de compte tiers
                    string sCtNum = mObj.UnSerializeFileStream(System.IO.Path.GetDirectoryName
                            (System.Reflection.Assembly.GetExecutingAssembly().CodeBase).Replace(@"file:\","") + @"\OID\CISEL.txt");
                    Console.WriteLine(sCtNum);
                    //Désérialisation d'une chaîne pour retourner le nom et prénom du contact tiers
                    string sContact = mObj.UnSerializeString("1114112|com.sage.om.cpta.TIERSCONTACT|CISEL|8");
                    Console.WriteLine(sContact);
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
```

```
        finally
        {
            //Fermeture de la connexion
            mObj.CloseCpta();
            Console.ReadLine();
        }
    }
}

public class SerializeOM
{
    // Objet base comptable
    private BSCPTAApplication100c _mCpta;
    // Méthode d'ouverture de la connexion à la base de données
    public bool OpenCpta(string uid, string pwd, string path)
    {
        try
        {
            _mCpta = new BSCPTAApplication100c();
            _mCpta.Name = path;
            _mCpta.Loggable.UserName = uid;
            _mCpta.Loggable.UserPwd = pwd;
            _mCpta.Open();
            return true;
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
            return false;
        }
    }
    // Méthode de fermeture de la connexion à la base de données
    public void CloseCpta()
```

```
{
    if (_mCpta != null)
        _mCpta.Close();
}
// Méthode de création d'un FileStream
private FileStream CreateFileStream(string ctnum)
{
    try
    {
        //Récupération de l'emplacement de l'exécutable
        string strPath = System.IO.Path.GetDirectoryName(System.Reflection.Assembly.GetExecutingAssembly().CodeBase);
        strPath += @"\OID";
        string strPathFolder = strPath.Replace(@"file:\", "");
        //Création du répertoire OID à côté de l'exécutable
        if (!System.IO.Directory.Exists(strPath))
            System.IO.Directory.CreateDirectory(strPathFolder);
        string sNameFile = strPathFolder + @"\" + ctnum + ".txt";
        //Création d'un fichier portant le nom passé en paramètre
        return new FileStream(sNameFile, System.IO.FileMode.Create);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
        return null;
    }
}
// Méthode de sérialisation d'un objet de type IBOTiers3 dans un FileStream
public string SerializeTiersToFileStream(string ctnum)
{
    string sKey = "";
    try
    {
        // Objet tiers correspondant au CT_Num passé en paramètre
```

```
            IBOTiers3 pTiers = _mCpta.FactoryTiers.ReadNumero(ctnum);
            // Création d'un FileStream
            FileStream mFileStream = CreateFileStream(ctnum);
            if (mFileStream != null)
            {
                // Instanciation de la classe d'implémentation IStream
                ImplementIStream mStream = new ImplementIStream(mFileStream);
                // Affectation du IStream correspondant au tiers
                pTiers.WriteTo(mStream);
                // Récupération de l'identifiant sous forme de chaîne de caractères
                sKey = mStream.ToString();
                // Fermeture du FileStream
                mFileStream.Close();
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
        return sKey;
    }
    // Méthode de sérialisation d'un objet de type IBOTiers3 dans une chaîne de caractères
    public string SerializeTiersToString(string ctnum)
    {
        string sKey = "";
        try
        {
            // Objet tiers correspondant au CT_Num passé en paramètre
            IBOTiers3 pTiers = _mCpta.FactoryTiers.ReadNumero(ctnum);
            // Création d'un MemoryStream
            MemoryStream mMemStream = new MemoryStream();
            // Instanciation de la classe d'implémentation IStream
            ImplementIStream mStream = new ImplementIStream(mMemStream);
```

```
            // Affectation du IStream correspondant au tiers
            pTiers.WriteTo(mStream);
            // Récupération de l'identifiant sous forme de chaîne de caractères
            sKey = mStream.ToString();
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
        return sKey;
    }
    //Méthode de désérialisation d'un fichier
    public string UnSerializeFileStream(string sPath)
    {
        try
        {
            //Création d'un filestream sur le fichier passé en paramètre
            FileStream mFileStream = new FileStream(sPath, System.IO.FileMode.Open);
            //Instanciation de la classe d'implémentation IStream
            ImplementIStream mStream = new ImplementIStream(mFileStream);
            //Récupération du tiers
            IBIPersistObject mObj = _mCpta.ReadFrom(mStream);
            //Test du type
            if (mObj is IBOTiers3)
            {
                IBOTiers3 mTiers = (IBOTiers3)mObj;
                return mTiers.CT_Num;
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
```

```
            return "";
        }
        //Méthode de désérialisation d'une chaîne de caractères
        public string UnSerializeString(string sKey)
        {
            try
            {
                //Conversion de la chaîne d'entrée en tableau de byte
                byte[] mByte = System.Text.Encoding.UTF8.GetBytes(sKey);
                //Création d'un objet MemoryStream
                MemoryStream mMemStream = new MemoryStream();
                //Initialisation du MemoryStream avec le tableau de byte
                mMemStream.Write(mByte, 0, mByte.Length);
                //Positionnement de la lecture au début du MemoryStream
                mMemStream.Seek(0, SeekOrigin.Begin);
                //Instanciation de la classe d'implémentation IStream
                ImplementIStream mStream = new ImplementIStream(mMemStream);
                //Récupération de l'objet
                IBIPersistObject mObj = _mCpta.ReadFrom(mStream);
                //Test du type
                if (mObj is IBOTiersContact3)
                    {
                        IBOTiersContact3 mContact = (IBOTiersContact3)mObj;
                        return mContact.Nom + " " + mContact.Prenom;
                    }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
            return "";
        }
    }
```

```
//Classe d'implémentation de l'interface IStream
public class ImplementIStream : IStream, IDisposable
{
    private Stream _mStream;

    public ImplementIStream(Stream s)
    {
        _mStream = s;
    }

    public void Clone(out IStream ppstm)
    {
        MemoryStream memBuffer = new MemoryStream();
        byte[] memByte = new byte[_mStream.Length];
        _mStream.Read(memByte, 0, memByte.Length);
        memBuffer.Write(memByte, 0, memByte.Length);
        ppstm = new ImplementIStream(memBuffer);
    }

    public void Commit(uint grfCommitFlags)
    {
        _mStream.Flush();
    }

    public void LockRegion(_ULARGE_INTEGER libOffset, _ULARGE_INTEGER cb, uint dwLockType)
    {
    }

    public void RemoteCopyTo(IStream pstm, _ULARGE_INTEGER cb, out _ULARGE_INTEGER pcbRead, out _ULARGE_INTEGER
pcbWritten)
    {
        pcbRead.QuadPart = pcbWritten.QuadPart = 0;
    }
```

```
public unsafe void RemoteRead(out byte pv, uint cb, out uint pcbRead)
{
    fixed (byte* bBuffer = &pv)
    {
        byte* pByte = bBuffer;
        long remainsRead = _mStream.Length - _mStream.Position;
        uint remains = (uint)(remainsRead > cb ? cb : remainsRead);
        for (uint i = 0; i < remains; i++)
        {
            *pByte = (byte)_mStream.ReadByte();
            pByte++;
        }
        pcbRead = remains;
    }
}

public unsafe void RemoteWrite(ref byte pv, uint cb, out uint pcbWritten)
{
    byte[] arBuf = new byte[cb];
    fixed (byte* bBuffer = &pv)
    {
        byte* pByte = bBuffer;
        for (int i = 0; i < cb; i++)
        {
            arBuf[i] = *pByte;
            _mStream.WriteByte(*pByte);
            pByte++;
        }
    }
    pcbWritten = cb;
}
```

```
public void RemoteSeek(_LARGE_INTEGER dlibMove, uint dwOrigin, out _ULARGE_INTEGER plibNewPosition)
{
    ulong pos = (ulong)_mStream.Seek(dlibMove.QuadPart, (SeekOrigin)dwOrigin);
    plibNewPosition.QuadPart = pos;
}

public void Revert()
{
}

public void SetSize(_ULARGE_INTEGER libNewSize)
{
    _mStream.SetLength((long)libNewSize.QuadPart);
}

public void Stat(out tagSTATSTG pstatstg, uint grfStatFlag)
{
    pstatstg = new tagSTATSTG();
}

public void UnlockRegion(_ULARGE_INTEGER libOffset, _ULARGE_INTEGER cb, uint dwLockType)
{
}

public override string ToString()
{
    StreamReader streamReader = new StreamReader(_mStream);
    _mStream.Seek(0, new SeekOrigin());
    return streamReader.ReadToEnd();
}

public void Dispose()
{
```

```
            if (_mStream == null)
            {
                return;
            }
            _mStream.Dispose();
            _mStream = null;
        }
    }

    public class StreamIStreamWrapper : IStream, IDisposable
    {
        private Stream _mStream;

        public StreamIStreamWrapper(Stream s)
        {
            _mStream = s;
        }

        public void Clone(out IStream ppstm)
        {
            MemoryStream buffer = new MemoryStream();
            byte[] pMem = new byte[_mStream.Length];
            _mStream.Read(pMem, 0, pMem.Length);
            buffer.Write(pMem, 0, pMem.Length);
            ppstm = new ImplementIStream(buffer);
        }

        public void Commit(uint grfCommitFlags)
        {
            _mStream.Flush();
        }

        public void LockRegion(_ULARGE_INTEGER libOffset, _ULARGE_INTEGER cb, uint dwLockType)
```

```
    {
    }

    public void RemoteCopyTo(IStream pstm, _ULARGE_INTEGER cb, out _ULARGE_INTEGER pcbRead, out _ULARGE_INTEGER
pcbWritten)
    {
        pcbRead.QuadPart = pcbWritten.QuadPart = 0;
    }

    public unsafe void RemoteRead(out byte pv, uint cb, out uint pcbRead)
    {
        fixed (byte* pBuf = &pv)
        {
            byte* pd = pBuf;

            long RemainsToReadFromStream = _mStream.Length - _mStream.Position;
            uint Remains = (uint)(RemainsToReadFromStream > cb ? cb : RemainsToReadFromStream);
            for (uint i = 0; i < Remains; i++)
            {
                *pd = (byte)_mStream.ReadByte();
                pd++;
            }
            pcbRead = Remains;
        }
    }

    public unsafe void RemoteWrite(ref byte pv, uint cb, out uint pcbWritten)
    {
        byte[] buf = new byte[cb];
        fixed (byte* pBuf = &pv)
        {
            byte* pd = pBuf;
            for (int i = 0; i < cb; i++)
            {
```

```
                buf[i] = *pd;
                _mStream.WriteByte(*pd);

                pd++;
            }
        }
        pcbWritten = cb;
    }

    public void RemoteSeek(_LARGE_INTEGER dlibMove, uint dwOrigin, out _ULARGE_INTEGER plibNewPosition)
    {
        ulong position = (ulong)_mStream.Seek(dlibMove.QuadPart, (SeekOrigin)dwOrigin);
        plibNewPosition.QuadPart = position;
    }

    public void Revert()
    {
    }

    public void SetSize(_ULARGE_INTEGER libNewSize)
    {
        _mStream.SetLength((long)libNewSize.QuadPart);
    }

    public void Stat(out tagSTATSTG pstatstg, uint grfStatFlag)
    {
        pstatstg = new tagSTATSTG();
    }

    public void UnlockRegion(_ULARGE_INTEGER libOffset, _ULARGE_INTEGER cb, uint dwLockType)
    {
    }
```

```
        public override string ToString()
        {
            StreamReader reader = new StreamReader(_mStream);
            _mStream.Seek(0, new SeekOrigin());
            return reader.ReadToEnd();
        }

        public void Dispose()
        {
            if (_mStream == null)
            {
                return;
            }
            _mStream.Dispose();
            _mStream = null;
        }
    }
}
```

Processus de conversion d'un prospect en client

L'exemple C# suivant permet, de créer un client de type prospect, pour ensuite le transformer en client. Dans cet exemple, la connexion à la base de données s'effectue directement en renseignant le nom de l'instance et nom de la base SQL (n'utilise pas les emplacements des fichiers mae et gcm).

Code source

```
using System;
using Objets100cLib;

namespace ProcessConversionClient
{
    class Program
    {
        // Objet de connexion
        private static BSCIALApplication100c cial = new BSCIALApplication100c();

        static void Main(string[] args)
        {
            try
            {
                // Ouverture de la connexion
                if (OpenDatabase(ref cial, @".\SQL2017", "BIJOU", "<Administrateur>", ""))
                {
```

```
            //Appel de la méthode de création et conversion d'un prospect
            CreateAndConvertProspect();
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    finally
    {
        CloseDatabase(ref cial);
    }
}

/// <summary>
/// Méthode permettant de créer puis convertir un prospect
/// </summary>
private static void CreateAndConvertProspect()
{
    //Initialisation d'un client de type prospect => Ne peut être réalisé que depuis la méthode FactoryProspect
    IBOClient3 prospect = (IBOClient3)cial.CptaApplication.FactoryProspect.Create();
    prospect.CT_Num = "JADE";
    prospect.CT_Intitule = "Prospect Jade";
    prospect.CT_Classement = "Jade";
    //Création du prospect dans la base de données
    prospect.WriteDefault();

    //Initialisation d'un processus de conversion d'un prospect en client
    IPMConversionClient processConversion = cial.CreateProcess_ConversionClient(prospect);
    processConversion.Intitule = "Client Jade";
    processConversion.NumPayeur = prospect;
    processConversion.NumPrinc = cial.CptaApplication.FactoryCompteG.ReadNumero("4110000");

    try
    {
        //Validation du processus
        processConversion.Process();
    }
    catch
    {
        //Parcours des erreurs du processus
        for (int i = 1; i <= processConversion.Errors.Count; i++)
        {
            IFailInfo failInfo = processConversion.Errors[i];
            Console.WriteLine(failInfo.Text);
        }
    }
}

/// <summary>
/// Méthode permettant d'ouvrir la connexion à une base commerciale
/// </summary>
/// <param name="stream">Objet stream commercial</param>
/// <param name="server">Serveur SQL</param>
/// <param name="database">Base de données SQL</param>
/// <param name="user">Utilisateur Sage 100c</param>
/// <param name="pwd">Mot de passe</param>
/// <returns></returns>
private static Boolean OpenDatabase(ref BSCIALApplication100c stream, string server, string database, string user, string pwd)
{
    stream.CompanyServer = server;
    stream.CompanyDatabaseName = database;
    stream.Loggable.UserName = user;
    stream.Loggable.UserPwd = pwd;
    stream.Open();

    return stream.IsOpen;
}

/// <summary>
/// Méthode permettant de fermer la connexion à une base commerciale
/// </summary>
/// <param name="stream">Objet stream commercial</param>
private static void CloseDatabase(ref BSCIALApplication100c stream)
```

```
        {
            if (stream != null && stream.IsOpen)
                stream.Close();
        }
    }
}
```

Annexes

Introduction

Les annexes détaillent les différentes fonctionnalités de Sage 100cloud Objets Métiers 100.

Elles sont classées par type d'application (Cpta.Stream et Cial.Stream).

Les interfaces sont décrites dans l'ordre suivant :

- Interfaces communes aux applications
- Classe application
- Interface Stream
- Interfaces métiers (Ixxx et IBIxxx)
- Interfaces factory paramètres (IBPxxxFactory)
- Interfaces factory métiers (IBOxxxFactory)
- Interfaces objets paramètres (IBPxxx)
- Interfaces objets métiers (IBOxxx)

Les propriétés et méthodes visibles sous un explorateur d'objets et dont le nom commence par le caractère « _ » (_SetSynchronized() par exemple) sont des éléments réservés à usage interne Sage. C'est pourquoi ces propriétés et méthodes ne sont **pas documentées et ne doivent donc pas être utilisées dans des développements spécifiques**.

Conventions d'écriture

Sage 100cloud Objets Métiers propose un certain nombre de classes, d'interfaces et d'énumérateurs répartis en 3 catégories :

Les classes application (BSxxxApplication)

Ce sont les classes de base qui autorisent l'accès aux bases de données, la création, l'ouverture et la fermeture des bases de données. Elles proposent également des propriétés *Factory* permettant la fabrication des Objets Métiers proprement dits (IBOxxx et IBPxxx).

Sage 100cloud Objets Métiers propose deux classes de type IBIPersistStream permettant d'accéder aux bases de données Sage 100c:

- BSCPTAApplication100c : accès à la base comptable ;
- BSCIALApplication100c : accès à la base commerciale.

Les propriétés et méthodes de ces deux classes sont en totalité héritées d'interfaces auquel il faut se reporter pour le détail des propriétés et méthodes disponibles (Cf. section *Les interfaces*).

Les interfaces (IBOxxx, IBPxxx, IBIxxx et Ixxx)

Les interfaces d'objets (IBOxxx) et d'objets paramètres (IBPxxx) héritent des méthodes et propriétés d'interfaces de type interface (IBIxxx).

Généralement, seules les interfaces objets (IBOxxx) et paramètres (IBPxxx) sont directement utilisables.

Il existe d'autres interfaces (Ixxx) utilisées par les d'interfaces objets (IBOxxx), paramètres (IBPxxx) ou interfaces (IBIxxx).

Les énumérateurs

Cf. *Description de Sage 100cloud Objets Métiers – Utilisation des énumérateurs*.

Interfaces communes aux applications

IBICollection

Collection d'objets.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Count() As Long	Nombre d'éléments dans la collection.
Lecture seule	Item(ByVal *lIndex* As Integer) As IBIPersistObject	N ieme élément de la collection.

IBICollectionDispatch

Collection générique représentant un paramétrage de processus, ou une collection d’objets non persistants.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Count() As Long	Nombre d'éléments dans la collection.
Lecture seule	Item(ByVal lIndex As Integer) As Object	N ieme élément de la collection.

Méthodes

Syntaxe	Description
AddNew() As Object	Ajoute un élément (vide par défaut) à la collection.
Remove(*pObjet* As Object)	Supprime l'élément de la collection.
RemoveAll()	Supprime tous les éléments de la collection.

IBIContact2

Contact.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Fonction() As String	Fonction.
Lecture / Ecriture	Nom() As String	Nom.
Lecture / Ecriture	Prenom() As String	Prenom.
Lecture / Ecriture	ServiceContact() As IBPServiceContact	Service contact.
Lecture / Ecriture	Telecom() As ITelecom	Interface télécommunication
Lecture / Ecriture	Civilite() As ContactCivilite	Civilité (Mademoiselle, Madame, Monsieur).
Lecture / Ecriture	TypeContact() As IBPContact	Type de contact.
Lecture / Ecriture	Facebook() As String	Compte Facebook.
Lecture / Ecriture	LinkedIn() As String	Compte LinkedIn.
Lecture / Ecriture	Skype() As String	Compte Skype.

IBIField

Champ (information libre).

Propriétés

Accès	Syntaxe	Description
Lecture seule	Formule() As String	Formule de calcul (Si IsCalculable = *True*).
Lecture seule	IsCalculable() As Boolean	Valeur calculée (*True*) ou non (*False*).
Lecture seule	Name() As String	Intitulé.
Lecture seule	Size() As Short	Longueur (Si Type = Texte).
Lecture seule	Type() As FieldType	Type (cf. énumérateur FieldType).

IBIFields

Collection de champs (informations libres).

Propriétés

Accès	Syntaxe	Description
Lecture seule	Count() As Integer	Retourne le nombre d'éléments dans la collection.
Lecture seule	Item(ByVal lIndex As Integer) As IBIField	Retourne un objet Champ information libre correspondant au Nieme élément de la collection ou à l'intitulé passé en paramètre.

IBILoggable

Autorisations d'accès à la base de données.

Propriétés

Accès	Syntaxe	Description
Lecture seule	IsAdministrator() As Boolean	Accès à la base en tant qu'administrateur (*True*) ou non (*False*).
Lecture seule	IsLogged() As Boolean	Loggé (*True*) ou non (*False*) à la base de données.
Lecture / Ecriture	UserName() As String	Nom de l'utilisateur.
Ecriture seule	UserPwd() As String	Mot de passe utilisateur.

IBIMedia

Fichier multimédia.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	ME_Commentaire() As String	Commentaire.
Lecture / Ecriture	ME_Fichier() As String	Chemin et nom du fichier multimédia. Possibilité de renseigner une url (V.8)
Lecture / Ecriture	ME_TypeMIME() As String	Type MIME.
Lecture / Ecriture	ME_Origine() As String	Origine du lien.

IBIObjectID

Identifiant d'objet.

Propriété

Accès	Syntaxe	Description
Lecture seule	ClassName() As String	Nom de la classe.

Méthode

Syntaxe	Description
Cmp(ByVal *pId* As IBIObjectID) As Short	Comparaison de l'objet avec un autre objet passé en paramètre : Si ObjetA = ObjetB alors ObjetA.Cmp(ObjetB) = **0** Si ObjetA < ObjetB alors ObjetA.Cmp(ObjetB) = **-1** Si ObjetA > ObjetB alors ObjetA.Cmp(ObjetB) = **1**

IBIPersistObject

Objet persistant.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Factory() As IBITypeObjectFactory	Fabrique un objet persistant.
Lecture seule	IsModified() As Boolean	Si au moins une des propriétés de l'objet est modifiée alors : IsModified = **True** Sinon : IsModified = **False** Lorsque l'objet est écrit dans la base de données alors : IsModified = **False**
Lecture seule	IsPersistant() As Boolean	Si l'objet est persistant alors : IsPersistant = **True** Sinon : IsPersistant = **False**
Lecture seule	OID() As IBIObjectID	Identifiant de l'objet.
Lecture seule	Stream() As IBIPersistStream	Objet application dont dépend l'objet persistant.
Lecture seule	OIDExternal() As IBIObjectIDExternal	Propriété réservée. Ne pas utiliser.

Méthodes

Syntaxe	Description
Cmp(ByVal *pObj* As IBIPersistObject) As Integer	Comparaison de l'objet persistant avec un autre objet persistant passé en paramètre : Si ObjetA = ObjetB alors ObjetA.Cmp(ObjetB) = **0** Si ObjetA < ObjetB alors ObjetA.Cmp(ObjetB) = **-1** Si ObjetA > ObjetB alors ObjetA.Cmp(ObjetB) = **1**
CouldModified()	Méthode réservée.
Read()	Lecture de l'objet.
Refresh()	Relit l'objet en base si celui-ci est à l'état modifié.
Remove()	Suppression de l'objet.
SetDefault()	Intialisation de certaines valeurs de l'objet (Cf. objets héritant de *IBIPersistObject*).
Write()	Ecriture de l'objet dans la base de données.

Syntaxe	Description
WriteDefault()	Ecriture de l'objet dans la base de données et création de certains sous-objets liés (Cf. objets héritant de *IBIPersistObject*).
WriteTo(*pStream* As IStream)	Permet d'écrire l'identifiant de l'élément courant dans un IStream.

IBIPersistStream

Base de données.

Propriétés

Accès	Syntaxe	Description
Lecture seule	IsOpen() As Boolean	Indique si la base de données est ouverte (True) ou non (False).
Lecture seule	DatabaseInfo() As IDatabaseinfo	Informations sur la base de données.

Méthodes

Syntaxe	Description
Close()	Ferme la base de données.
Create()	Créé une nouvelle base de données.
Open()	Ouvre la base de données.

IBITypeObjectFactory

Fabrique un objet type.

Propriété

Accès	Syntaxe	Description
Lecture seule	List() As IBICollection	Retourne une collection d'objets.

Méthodes

Syntaxe	Description
Create() As IBIPersistObject	Créé un objet persistant.
Read(ByVal *pOID* As IBIObjectID) As IBIPersistObject	Renvoi l'objet persistant correspondant à l'identifiant d'objet passé en paramètre.
ReadFrom(pStream As IStream) As IBIPersistObject	Retourne l'objet correspondant au IStream passé en paramètre.

IBIValues

Collection de valeurs (informations libres).

Propriétés

Accès	Syntaxe	Description
Lecture seule	Count() As Integer	Retourne le nombre d'éléments dans la collection.
Lecture seule	Item(ByVal *vIndex* As Object) As Object	Retourne la valeur correspondant au Nieme élément de la collection ou à l'intitulé de l'information libre passé en paramètre

IBIValuesInserTable

Collection des énumérés statistiques (articles et tiers).

Propriétés

Accès	Syntaxe	Description
Lecture seule	Count() As Integer	Retourne le nombre d'éléments dans la collection.
Lecture seule	Item(ByVal *vIndex* As Object) As Object	Retourne la valeur correspondant au Nieme élément de la collection ou à l'intitulé de l'information libre passé en paramètre.

Méthodes

Syntaxe	Description
Add(ByVal *vVal* As Object)	Ajoute à la collection l'élément passé en paramètre.
Remove(ByVal *vVal* As Object)	Supprime de la collection l'élément passé en paramètre.

IAdresse

Informations d'adresse.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Adresse() As String	Adresse.
Lecture seule	bGereCodeRegion() As Boolean	Permet de tester si l'adresse gère la propriété Code région.

Accès	Syntaxe	Description
Lecture seule	bGerePays() As Boolean	Permet de tester si l'adresse gère la propriété Code pays.
Lecture / Ecriture	CodePostal() As String	Code postal.
Lecture / Ecriture	CodeRegion() As String	Code région.
Lecture / Ecriture	Complement() As String	Complément d'adresse.
Lecture / Ecriture	Pays() As String	Pays.
Lecture / Ecriture	Ville() As String	Ville.

ICompanies

Liste des bases de données Sage 100c de l'instance SQL.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Count() As Long	Nombre d'éléments.
Lecture seule	Item(ByVal *IIndex* As Long) As Company	Informations sur la base de données.

ICompany

Informations sur la base de données.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Id() As String	GUID de la base de données.
Lecture seule	Name() As String	Nom de la base de données.

IDatabaseInfo

Informations sur la base de données.

Propriétés

Accès	Syntaxe	Description
Lecture seule	ServerName() As String	Nom du serveur de base de données.
Lecture seule	DatabaseName() As String	Nom de la base de données.
Lecture seule	DatabaseType() As DatabaseType	Type de base de données (cf. énumérateur *DatabaseType*).

IDateTimePeriod

Période de date.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Start() As Date	Date de début.
Lecture / Ecriture	End() As Date	Date de fin.

IDevise

Elément Devise.

Propriétés

Accès	Syntaxe	Description
Lecture seule	D_AncCours() As Double	Cours ancienne cotation.
Lecture seule	D_AncDate() As Date	Date ancienne cotation.
Lecture seule	D_AncMode() As DeviseMode	Mode ancienne cotation (cf. énumérateur *DeviseMode*).
Lecture seule	D_CodeISO() As String	Code ISO.
Lecture seule	D_CodeISONum() As String	Code ISO numérique.
Lecture seule	D_CodeRemise() As DeviseRemise	Code remise (cf. énumérateur *DeviseRemise*).
Lecture seule	D_Cours() As Double	Cours.
Lecture seule	D_CoursClot() As Double	Cours clôture.
Lecture seule	D_CoursP() As Double	Cours période.
Lecture seule	D_Euro() As Boolean	Zone euro.
Lecture seule	D_Format() As String	Format.
Lecture seule	D_Intitule() As String	Intitulé devise.
Lecture seule	D_Mode() As DeviseMode	Mode de cotation (cf. énumérateur *DeviseMode*).
Lecture seule	D_Monnaie() As String	Unité monétaire.
Lecture seule	D_Sigle() As String	Sigle.
Lecture seule	D_SousMonnaie() As String	Sous-unité monétaire.

Méthode

Syntaxe	Description
Convert(*pDevise* As IDevise, ByVal *dVal* As Double) As Double	Permet de convertir la valeur *dVal* dans la devise passée en paramètre.

ILicence

Licence d'une application (Cf. IBSCPTAApplication100c et IBSCIALApplication100c).

Propriétés

Accès	Syntaxe	Description
Lecture seule	IsValid() As Boolean	Licence valide (True) ou non (False).
Ecriture seule	Key() As String	Clé (réservé à l'usage interne).
Lecture seule	Product() As ILicenceProduct	Produit "Objets 100".
Lecture seule	Products() As ILicenceProducts	Produits installés sur le poste de travail.

ILicenceProduct

Licence d'un produit (Cf. ILicence).

Propriétés

Accès	Syntaxe	Description
Lecture seule	Name() As String	Nom du produit.
Lecture seule	SerialNumber() As String	Numéro de série.
Lecture seule	UserInfo() As ILicenceUserInfo	Informations utilisateur.

ILicenceProducts

Licence de plusieurs produits (Cf. ILicence).

Propriétés

Accès	Syntaxe	Description
Lecture seule	Count() As Integer	Nombre de licences produit sur le installées sur le poste.
Lecture seule	Item(ByVal *lIndex* As Integer) As ILicenceProduct	N ieme licence produit.

ILicenceUserInfo

Informations sur l'utilisateur de la licence d'un produit (Cf. ILicenceProduct).

Propriétés

Accès	Syntaxe	Description
Lecture seule	Adresse1() As String	Adresse 1.
Lecture seule	Adresse2() As String	Adresse 2.
Lecture seule	CodePostal() As String	Code postal.
Lecture seule	NAF() As String	N.A.F. (APE).
Lecture seule	Pays() As String	Pays.
Lecture seule	RaisonSociale1() As String	Raison sociale 1.
Lecture seule	RaisonSociale2() As String	Raison sociale 2.
Lecture seule	Siret() As String	SIRET.
Lecture seule	Telecopie() As String	Télécopie.
Lecture seule	Telephone() As String	Téléphone.
Lecture seule	Ville() As String	Ville.

ITelecom

Informations de télécommunication.

Propriétés

Accès	Syntaxe	Description
Lecture seule	bGereEmail() As Boolean	Permet de tester si l'objet ITelecom gère la propriété EMail.
Lecture seule	bGerePortable() As Boolean	Permet de tester si l'objet ITelecom gère la propriété Téléphone portable.
Lecture seule	bGereSite() As Boolean	Permet de tester si l'objet ITelecom gère la propriété Site Web.
Lecture seule	GereTelecopie() As Boolean	Permet de tester si l'objet ITelecom gère la propriété Télécopie.
Lecture / Ecriture	EMail() As String	Adresse email.
Lecture / Ecriture	Portable() As String	Téléphone portable.
Lecture / Ecriture	Site() As String	Site Web.
Lecture / Ecriture	Telecopie() As String	Télécopie.
Lecture / Ecriture	Telephone() As String	Téléphone.

Interfaces Application Comptabilité : Objets100c.CPTA

Classe d’application / BSCPTAApplication100c

Accès à la base comptable.

Interfaces héritées

Syntaxe	Description
IBILoggable	Cf. Interface IBILoggable pour les propriétés et méthodes héritées.
IBIPersistStream	Cf. Interface IBIPersistStream pour les propriétés et méthodes héritées.
IBSCPTAApplication100c	Cf. Interface IBSCPTAApplication100c pour les propriétés et méthodes héritées.

Interface Stream / IBSCPTAApplication100c

Base de données comptable.

Interface héritée

Syntaxe	Description
IBIPersistStream	Cf. Interface IBIPersistStream pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Companies() As ICompanies	Liste des bases Sage 100c.
Lecture / Ecriture	CompanyServer() As String	Nom du serveur/instance SQL.
Lecture / Ecriture	CompanyDatabaseName() As String	Nom de la base SQL.
Lecture seule	DatabaseInfo() As IDatabaseInfo	Informations sur la base de données.
Lecture seule	FactoryAnalyse() As IBPAnalyseFactory	Fabrique un objet Niveau d'analyse.
Lecture seule	FactoryAnalytique() As IBPAnalytiqueFactory3	Fabrique un objet Plan analytique.
Lecture seule	FactoryClient() As IBOClientFactory3	Fabrique un objet Client.
Lecture seule	FactoryClientLivraison() As IBOClientLivraisonAllFactory	Accès aux lieux de livraison sans passer par le Client.
Lecture seule	FactoryClientProspect() As IBOClientProspectFactory	Fabrique un objet Client de type prospect ou non.
Lecture seule	FactoryCodeRisque() As IBPCodeRisqueFactory	Fabrique un objet Code risque.

Accès	Syntaxe	Description
Lecture seule	FactoryCollaborateur() As IBOCollaborateurFactory	Fabrique un objet Collaborateur.
Lecture seule	FactoryCompteA() As IBOCompteAFactory3	Fabrique un objet Compte analytique.
Lecture seule	FactoryCompteG() As IBOCompteGFactory3	Fabrique un objet Compte général.
Lecture seule	FactoryCompteR() As IBOCompteRFactory	Fabrique un objet Compte reporting.
Lecture seule	FactoryDevise() As IBPDeviseFactory2	Fabrique un objet Devise.
Lecture seule	FactoryDossier() As IBITypeObjectFactory	Fabrique un objet Dossier entreprise.
Lecture seule	FactoryDossierContact() As IBODossierContactFactory2	Fabrique un objet Contact du dossier.
Lecture seule	FactoryEcriture() As IBOEcritureFactory3	Fabrique un objet Ecriture générale.
Lecture seule	FactoryEcritureA() As IBOEcritureAAllFactory	Accès aux écritures analytiques sans passer par les écritures générales.
Lecture seule	FactoryEcritureOD() As IBOEcritureODFactory	Fabrique un objet écriture d'OD analytique.
Lecture seule	FactoryFournisseur() As IBOFournisseurFactory3	Fabrique un objet Fournisseur.
Lecture seule	FactoryJournal() As IBOJournalFactory3	Fabrique un objet Journal.
Lecture seule	FactoryJournalA() As IBOJournalAFactory	Fabrique un objet code journal analytique.
Lecture seule	FactoryLibelleCpta() As IBITypeObjectFactory	Accès au paramétrage des libellés.
Lecture seule	FactoryModeleEcriture() As IBOModeleEcritureFactory2	Fabrique un objet modèle de saisie d'écriture.
Lecture seule	FactoryModeleGrille() As IBOModeleGrilleFactory	Fabrique un objet modèle de grille.
Lecture seule	FactoryModelReglement() As IBOModeleReglementFactory	Fabrique un objet modèle de règlement.
Lecture seule	FactoryNatureCompte() As IBPNatureCompteFactory	Fabrique un objet nature de compte.
Lecture seule	FactoryPays() As IBOPaysFactory	Fabrique un objet pays.
Lecture seule	FactoryProspect() As IBOProspectFactory	Fabrique un objet Client de type prospect uniquement.
Lecture seule	FactoryRappel() As IBPRappelFactory	Fabrique un objet période de rappel.
Lecture seule	FactoryReglement() As IBPReglementFactory3	Fabrique un objet Mode de règlement.

Accès	Syntaxe	Description
Lecture seule	FactoryServiceContact() As IBPServiceContactFactory	Fabrique un objet Service contact.
Lecture seule	FactoryStructBanque() As IBPStructBanqueFactory	Fabrique un objet Structure banque.
Lecture seule	FactoryTaxe() As IBOTaxeFactory3	Fabrique un objet Taux de taxe.
Lecture seule	FactoryTiers() As IBOTiersFactory3	Fabrique un objet Tiers.
Lecture seule	FactoryTiersAutre() As IBOTiersAutreFactory2	Fabrique un objet Tiers de type Autre.
Lecture seule	FactoryTiersContact() As IBITypeObjectFactory	Accès aux contacts tiers sans passer par les tiers.
Lecture seule	FactoryTiersSalarie() As IBOTiersSalarieFactory2	Fabrique un objet Tiers de type Salarié.
Lecture seule	FactoryTiersStat() As IBPTiersStatFactory	Fabrique un objet statistique tiers.
Lecture seule	FactoryTiersType() As IBPTiersTypeFactory	Fabrique un objet paramétrage tiers.
Lecture seule	FactoryTypeContacts() As IBPContactFactory	Fabrique un objet type de contact.
Lecture seule	Licence() As ILicence	Licence.
Lecture seule	Loggable() As IBILoggable	Autorisations d'accès à la base de données.
Lecture / Ecriture	Name() As String	Chemin et nom de la base de données.
Lecture	FactoryBonAPayer () As IBPBonAPayer	Donne le factory de l'objet IBPBonAPayer

Méthodes

Syntaxe	Description
Close()	Ferme la base comptable.
Create()	Créé un fichier comptable.
CreateProcess_Encoder() As IPMEncoder	Créé un processus permettant la saisie d'une pièce comptable. Cf. IPMEncoder
CreateProcess_Lettrer() As IPMLettrer	Créé un processus de lettrage. Cf. IPMLettrer.
Open()	Ouvre la base comptable.
CreateProcess_BAPValider(as IBOCollaborateur) as IPMBonAPayer	Crée un processus permettant de Valider un Bon à Payer
CreateProcess_BAPAttendre (as IBOCollaborateur) as IPMBonAPayer	Crée un processus permettant de mettre en attente un Bon à Payer
CreateProcess_BAPPayer (as IBOCollaborateur) as IPMBonAPayer	Crée un processus permettant de Payer un Bon à Payer
CreateProcess_BAPConformer (as IBOCollaborateur) as IPMBonAPayer	Crée un processus permettant de mettre Conforme un Bon à Payer

Syntaxe	Description
CreateProcess_BAPRejeter (as IBOCollaborateur) as IPMBonAPayer	Crée un processus permettant de rejeter un Bon à Payer

Interfaces métiers (Ixxx, IBIxxx et IDOCxxx)

IDossierExercice

Paramétrage des exercices accessible depuis la propriété ***Exercice*** de ***IBPDossier2***.

Propriétés

Accès	Syntaxe	Description
Lecture seule	ReportANouveaux() As IDossierExerciceReportANouveaux	Paramétrage du journal des reports à nouveaux.
Lecture seule	ReportANouveauxIFRS() As IDossierExerciceReportANouveauxIFRS	Paramétrage du journal des reports à nouveaux IFRS.

IDossierExerciceReportANouveaux

Paramétrage du journal des reports à nouveaux.

Propriété

Accès	Syntaxe	Description
Lecture / Ecriture	Journal() As IBOJournal3	Journal des reports à nouveaux.

Après affectation du journal, la validation s'effectue à l'appel de **Write()** sur **IBPDossier2**.

IDossierExerciceReportANouveauxIFRS

Paramétrage du journal des reports à nouveaux IFRS.

Propriété

Accès	Syntaxe	Description
Lecture / Ecriture	Journal() As IBOJournal3	Journal des reports à nouveaux IFRS.

Après affectation du journal IFRS, la validation s'effectue à l'appel de **Write()** sur **IBPDossier2**.

IDossierTiers

Paramétrage tiers accessible depuis la propriété **Tiers** de **IBPDossier2**.

Propriété

Accès	Syntaxe	Description
Lecture / Ecriture	EcheanceMax() As Date	Echéance maximum (exprimée en jours).

Après affectation du journal IFRS, la validation s'effectue à l'appel de **Write()** sur **IBPDossier2**.

IRegistreRevision

Régularisation des charges et produits accessible depuis la propriété ***RegistreRevision*** de ***IBOEcriture3***.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Commentaire() As String	Commentaire.
Lecture / Ecriture	Controleur() As String	Contrôleur.
Lecture / Ecriture	DateControle() As Date	Date de contrôle.
Lecture / Ecriture	DateDebut() As Date	Date de début.
Lecture / Ecriture	DateFin() As Date	Date de fin.
Lecture / Ecriture	DateRevision() As Date	Date de révision.

Accès	Syntaxe	Description
Lecture / Ecriture	Reviseur() As String	Réviseur.

Pour créer un regularisation, il faut au **minimum** renseigner les propriétés **DateDebut()** et **DateFin()**.

*Pour la suppression d'une régularisation, il faut mettre à null la propriété **DateDebut()** ou **DateFin()**.*

*La création ou suppression d'une régularisation sur une écriture s'effectue après le **write()** sur l'objet **IBOEcriture3**.*

IBonAPayer Nouveau

Historique du Bon à payer pour une pièce d'écriture

Propriétés

Accès	Syntaxe	Description
Lecture	Statut () As eTypeBAP	Statut du Bon à payer.
Lecture	Responsable() As IBOCollaborateur	Collaborateur du Bon à payer.
Lecture	DateAction () As Date	Date du Bon à payer.
Lecture	Commentaire () As String	Commentaire
Lecture	Application () As integer	Application à l'origine du Bon.
Lecture	Retour () As integer	Retour fournisseur.

Interfaces factory paramètres (IBPxxxFactory)

IBPAnalyseFactory

Fabrique un objet niveau d'analyse.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *sIntitule* As String) As Boolean	Test l'existence de l'objet Niveau d'analyse correspondant à l'intitulé passé en paramètre. Retourne *True* s'il existe sinon *False*.
ReadIntitule(ByVal *sIntitule* As String) As IBPAnalyse	Retourne l'objet Niveau d'analyse correspondant à l'intitulé passé en paramètre.

IBPAnalytiqueFactory3

Fabrique un objet Plan analytique.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *sIntitule* As String) As Boolean	Test l'existence de l'objet Plan analytique correspondant à l'intitulé passé en paramètre. Retourne *True* s'il existe sinon *False*.
ReadIntitule(ByVal *sIntitule* As String) As IBPAnalytique3	Retourne l'objet Plan analytique correspondant à l'intitulé passé en paramètre.

IBPCodeRisqueFactory

Fabrique un objet Code risque.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *sIntitule* As String) As Boolean	Test l'existence de l'objet Code risque correspondant à l'intitulé passé en paramètre.

Syntaxe	Description
	Retourne *True* s'il existe sinon *False*.
ReadIntitule(ByVal *sIntitule* As String) As IBPCodeRisque	Retourne l'objet Code risque correspondant à l'intitulé passé en paramètre.

IBPContactFactory

Fabrique un objet Type de contact.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *sIntitule* As String) As Boolean	Test l'existence de l'objet Type de contact correspondant à l'intitulé passé en paramètre. Retourne *True* s'il existe sinon *False*.
ReadIntitule(ByVal *sIntitule* As String) As IBPContact	Retourne l'objet Type de contact correspondant à l'intitulé passé en paramètre.

IBPDeviseFactory2

Fabrique un objet Devise.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistCodeISO(ByVal *sIntitule* As String) As Boolean	Test l'existence de l'objet Devise correspondant au code ISO passé en paramètre. Retourne *True* s'il existe, sinon *False*.
ReadCodeISO(ByVal *sIntitule* As String) As IBPDevise2	Retourne l'objet Devise correspondant au code ISO passé en paramètre.

Syntaxe	Description
ExistIntitule(ByVal *sIntitule* As String) As Boolean	Test l'existence de l'objet Devise correspondant à l'intitulé passé en paramètre. Retourne *True* s'il existe, sinon *False*.
ReadIntitule(ByVal *sIntitule* As String) As IBPDevise2	Retourne l'objet Devise correspondant à l'intitulé passé en paramètre.

IBPRappelFactory

Fabrique un objet période de rappel.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *sIntitule* As String) As Boolean	Test l'existence de l'objet période de rappel correspondant à l'intitulé passé en paramètre.
ReadIntitule(ByVal *sIntitule* As String) As IBPRappel	Retourne l'objet période de rappel correspondant à l'intitulé passé en paramètre.

IBPReglementFactory3

Fabrique un objet Mode de règlement.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *sIntitule* As String) As Boolean	Test l'existence de l'objet Mode de règlement correspondant à l'intitulé passé en paramètre.
ReadIntitule(ByVal *sIntitule* As String) As IBPReglement3	Retourne l'objet Mode de règlement correspondant à l'intitulé passé en paramètre.

IBPServiceContactFactory

Fabrique un objet Service contact.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *sIntitule* As String) As Boolean	Test l'existence de l'objet Service contact correspondant à l'intitulé passé en paramètre.
ReadIntitule(ByVal *sIntitule* As String) As IBPServiceContact	Retourne l'objet Service contact correspondant à l'intitulé passé en paramètre.

IBPStructBanqueFactory

Fabrique un objet Structure banque.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthode

Syntaxe	Description
ReadType(ByVal *sType* As RibType) As IBPStructureBanque	Test l'existence de l'objet Structure banque correspondant au type passé en paramètre.

IBPTiersStatFactory

Fabrique un objet statistique tiers.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *sIntitule* As String) As Boolean	Test l'existence de l'objet statistique tiers correspondant à l'intitulé passé en paramètre.
ReadIntitule(ByVal *sIntitule* As String) As IBPTiersStat	Retourne l'objet statistique tiers correspondant à l'intitulé passé en paramètre.

IBPTiersFactory

Fabrique un objet paramétrage type tiers.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthode

Syntaxe	Description
ReadTypeTiers(ByVal *sType* As TypeTiers) As IBPTiers	Retourne l'objet paramétrage type tiers correspondant au type passé en paramètre.

IBPBonAPayer Nouveau

Paramètres de comptabilisation – Gestion du Bon à payer (Paramètres société / Comptabilisation)

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Autorisation () As eBAPAutorisationType	Niveau de validation.
Lecture / Ecriture	Responsable() As IBOCollaborateur	Responsable financier.
Lecture / Ecriture	Facture () As eBAPAValider	Facture à Valider
Lecture / Ecriture	SeuilValidation () As double	Seuil de validation.

Interfaces factory métiers (IBOxxxFactory)

IBOBanqueFactory

Fabrique un objet Banque.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistAbrege(ByVal *sAbrege* As String) As Boolean	Test l'existence d'une banque en fonction de l'abrégé passé en paramètre. Retourne *True* s'il existe, sinon *False*.
ReadAbrege(ByVal *sAbrege* As String) As IBOBanque	Retourne un objet Banque correspondant à l'abrégé passé en paramètre.

IBOBanqueRibFactory

Fabrique un objet RIB Banque.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistAbrege(ByVal *sAbrege* As String) As Boolean	Test l'existence d'un RIB banque en fonction de l'abrégé passé en paramètre. Retourne *True* s'il existe, sinon *False*.
ReadAbrege(ByVal *sAbrege* As String) As IBOBanque	Retourne un objet RIB banque correspondant à l'abrégé passé en paramètre.

IBOClientFactory3

Fabrique un objet Client.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Propriété

Accès	Syntaxe	Description
Lecture seule	ListOrderClassement() As IBICollection	Renvoi une collection d'objets Client ordonnés en fonction de leur champ Classement.

Méthodes

Syntaxe	Description
ExistNumero(ByVal *sNum* As String) As Boolean	Test l'existence d'un client en fonction de son numéro. Retourne *True* s'il existe, sinon *False*.
ReadNumero(ByVal *sNum* As String) As IBOClient3	Retourne un objet Client correspondant au numéro passé en paramètre.

IBOClientLivraisonAllFactory

Permet d'accéder à une collection de lieux de livraison sans passer par le client.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthode

Syntaxe	Description
QueryClient(ByRef cClient As IBOClient3) As IBICollection	Collection des lieux de livraison associés au client passé en paramètre.

IBOClientProspectFactory

Fabrique un objet Client de type prospect ou non.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Propriété

Accès	Syntaxe	Description
Lecture seule	ListOrderClassement() As IBICollection	Collection d'objets Client (de type prospect ou non) ordonnés en fonction de leur propriété *Classement*.

Méthodes

Syntaxe	Description
ExistNumero(ByVal *sNum* As String) As Boolean	Teste l'existence d'un client (de type prospect ou non) correspondant au numéro de compte passé en paramètre. Retourne *True* s'il existe, sinon retourne *False*.
ReadNumero(ByVal *sNum* As String) As IBOClient3	Retourne un objet Client (de type prospect ou non) correspondant au numéro de compte passé en paramètre.

IBOClientTarifFactory3

Fabrique un objet Tarif client.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistArticle(ByVal *pArticle* As IBOArticle2) As Boolean	Test l'existence d'un tarif client correspondant à l'article passé en paramètre. Retourne *True* si un tarif client existe, sinon *False*.
ReadArticle(ByVal *pArticle* As IBOArticle2) As IBOArticleTarifClient3	Retourne un objet Tarif client correspondant à l'article passé en paramètre.

IBOCompteAFactory3

Fabrique un compte analytique.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Propriété

Accès	Syntaxe	Description
Lecture seule	InfoLibreFields() As IBIFields	Collection de champs (informations libres).

Méthodes

Syntaxe	Description
ExistNumero(ByVal *pAnalytique* As IBPAnalytique3, ByVal *sNum* As String) As Boolean	Test l'existence d'un compte analytique correspondant au numéro de section analytique et au plan analytique passés en paramètres. Retourne *True* s'il existe, sinon retourne *False*.
QueryPlanAna(ByVal *pAnalytique* As IBPAnalytique3) As IBICollection	Retourne une collection de comptes analytiques en fonction d'un plan analytique passé en paramètre.
ReadNumero(ByVal *pAnalytique* As IBPAnalytique2, ByVal *sNum* As String) As IBOCompteA3	Retourne un compte analytique correspondant au numéro de section analytique et au plan analytique passés en paramètres.

IBOCompteGFactory3

Fabrique un compte général.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	InfoLibreFields() As IBIFields	Collection de champs (informations libres).
Lecture seule	ListOrderClassement() As IBICollection	Retourne une collection d'objets Comptes généraux classés par Classement.
Lecture seule	ListOrderType() As IBICollection	Retourne une collection d'objets Comptes généraux classés par Type.

Méthodes

Syntaxe	Description
ExistNumero(ByVal *sNum* As String) As Boolean	Test l'existence du compte général correspondant au numéro de compte général passé en paramètre. Retourne *True* s'il existe, sinon *False*.
ReadNumero(ByVal *sNum* As String) As IBOCompteG3	Retourne un compte général correspondant au numéro de compte général passé en paramètre.
QueryActifOrderClassement() As IBICollection	Retourne une collection de comptes généraux actifs triés par classement.
QueryActifOrderNumero() As IBICollection	Retourne une collection de comptes généraux actifs triés par numéro.
QueryActifOrderType() As IBICollection	Retourne une collection de comptes généraux actifs triés par type.
QueryTypeNumeroOrderNumero(ByVal *sType* As CompteGType, ByVal *sNumDe* As String, ByVal *sNumA* As String) As IBICollection	Retourne une collection de comptes généraux de type *sType* (Cf. énumérateur CompteGType), compris dans la plage de numéro de compte de *sNumDe* à *sNumA*.
QueryCompteR(ByRef *CompteR* As IBOCompteR) As IBICollection	Retourne une collection de comptes généraux associés au compte reporting passé en paramètre.

IBOCompteGTiersFactory3

Fabrique un tiers rattaché au compte général.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthode

Syntaxe	Description
AddTiers(ByVal *pTiers* AsIBOTiers3)	Rattache un tiers (passé en paramètre) au compte général.

IBOCompteRFactory

Fabrique un objet compte reporting.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ListOrderClassement() As IBICollection	Collection de comptes reportings triée par classement.
ListOrderType() As IBICollection	Collection de comptes reprotings triée par type.
ExistNumero(ByVal *sNum* As String) As Boolean	Teste l'existence du compte reporting passé en paramètre.
ReadNumero(ByVal *sNum* As String) As IBOCompteR	Retourne l'objet compte reporting correspondant au numéro passé en paramètre.

IBODossierContactFactory2

Fabrique un objet contact du dossier.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistNomPrenom(ByVal *sNom* As String, ByVal *sPrenom* As String) As Boolean	Test l'existence d'un objet contact du dossier correspondant au nom et prénom passés en paramètres. Retourne *True* s'il existe, sinon *False*.
ReadNomPrenom(ByVal *sNom* As String, ByVal *sPrenom* As String) As IBODossierContact3	Retourne l'objet contact du dossier correspondant au nom et prénom passés en paramètres.

IBOEcritureFactory3

Fabrique un objet Ecriture.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Propriété

Accès	Syntaxe	Description
Lecture seule	InfoLibreFields() As IBIFields	Collection de champs (Informations libres).

Méthodes

Syntaxe	Description
ExistNumero(ByVal *EC_No* As Long) As Boolean	Teste l'existence de l'écriture générale en fonction du numéro interne d'écriture passé en paramètre.
QueryCompteG(*CompteG* As IBOCompteG3, ByVal *DateDebut* As Date, ByVal *DateFin* As Date) As IBOCollection	Renvoi une collection d'objets Ecriture pour le compte général et la période passés en paramètre.
QueryJournalPeriode(ByVal *Journal* As IBOJournal3, ByVal *DatePeriode* As Date) As IBICollection	Renvoi une collection d'objets Ecriture correspondant à un journal et à une période particuliers.
QueryTiers(ByVal *Tiers* As IBOTiers3, ByVal *CompteG* As IBOCompteG3, ByVal *Journal* As IBOJournal3, ByVal *DateDebut* As Date, ByVal *DateFin* As Date) As IBICollection	Renvoi une collection d'objets Ecriture correspondant à un tiers, un compte général, un journal et une période particuliers.
ReadNumero(ByVal *EC_No* As Long) As IBOEcriture3	Retourne une écriture générale en fonction du numéro interne d'écriture passé en paramètre.

IBOEcritureAFactory2

Fabrique un objet Ecriture analytique.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
QueryPlanA(*pAnalytique* As IBPAnalytique3) As IBICollection	Retourne une collection d'écritures analytique correspondant au plan analytique passé en paramètre.
QuerySectionA(*CompteA* As IBOCompteA3) As IBICollection	Retourne une collection d'écritures analytique correspondant à la section analytique passée en paramètre.

IBOEcritureAAllFactory

Permet d'accéder à une collection d'écritures analytiques sans passer par les écritures générales.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
QueryPlanA(*pAnalytique* As IBPAnalytique3) As IBICollection	Collection des écritures analytiques associées au plan analytique passé en paramètre.
QuerySectionA(*CompteA* As IBOCompteA) As IBICollection	Collection des écritures analytiques associées à la section analytique passée en paramètre.

IBOEcritureODFactory

Fabrique un objet Ecriture d'OD analytique.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
QueryJournal(*Journal* As IBOJournalA, *DateDe* As Date, *DateA* As Date) As IBICollection	Retourne une collection d'écritures d'od analytique correspondant au journal analytique et la période passés en paramètres.
QueryCompte(*CompteG* As IBOCompteG3, *CompteA* As IBOCompteA3) As IBICollection	Retourne une collection d'écritures d'od analytique correspondant au compte général et à la section analytique passés en paramètres

Syntaxe	Description
	Remarque : les paramètres *CompteG* et *CompteA* sont optionnels mais pas tous les deux en même temps.

IBOFournisseurFactory3

Fabrique un objet Fournisseur.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Propriété

Accès	Syntaxe	Description
Lecture seule	ListOrderClassement() As IBICollection	Retourne une collection d'objets Fournisseur ordonnés en fonction du champ Classement.

Méthodes

Syntaxe	Description
ExistNumero(ByVal *sNum* As String) As Boolean	Test l'existence d'un objet Fournisseur correspondant au numéro passé en paramètre. Retourne *True* s'il existe, sinon *False*.
ReadNumero(ByVal *sNum* As String) As IBOFournisseur3	Retourne un objet Fournisseur correspondant au numéro passé en paramètre.

IBOFournisseurTarifFactory3

Fabrique un objet Tarif fournisseur.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistArticle(ByVal *pArticle* As IBOArticle3) As Boolean	Test l'existence d'un objet Tarif fournisseur correspondant à l'objet Article passé en paramètre. Retourne *True* s'il existe, sinon *False*.
ReadArticle(ByVal *pArticle* As IBOArticle3) As IBOArticleTarifFournisseur3	Retourne un objet Tarif fournisseur correspondant à l'objet Article passé en paramètre.

IBOJournalFactory3

Fabrique un objet Journal.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistNumero(ByVal *sNum* As String) As Boolean	Test l'existence d'un objet Journal correspondant au numéro passé en paramètre. Retourne *True* s'il existe, sinon *False*.
ReadNumero(ByVal *sNum* As String) As IBOJournal3	Retourne l' objet Journal correspondant au numéro passé en paramètre.

IBOJournalAFactory

Fabrique un objet code Journal analytique.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistNumero(ByVal *sNum* As String) As Boolean	Test l'existence d'un objet Journal analytique correspondant au numéro passé en paramètre.

Syntaxe	Description
	Retourne *True* s'il existe, sinon *False*.
ReadNumero(ByVal *sNum* As String) As IBOJournalA	Retourne l' objet Journal analytique correspondant au numéro passé en paramètre.

IBOModeleEcritureFactory2

Fabrique un objet modèle de saisie.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *sIntitule* As String) As Boolean	Test l'existence d'un objet modèle de saisie correspondant à l'intitulé passé en paramètre. Retourne *True* s'il existe, sinon *False*.
ReadIntitule(ByVal *sIntitule* As String) As IBOModeleEcriture3	Retourne l'objet modèle de saisie correspondant à l'intitulé passé en paramètre.

IBOModeleFactory

Fabrique un objet modèle d'enregistrement.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *sIntitule* As String) As Boolean	Test l'existence d'un objet modèle d'enregistrement correspondant à l'intitulé passé en paramètre.

Syntaxe	Description
	Retourne *True* s'il existe, sinon *False*.
ReadIntitule(ByVal *sIntitule* As String) As IBOModele2	Retourne l'objet modèle d'enregistrement correspondant à l'intitulé passé en paramètre.

IBOModeleGrilleFactory

Fabrique un objet modèle de grille.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *sIntitule* As String) As Boolean	Test l'existence d'un objet modèle de grille correspondant à l'intitulé passé en paramètre. Retourne *True* s'il existe, sinon *False*.
ReadIntitule(ByVal *sIntitule* As String) As IBOModeleGrille	Retourne l'objet modèle de grille correspondant à l'intitulé passé en paramètre.

IBOModeleGrilleLigneFactory

Fabrique un objet ligne de modèle de grille.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthode

Syntaxe	Description
QueryPlanA(*pAnalytique* As IBPAnalytique3) As IBICollection	Retourne une collection de lignes de modèle de grille correspondant au plan analytique passé en paramètre.

IBOModeleReglementFactory

Fabrique un objet modèle de règlement.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *sIntitule* As String) As Boolean	Test l'existence d'un objet modèle de règlement correspondant à l'intitulé passé en paramètre. Retourne *True* s'il existe, sinon *False*.
ReadIntitule(ByVal *sIntitule* As String) As IBOModeleReglement	Retourne l'objet modèle de règlement correspondant à l'intitulé passé en paramètre.

IBOPaysFactory

Fabrique un objet pays.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *sIntitule* As String) As Boolean	Test l'existence d'un pays correspondant à l'intitulé passé en paramètre. Retourne *True* s'il existe, sinon *False*.
ReadIntitule(ByVal *sIntitule* As String) As IBOPays	Retourne l'objet pays correspondant à l'intitulé passé en paramètre.

IBOProspectFactory

Fabrique un objet Client de type prospect uniquement.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Propriété

Accès	Syntaxe	Description
Lecture seule	ListOrderClassement() As IBICollection	Collection d'objets Client (de type prospect uniquement) ordonnés en fonction de leur propriété *Classement*.

Méthodes

Syntaxe	Description
ExistNumero(ByVal *sNum* As String) As Boolean	Teste l'existence d'un client (de type prospect uniquement) correspondant au numéro de compte passé en paramètre. Retourne *True* s'il existe, sinon retourne *False*.
ReadNumero(ByVal *sNum* As String) As IBOClient3	Retourne un objet Client (de type prospect uniquement) correspondant au numéro de compte passé en paramètre.

La création d'un client de type prospect ne peut être réalisée qu'en utilisant ce Factory.

IBOTaxeFactory3

Fabrique un objet Taxe.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistCode(ByVal *sNum* As String) As Boolean	Test l'existence d'un objet Taxe correspondant au code passé en paramètre. Retourne *True* s'il existe, sinon *False*.
ReadNumero(ByVal *sNum* As String) As IBOTaxe3	Retourne l'objet Taxe correspondant au code passé en paramètre.

IBOTiersCompteGFactory3

Fabrique un objet Compte général tiers.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthode

Syntaxe	Description
AddCompteG(ByVal *pCompteG* As IBOCompteG3)	Ajoute un compte général rattaché au compte tiers.

IBOTiersFactory3

Fabrique un objet Tiers.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	InfoLibreFields() As IBIFields	Collection de champs (informations libres).
Lecture seule	ListOrderClassement() As IBICollection	Retourne une collection de tiers ordonnée en fonction du champ Classement.
Lecture seule	ListOrderNumero() As IBICollection	Retourne une collection de tiers ordonnée en fonction du champ Numéro.
Lecture seule	ListOrderType() As IBICollection	Retourne une collection de tiers ordonnée en fonction du champ Type.

Méthodes

Syntaxe	Description
ExistNumero(ByVal *sNum* As String) As Boolean	Test l'existence d' un objet tiers correspondant au Numéro passé en paramètre. Retourne *True* s'il existe, sinon *False*.
QueryActifOrderNumero() As IBICollection	Retourne une collection d'objets tiers actifs (non mis en sommeil) ordonnée en fonction du champ Numéro.
QueryCollaborateur(ByVal *pObj* As IBOCollaborateur) As IBICollection	Retourne une collection de tiers affectés au collaborateur passé en paramère.
QueryMouvementeOrderNumero() As IBICollection	Retourne une collection d'objets tiers mouvementés ordonnée en fonction du champ Numéro.
QueryReplicate(ByVal *lIndice* As Integer) As IBICollection	Retourne une collection d'objets tiers dont le nombre de réplication correspond à *lIndice* passé en paramètre.
QueryTypeNumeroOrderNumero(ByVal *sType* As TiersType, ByVal *sNumDe* As String, ByVal *sNumA* As String) As IBICollection	Retourne une collection d'objets tiers d'une plage de tiers donnée (de *sNumDe* à *sNumA*) et d'un type *sType* donné (Cf. énumérateur *TiersType*).
ReadNumero(ByVal *sNum* As String) As IBOTiers3	Retourne un objet tiers correspondant au Numéro passé en paramètre.

IBOTiersAutreFactory2

Fabrique un objet tiers de type Autre.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistNumero(ByVal *sNum* As String) As Boolean	Test l'existence d'un objet tiers de type Autre correspondant au Numéro passé en paramètre. Retourne *True* s'il existe, sinon *False*.
ReadNumero(ByVal *sNum* As String) As IBOTiers3	Retourne l'objet tiers de type Autre correspondant au Numéro passé en paramètre.

IBOTiersSalarieFactory2

Fabrique un objet tiers de type Salarié.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistNumero(ByVal *sNum* As String) As Boolean	Test l'existence d' un objet tiers de type Salarié correspondant au Numéro passé en paramètre. Retourne *True* s'il existe, sinon *False*.
ReadNumero(ByVal *sNum* As String) As IBOTiers3	Retourne l'objet tiers de type Salarié correspondant au Numéro passé en paramètre.

Interfaces objets paramètres (IBPxxx)

Correspondance des objets

Ce tableau permet de faire la correspondance entre les objets et les fonctions et tables des applications Sage 100c.

Type objet	Description	Correspondance dans l'application	Table
IBPAnalyse	Niveau d'analyse	Compatiblité – Gestion commerciale / Menu Fichier / Paramètres société / Analytique / Niveau d'analyse	P_ANALYSE
IBPAnalytique3	Plan analytique	Compatiblité – Gestion commerciale / Menu Fichier / Paramètres société / Analytique / Plan analytique	P_ANALYTIQUE
IBPAnalytiqueRupture	Paramétrage des ruptures d'un plan analytique structuré	Compatiblité – Gestion commerciale / Menu Fichier / Paramètres société / Analytique / Plan analytique / Bouton – Structurer le plan analytique	P_ANALYTIQUE
IBPContact	Type de contact	Comptabilité – Gestion commerciale / Menu	P_CONTACT

Type objet	Description	Correspondance dans l'application	Table
		Fichier / Paramètres société /Contacts / Type	
IBPCodeRisque	Code risque	Comptabilité – Gestion commerciale / Menu Fichier / Paramètres société / Tiers / Code risque	P_CRISQUE
IBPDevise2	Devise	Comptabilité – Gestion commerciale / Menu Fichier / Paramètres société / International / Devise	P_DEVISE
IBPDossier2	Dossier	Comptabilité – Gestion commerciale / Menu Fichier / Paramètres société / Identification	P_DOSSIER
IBPLibelleCpta	Paramétrage des libellés	Comptabilité – Gestion commerciale / Menu Fichier / Paramètres société / Libellé	P_LIBELLECPTA
IBPRappel	Période de rappel	Comptabilité / Menu Fichier / Paramètres société / Rappel Recouvrement / Rappel	P_RAPPEL
IBPReglement3	Mode de règlement	Comptabilité / Menu Fichier / Paramètres société / Tiers / Mode de règlement	P_REGLEMENT
IBPServiceContact	Service contact	Comptabilité – Gestion commerciale/ Menu Fichier / Paramètres société / Contacts / Service	P_SERVICECPTA
IBPStructBanque	Structure des banques	Comptabilité – Gestion commerciale / Menu Fichier / Paramètres société / Banques / Structure compte	P_BANQUESTRUCT
IBPTiers	Paramétrage type tiers	Comptabilité – Gestion commerciale / Menu Fichier / Paramètres société / Tiers	P_TIERS
IBPTiersStat	Champ statistique tiers	Comptabilité – Gestion commerciale / Menu Fichier / Paramètres société / Tiers / Champ statistique	P_STATISTIQUE

Liens entre les objets

Le tableau ci-dessous permet pour chaque type d'objet, de retrouver la propriété *Factory* à partir de laquelle l'objet peut être fabriqué. Ces propriétés sont toutes issues de l'objet maître *BSCPTAApplication100c.*

Exemple :

Type d'objet ***IBPAnalyse*** *:*

- *L'objet* ***IBPAnalyse*** *(Niveau d'analyse) est issu d'un objet maître de type* ***BSCPTAApplication100c*** *;*
- *L'objet maître fabrique un objet* ***IBPAnalyse*** *par appel de la propriété* ***FactoryAnalyse()****.*

Interface objet	Propriété factory objet	Description objet
IBPAnalyse	FactoryAnalyse()	Niveau d'analyse
IBPAnalytique3	FactoryAnalytique()	Plan analytique
IBPCodeRisque	FactoryCodeRisque()	Code risque
IBPDevise2	FactoryDevise()	Devise
IBPDossier2	FactoryDossier()	Dossier entreprise
IBPRappel	FactoryRappel()	Période de rappel
IBPReglement3	FactoryReglement()	Mode de règlement
IBPServiceContact	FactoryServiceContact()	Service contact
IBPTiers	FactoryTiersType()	Paramétrage type tiers
IBPTiersStat	FactoryTiersStat()	Champ statistique tiers
IBPContact	FactoryTypeContacts	Type de contact

IBPAnalyse

Niveau d'analyse.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	A_Intitule() As String	Intitulé.

Accès	Syntaxe	Description
Lecture seule	FactoryAnalyse() As IBPAnalyseFactory	Fabrique un objet Niveau d'analyse.

IBPAnalytique3

Plan analytique.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	A_Intitule() As String	Intitulé.
Lecture / Ecriture	A_Obligatoire() As Boolean	Plan obligatoire (true) sinon false.
Lecture seule	FactoryAnalytique() As IBPAnalytiqueFactory3	Fabrique un objet Plan analytique.
Lecture seule	Rupture(ByVal sElt As Short) As IPBAnalytiqueRupture	Paramétrage rupture analytique pour l'indice de rupture passé en paramètre (sElt entre 1 et 6).
Lecture seule	IsStructured() As Boolean	Indique si le plan est un plan analytique structuré.
Lecture / Ecriture	SectionAttente() As IBOCompteA3	Section analytique d'attente.

IBPAnalytiqueRupture

Paramétrage des ruptures d'un plan analytique structuré.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Intitule() As String	Intitulé de la rupture.
Lecture / Ecriture	Lg() As short	Longueur de la rupture.

Accès	Syntaxe	Description
Lecture / Ecriture	Type() As AnalytiqueRuptureType	Type de la rupture (utilisé uniquement pour le plan analytique IFRS).

IBPCodeRisque

Code risque.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	FactoryCodeRisque() As IBPCodeRisqueFactory	Fabrique un objet Code risque.
Lecture / Ecriture	R_Intitule() As String	Intitulé.
Lecture / Ecriture	R_Max() As Double	Borne maximale du dépassement d'encours.
Lecture / Ecriture	R_Min() As Double	Borne minimale du dépassement d'encours.
Lecture / Ecriture	R_Type() As RisqueType	Action (cf. énumérateur RisqueType).

IBPContact

Type de contact.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	FactoryTypeContact() As IBPContactFactory	Fabrique un objet Type de contact.
Lecture / Ecriture	Intitule() As String	Intitulé.

IBPDevise2

Devise.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	D_AncCours() As Double	Cours ancienne cotation.
Lecture / Ecriture	D_AncDate() As Date	Date limite ancienne cotation.
Lecture / Ecriture	D_AncMode() As DeviseMode	Mode ancienne cotation (cf. énumérateur DeviseMode).
Lecture / Ecriture	D_CodeISO() As String	Code ISO.
Lecture / Ecriture	D_CodeISONum()	Numéro code ISO.
Lecture / Ecriture	D_CodeRemise() As DeviseRemise	Code de remise (cf. énumérateur DeviseRemise).
Lecture / Ecriture	D_Cours() As Double	Cours.
Lecture / Ecriture	D_CoursClot() As Double	Cours clôture.
Lecture / Ecriture	D_CoursP() As Double	Cours période.
Lecture / Ecriture	D_Euro() As Boolean	Zone Euro : **True** : Oui **False** : Non
Lecture / Ecriture	D_Format() As String	Format.
Lecture / Ecriture	D_Intitule() As String	Intitulé.
Lecture / Ecriture	D_Mode() As DeviseMode	Mode de cotation (cf. énumérateur DeviseMode).
Lecture / Ecriture	D_Monnaie() As String	Unité monétaire.
Lecture / Ecriture	D_Sigle() As String	Sigle monnaie.
Lecture / Ecriture	D_SousMonnaie() As String	Sous unité monétaire.
Lecture / Ecriture	DeviseAncCot() As IBPDevise2	Devise ancienne cotation.
Lecture / Ecriture	DeviseCot() As IBPDevise2	Devise cotation.
Lecture seule	FactoryDevise() As IBPDeviseFactory2	Fabrique un objet Devise.

Méthode

Syntaxe	Description
Convert(ByVal *pDevise* As IBPDevise2, ByVal *dVal* As Double) As Double	Converti la valeur *dVal* en devise *pDevise*.

IBPDossier2

Dossier entreprise.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Adresse() As IAdresse	Adresse.
Lecture / Ecriture	AnalytiqueIFRS() As IBPAnalytique3	Plan analytique de type IFRS.
Lecture / Ecriture	D_Ape() As String	Code APE (NAF).
Lecture seule	D_ArchivPeriod() As Date	Date d'archivage
Lecture seule	D_CloturePeriod() As Date	Date de la dernière période clôturée.
Lecture / Ecriture	D_Commentaire() As String	Commentaire.
Lecture / Ecriture	D_DebutExo(ByVal *sElt* As Short) As Date	Début du Nieme exercice.
Lecture / Ecriture	D_EMail() As String	E-mail client.
Lecture / Ecriture	D_EMailExpert() As String	E-mail Expert-Comptable.
Lecture / Ecriture	D_Expert() As String	Intitulé expert.
Lecture / Ecriture	D_FinExo(ByVal *sElt* As Short) As Date	Fin du Nieme exercice.
Lecture / Ecriture	D_FormatQtes() As String	Format quantité.
Lecture / Ecriture	D_Identifiant() As String	N° d'identifiant.

Accès	Syntaxe	Description
Lecture / Ecriture	D_LgAn() As Short	Longueur compte analytique.
Lecture / Ecriture	D_LgCg() As Short	Longueur compte général et reporting.
Lecture / Ecriture	D_NumDossier() As String	N° de dossier expert-comptable.
Lecture / Ecriture	D_Profession() As String	Activité.
Lecture / Ecriture	D_RaisonSoc() As String	Raison sociale.
Lecture / Ecriture	D_Siret() As String	SIRET.
Lecture / Ecriture	DeviseCompte() As IBPDevise2	Devise de tenue.
Lecture / Ecriture	DeviseEquivalence() As IBPDevise2	Devise d'équivalence.
Lecture seule	Exercice() As IDossierExercice	Accès au paramétrage exercice.
Lecture seule	IsExoCloture(ByVal *sElt* As Short) As Boolean	Etat de clôture pour l'indice de l'exercice passé en paramètre (de 1 à 5).
Lecture seule	IsInterditSommeil() As Boolean	Interdire l'utilisation des éléments mis en sommeil.
Lecture / Ecriture	Telecom() As ITelecom	Télécommunications.
Lecture seule	Tiers() As IDossierTiers	Accès au paramétrage tiers.
Lecture seule	FactoryDossier() As IBITypeObjectFactory	Propriété réservée.
Lecture / Ecriture	D_Facebook() As String	Compte Facebook.
Lecture / Ecriture	D_LinkedIn() As String	Compte LinkedIn.

IBPLibelleCpta

Paramétrage des libellés.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	LangueStr(ByVal *sLangueType* As LangueType) As String	Libellé associé à la langue passée en paramètre.

IBPNatureCompte

Nature de compte.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	FactoryNatureCompte() As IBPNatureCompteFactory	Fabrique un objet Nature de compte.
Lecture seule	FactoryNatureCompteDet() As IBPNatureCompteDetFactory	Fabrique un objet Détail de nature de compte.
Lecture seule	NatureCompteType() As NatureCompteType	Type de nature de compte (cf. énumérateur NatureCompteType).
Lecture seule	Intitule() As String	Intitulé de la nature de compte.

IBPNatureCompteDet

Détail de nature de compte.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	FactoryNatureCompte() As IBPNatureCompteFactory	Fabrique un objet Nature de compte.
Lecture / Ecriture	FourchetteDebut() As String	Début de fourchette de compte.

Accès	Syntaxe	Description
Lecture / Ecriture	FourchetteFin() As String	Fin de fourchette de compte.
Lecture / Ecriture	CG_Regroup() As Boolean	Détermine l'application de l'option de regroupement.
Lecture / Ecriture	CG_Analytique() As Boolean	Détermine l'application de l'option saisie analytique.
Lecture / Ecriture	CG_Devise() As Boolean	Détermine l'application de l'option devise.
Lecture / Ecriture	Devise () As IBPDevise	Devise.
Lecture / Ecriture	CG_Echeance() As Boolean	Détermine l'application de l'option de saisie de l'échéance.
Lecture / Ecriture	CG_Quantite() As Boolean	Détermine l'application de l'option de saisie de la quantité.
Lecture / Ecriture	CG_Lettrage() As Boolean	Détermine l'application de l'option de lettrage automatique.
Lecture / Ecriture	CG_Tiers() As Boolean	Détermine l'application de l'option de saisie compte tiers.
Lecture / Ecriture	CG_Report() As CompteGReportType	Type de report à nouveau (cf énumérateur CompteGReportType).

IBPRappel

Période de rappel.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	FactoryRappel() As IBPRappelFactory	Fabrique un objet période de rappel.
Lecture / Ecriture	Intitule() As String	Intitulé de la période de rappel.
Lecture / Ecriture	R_Debut() As Integer	Nombre de jours de retard : jour de début.
Lecture / Ecriture	R_Fin() As Integer	Nombre de jours de retard : jour de fin.
Lecture / Ecriture	R_NbJour() As Integer	Nombre de jours entre deux rappels.

IBPReglement3

Mode de règlement.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	FactoryReglement() As IBPReglementFactory3	Fabrique un objet Mode de règlement.
Lecture / Ecriture	JournalClient() As IBOJournal3	Code journal règlements clients
Lecture / Ecriture	JournalFournisseur() As IBOJournal3	Code journal règlements fournisseurs
Lecture / Ecriture	R_Code() As String	Code règlement.
Lecture / Ecriture	R_Intitule() As String	Intitulé.
Lecture / Ecriture	R_Type() As ReglementType	Type de règlement (Cf. énumérateur ReglementType). *

** : nécessite l'ouverture d'une base de Gestion Commerciale*

IBPServiceContact

Service contact.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	FactoryServiceContact() As IBPServiceContactFactory	Fabrique un objet Service contact.
Lecture / Ecriture	S_Abrege() As String	Abrégé.
Lecture / Ecriture	S_Intitule() As String	Intitulé.

IBPStructBanque

Structure des banques.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	B_Controle() As StructBanqueControleRib	Contrôle clé (cf. énumérateur StructBanqueControleRib).
Lecture / Ecriture	B_Edi() As Boolean	Structure EDI.
Lecture / Ecriture	B_LBanque() As Integer	Longueur code banque.
Lecture / Ecriture	B_LCle() As Integer	Longueur clé.
Lecture / Ecriture	B_LCompte() As Integer	Longueur compte.
Lecture / Ecriture	B_LGuichet() As Integer	Longueur code guichet.
Lecture / Ecriture	B_TBanque() As StructBanqueFieldType	Type code banque (cf. énumérateur StructBanqueFieldType).
Lecture / Ecriture	B_TCle() As StructBanqueFieldType	Type clé (cf. énumérateur StructbanqueFieldType).
Lecture / Ecriture	B_TCompte() As StructBanqueFieldType	Type compte (cf. énumérateur StructbanqueFieldType).
Lecture / Ecriture	B_TGuichet() As StructBanqueFieldType	Type guichet cf. énumérateur StructBanqueFieldType).
Lecture seule	FactoryStructBanque() As IBPStuctBanqueFactory	Fabrique un objet structure banque.
Lecture seule	Type() As RibType	Type de RIB.

IBPTiers

Paramétrage type tiers.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	CT_Type() As TypeTiers	Type de tiers.
Lecture seule	FactoryTiersType() As IBPTiersFactory	Fabrique un objet paramétrage type tiers.

Accès	Syntaxe	Description
Lecture seule	NextCt_Num() As String	Retourne le numéro de compte tiers calculé.
Lecture / Ecriture	T_Compte(ByVal sElt As Integer) As String	Compte général par défaut pour l'indice passé en paramètre.
Lecture / Ecriture	T_Intitule(ByVal sElt As Integer) As String	Intitulé du compte général par défaut pour l'indice passé en paramètre.
Lecture / Ecriture	T_Lg() As Integer	Longueur de la racine de compte.
Lecture / Ecriture	T_Numerotation() As TypeTiersNumerotation	Type de numérotation (cf. énumérateur TypeTiersNumerotation).
Lecture / Ecriture	T_Racine() As String	Racine de compte pour la numérotation.
Lecture / Ecriture	T_TCompte(ByVal sElt As Integer) As TypeTiersCompteType	Type de compte (compte ou radical).

IBPTiersStat

Champ statistique tiers.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Enums() As IBIValuesInsertable	Fabrique un objet champ statistique tiers.
Lecture seule	FactoryTiersStat() As IBPTiersStatFactory	Fabrique un objet champ statistique tiers.
Lecture / Ecriture	Intitule() As String	Intitulé du champ statistique.

Interfaces objets métiers (IBOxxx)

Correspondance des objets

Ce tableau permet de faire la correspondance entre les objets et les fonctions et tables des applications Sage 100c pour SQL Server.

Objet	Description	Correspondance dans l'application	Table correspondante
IBOBanque	Banque	Comptabilité / Menu Structure / Banques Gestion Commerciale / Menu Structure / Comptabilité / Banques	F_BANQUE
IBOBanqueRib	RIB Banque	Comptabilité / Menu Structure / Banques / Onglet Comptes bancaires Gestion Commerciale / Menu Structure / Comptabilité / Banques / Onglet Comptes bancaires	F_EBANQUE
IBOClient3	Client	Comptabilité / Menu Structure / Plan tiers / Fenêtre Tiers Gestion Commerciale / Menu Structure / Clients / Fenêtre Client / Prospect	F_COMPTET
IBOClientLivraison3	Lieu de livraison d'un client	Gestion Commerciale / Menu Structure / Clients / Fenêtre Client / Onglet Livraisons	F_LIVRAISON
IBOCompteA3	Compte analytique	Comptabilité / Menu Structure / Plan analytique / Fenêtre Section Gestion Commerciale / Menu Structure / Comptabilité / Plan analytique / Fenêtre Section Gestion Commerciale / Menu Structure / Codes affaire / Fenêtre Affaire	F_COMPTEA
IBOCompteAContact	Contacts des codes affaires	Gestion commerciale / Menu Structure / Codes affaires / Fenêtre code affaire / onglet contacts	F_COMPTEACONTACT
IBOCompteAMedia	Documents rattachés aux codes affaires	Gestion commerciale / Menu Structure / Codes affaires / Fenêtre code affaire / onglet Champs libres / Documents rattachés	F_COMPTEAMEDIA

Objet	Description	Correspondance dans l'application	Table correspondante
IBOCompteG3	Compte général	Comptabilité / Menu Structure / Plan comptable / Fenêtre Compte Gestion Commerciale / Menu Structure / Comptabilité / Plan comptable / Fenêtre Compte	F_COMPTEG
IBOCompteR	Compte reporting	Comptabilité / Menu Structure / Plan reporting / Fenêtre Compte	F_COMPTER
IBODossierContact3	Contact du dossier	Comptabilité – Gestion commerciale / Menu Fichier / Paramètres société / Vos Contacts	F_CONTACTD
IBOEcriture3	Ecriture comptable	Comptabilité / Menu Traitement / Saisie des écritures, etc…	F_ECRITUREC
IBOEcritureA3	Ecriture analytique	Comptabilité / Menu Traitement / Saisie des écritures, etc…	F_ECRITUREA
IBOFournisseur3	Fournisseur	Gestion Commerciale / Menu Structure / Fournisseurs / Fenêtre Fournisseur	F_COMPTET
IBOJournal3	Journal	Comptabilité / Menu Structure / Codes journaux / Fenêtre Code journal	F_JOURNAUX
IBOModeleEcriture3	Entête de modèle de saisie	Comptabilité / Menu Structure / Modèles / Modèles de saisie / Fenêtre Modèle de saisie	F_PIECEG
IBOModeleEcritureLigne3	Ligne de modèle de saisie (écriture générale)	Comptabilité / Menu Structure / Modèles / Modèles de saisie / Fenêtre Modèle de saisie	F_PIECEG
IBOModeleEcritureLigneA3	Ligne de modèle de saisie (analytique)	Comptabilité / Menu Structure / Modèles / Modèles de saisie / Fenêtre Modèle de saisie / Icône Modèle analytique	F_PIECEA
IBOModeleGrille	Entête de modèle de règlement	Comptabilité / Menu Structure / Modèles / Modèles de grille / Fenêtre Modèle de grille	F_MODELEG
IBOModeleGrilleLigne	Ligne de modèle de règlement	Comptabilité / Menu Structure / Modèles / Modèles de grille / Fenêtre Modèle de grille	F_EMODELEG
IBOModeleReglement	Entête de modèle de grille	Comptabilité / Menu Structure / Modèles / Modèles de règlement / Fenêtre Modèle de règlement	F_MODELER
IBOModeleReglementLigne	Ligne de modèle de grille	Comptabilité / Menu Structure / Modèles / Modèles de règlement / Fenêtre Modèle de règlement	F_EMODELER

Objet	Description	Correspondance dans l'application	Table correspondante
IBOPays	Pays	Comptabilité – Gestion commerciale / Paramètres société / International / Pays	F_PAYS
IBOTaxe3	Taxe	Comptabilité / Menu Structure / Taux de taxe / Fenêtre Taxe Gestion Commerciale / Menu Structure / Comptabilité / Taux de taxe / Fenêtre Taxe	F_TAXE
IBOTiers3	Tiers	Comptabilité / Menu Structure / Plan tiers / Fenêtre Tiers Gestion Commerciale / Menu Structure / Clients / Fenêtre Client Gestion Commerciale / Menu Structure / Fournisseurs / Fenêtre Fournisseur	F_COMPTET
IBOTiersBanque3	Banque tiers	Comptabilité / Menu Structure / Plan tiers / Fenêtre Tiers / Onglet Banques Gestion Commerciale / Menu Structure / Clients / Fenêtre Client / Onglet Banques Gestion Commerciale / Menu Structure / Fournisseurs / Fenêtre Fournisseur / Onglet Banques	F_BANQUET
IBOTiersContact3	Contact tiers	Comptabilité / Menu Structure / Plan tiers / Fenêtre Tiers / Onglet Contacts Gestion Commerciale / Menu Structure / Clients / Fenêtre Client / Onglet Contacts Gestion Commerciale / Menu Structure / Fournisseurs / Fenêtre Fournisseur / Onglet Contacts	F_CONTACTT
IBOTiersMedia3	Documents multimédia du tiers	Comptabilité / Menu Structure / Plan tiers / Fenêtre Tiers / Onglet Champs libres / Documents rattachés Gestion Commerciale / Menu Structure / Clients / Fenêtre Client / Onglet Champs libres / Documents rattachés Gestion Commerciale / Menu Structure / Fournisseurs / Fenêtre Fournisseur / Onglet	F_COMPTETMEDIA

Objet	Description	Correspondance dans l'application	Table correspondante
		Champs libres / Documents rattachés	
IBOTiersPart3	Tiers de type client ou fournisseur	Comptabilité / Menu Structure / Plan tiers / Fenêtre Tiers Gestion Commerciale / Menu Structure / Clients / Fenêtre Client Gestion Commerciale / Menu Structure / Fournisseurs / Fenêtre Fournisseur	F_COMPTET
IBOTiersReglement3	Règlement tiers	Comptabilité / Menu Structure / Plan tiers / Fenêtre Tiers / Onglet Paramètres / Conditions de paiement Gestion Commerciale / Menu Structure / Clients / Fenêtre Client / Onglet Paramètres / Conditions de paiement Gestion Commerciale / Menu Structure / Fournisseurs / Fenêtre Fournisseur / Onglet Paramètres / Conditions de paiement	F_REGLEMENTT
IBOCollaborateur	Collaborateurs	Comptabilité / Menu Structure / Collaborateurs Gestion commerciale / Menu Structure / Collaborateurs	F_COLLABORATEUR

Liens entre les objets

Le tableau ci-dessous permet pour chaque type d'objet de retrouver :

- L'interface de l'objet maître dont est issu l'objet ;
- la propriété *Factory* de l'objet maître permettant de fabriquer l'objet ;
- les différentes propriétés de l'objet permettant de fabriquer des sous-objets ;
- l'interface des sous-objets fabriqués.

Exemple :

IBOClient3 :

- *L'objet **IBOClient3** (Client) est issu d'un objet maître de type **BSCPTAApplication100c** ;*
- *L'objet maître fabrique un objet **IBOClient3** par appel de la propriété **FactoryClient()** ;*

- *L'objet **IBOClient3** propose différents Factory dont la propriété **FactoryClientLivraison()** ;*
- *Les sous-objets fabriqués par appel de la propriété **FactoryClientLivraison()** de l'objet **IBOClient3** sont de type **IBOClientLivraison3** (Lieu de livraison client).*

Interface objet *Interface objet maître* Propriété factory objet	**Description objet**	**Propriété factory sous-objet** *Interface sous-objet*	**Description sous-objet**
IBOClient3 *BSCPTAApplication100c* FactoryClient() FactoryClientProspect() FactoryProspect() Client		**FactoryClientLivraison()** *IBOClientLivraison3*	Lieu de livraison client
		FactoryClientTarif() *IBOArticleTarifClient3*	Tarif client article
		FactoryTiers() *IBOTiers3*	Tiers
		FactoryTiersBanque() *IBOTiersBanque2*	Banque tiers
		FactoryTiersCompteG() *IBOCompteG3*	Compte général du tiers
		FactoryTiersContact() *IBOTiersContact3*	Contact tiers
		FactoryTiersMedia() *IBOTiersMedia3*	Media tiers
		FactoryTiersReglement() *IBOTiersReglement3*	Règlement tiers
		FactoryClientTarifFamille() *IBOFamilleTarifClient*	Tarif famille client
		FactoryInfoComplement() *IBOInfoComplementClient*	Informations complémentaires Client
IBOClientLivraison3 *IBOClient3* FactoryClientLivraison()	Lieu de livraison d'un client	-	-
IBOCompteA3 *BSCPTAApplication100c* FactoryCompteA()	Compte analytique	**FactoryCompteAContact()** *IBOCompteAContact*	Contacts du code affaire
		FactoryCompteAMedia() *IBOCompteAMedia*	Documents rattachés au code affaire
IBOCompteG3 *BSCPTAApplication100c*	Compte général	**FactoryCompteGTiers()** *IBOTiers3*	Tiers rattaché

Interface objet *Interface objet maître* Propriété factory objet	**Description objet**	**Propriété factory sous-objet** *Interface sous-objet*	**Description sous-objet**
FactoryCompteG()			
IBOCollaborateur *BSCPTAApplication100c* FactoryCollaborateur()	Collaborateur	-	-
IBODossierContact3 *BSCPTAApplication100c* FactoryDossierContact()	Contact du dossier	-	-
IBOEcriture3 *BSCPTAApplication100c* FactoryEcriture()	Ecriture comptable ou analytique	**FactoryEcritureA()** *IBOEcritureA3*	Ecriture analytique
IBOEcritureA3 *IBOEcriture3* FactoryEcritureA()	Ecriture analytique	-	-
IBOFournisseur3 *BSCPTAApplication100c* FactoryFournisseur()	Fournisseur	**FactoryFournisseurTarif()** *IBOArticleTarifFournisseur3*	Tarif article fournisseur
		FactoryFournisseurTarifFamille() *IBOFamilletarifFournisseur*	Tarif famille fournisseur
		FactoryTiers() *IBOTiers3*	Tiers
		FactoryTiersBanque() *IBOTiersBanque3*	Banque tiers
		FactoryTiersCompteG() *IBOCompteG3*	Compte général tiers
		FactoryTiersContact() *IBOTiersContact3*	Contact tiers
		FactoryTiersMedia() *IBOTiersMedia3*	MediaTiers
		FactoryTiersReglement() *IBOTiersReglement3*	Règlement tiers
IBOJournal3 *BSCPTAApplication100c* FactoryJournal()	Journal	-	-
IBOModeleEcriture3 *BSCPTAApplication100c*	Entête de modèle de saisie	**FactoryModeleEcritureLigne()**	Ligne de modèle de saisie (écriture générale).

Interface objet *Interface objet maître* Propriété factory objet	**Description objet**	**Propriété factory sous-objet** *Interface sous-objet*	**Description sous-objet**
FactoryModeleEcriture()		*IBOModeleEcritureLigne3*	
IBOModeleEcritureLigne2 *IBOModeleEcriture3* FactoryModeleEcritureLigne()	Ligne de modèle de saisie (écriture générale).	**FactoryModeleEcritureLigneA()** *IBOModeleEcritureLigneA3*	Ligne de modèle de saisie (écriture analytique).
IBOModeleEcritureLigneA3 *IBOModeleEcritureLigne3* FactoryModeleEcritureLigneA()	Ligne de modèle de saisie (écriture analytique).	**-**	-
IBOTaxe3 *BSCPTAApplication100c* FactoryTaxe()	Taxe	-	-
IBOTiers3 *BSCPTAApplication100c* FactoryTiers()	Tiers	**FactoryTiersBanque()** *IBOTiersBanque3*	Banque tiers
		FactoryTiersCompteG() *IBOCompteG3*	Compte général tiers
		FactoryTiersContact() *IBOTiersContact3*	ContactTiers
		FactoryTiersMedia() *IBOTiersMedia3*	Media tiers
		FactoryTiersReglement() *IBOTiersReglement3*	Reglement tiers
IBOTiersBanque3 *IBOTiers3* FactoryTiersBanque()	Banque tiers	-	-
IBOTiersContact3 *IBOTiers3* FactoryTiersContact()	Contact tiers	-	-
IBOTiersMedia3 *IBOTiers3* FactoryTiersMedia()	Documents multimédia du tiers	-	-
IBOTiersPart3	Tiers faisant référence à un partenaire	**FactoryTiersBanque()** *IBOTiersBanque3*	Banque tiers

Interface objet *Interface objet maître* Propriété factory objet	**Description objet**	**Propriété factory sous-objet** *Interface sous-objet*	**Description sous-objet**
	commercial (Client ou Fournisseur)	**FactoryTiersCompteG()** *IBOCompteG3*	Compte général tiers
		FactoryTiersContact() *IBOTiersContact3*	ContactTiers
		FactoryTiersMedia() *IBOTiersMedia3*	Media tiers
		FactoryTiersReglement() *IBOTiersReglement3*	Reglement tiers
IBOTiersReglement3 *IBOTiers3* FactoryTiersReglement()	Règlement tiers	-	-
IBOModeleGrille	Modèle de grille	**FactoryModeleGrilleLigne()** *IBOModeleGrilleLigne*	Ligne de modèle de grille
IBOModeleGrilleLigne	Ligne de modèle de grille	-	-
IBOModeleReglement	Modèle de règlement	**FactoryModeleReglementLigne()** *IBOModeleReglementLigne*	Ligne de modèle de règlement
IBOModeleReglementLigne	Ligne de modèle de règlement	-	-

Chaque objet propose une propriété permettant de créer un objet de même type. Il ne s'agit pas d'un sous-objet, par conséquent cette propriété n'est pas listée dans ce tableau.

Exemple :

Les objets de type IBOClient3 *proposent une propriété* FactoryClient *permettant de créer un nouvel objet de type* IBOClient3*.*

IBOBanque

Banque.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Adresse() As IAdresse	Adresse.
Lecture / Ecriture	BQ_Abrege() As String	Abrégé.
Lecture / Ecriture	BQ_AchatDevise() As Boolean *	Banques / Donneur d'ordres : Achat en devise.
Lecture / Ecriture	BQ_BBAdresse() As Boolean *	Banques / Banque bénéficiaire : Adresse.
Lecture / Ecriture	BQ_BBBIC() As Boolean *	Banques / Banque bénéficiaire : Code BIC.
Lecture / Ecriture	BQ_BBCodePostal() As Boolean *	Banques / Banque bénéficiaire : Code postal.
Lecture / Ecriture	BQ_BBCompte() As Boolean *	Banques / Banque bénéficiaire : Compte.
Lecture / Ecriture	BQ_BBIntitule() As Boolean *	Banques / Banque bénéficiaire : intitulé.
Lecture / Ecriture	BQ_BBVille() As Boolean *	Banques / Banque bénéficiaire : Ville.
Lecture / Ecriture	BQ_BIAdresse() As Boolean *	Banques / Banque intermédiaire : Adresse.
Lecture / Ecriture	BQ_BIBIC() As Boolean *	Banques / Banque intermédiaire : Code BIC.
Lecture / Ecriture	BQ_BIC() As String *	Code BIC.
Lecture / Ecriture	BQ_BICodePostal() As Boolean *	Banques / Banque intermédiaire : Code postal.
Lecture / Ecriture	BQ_BIIntitule() As Boolean *	Banques / Banque intermédiaire : Intitulé.
Lecture / Ecriture	BQ_BIPays() As Boolean *	Banques / Banque intermédiaire : Pays.
Lecture / Ecriture	BQ_BIVille() As Boolean *	Banques / Banque intermédiaire : Ville.
Lecture / Ecriture	BQ_BordDevise() As Boolean *	Bordereau en devise.
Lecture / Ecriture	BQ_CodeIdent() As String *	Code identification.
Lecture / Ecriture	BQ_Contact() As String *	Contact.
Lecture / Ecriture	BQ_BOAdresse() As Boolean *	Banques / Donneur d'ordres : Adresse.
Lecture / Ecriture	BQ_DOAgenceCP() As Boolean *	Banques / Donneur d'ordres : Code postal de l'agence bancaire.
Lecture / Ecriture	BQ_DOAgenceVille() As Boolean *	Banques / Donneur d'ordres : Ville de l'agence bancaire.
Lecture / Ecriture	BQ_DOCle() As Boolean *	Banques / Donneur d'ordres : Clé RIB.
Lecture / Ecriture	BQ_DOCodeIdent() As Boolean *	Banques / Donneur d'ordres : Code identification.
Lecture / Ecriture	BQ_DOCodePostal() As Boolean *	Banques / Donneur d'ordres : Code postal.
Lecture / Ecriture	BQ_DOSiret() As Boolean *	Banques / Donneur d'ordres : N° SIRET.
Lecture / Ecriture	BQ_DOTypeIdent() As Boolean	Banques / Donneur d'ordres : Type identifiant du compte.

Accès	Syntaxe	Description
Lecture / Ecriture	BQ_DOVille() As Boolean *	Banques / Donneur d'ordres : Ville.
Lecture / Ecriture	BQ_FormatPrel() As BanqueFormatVirement	Format de prélèvelement (cf. énumérateur BanqueFormatVirement).
Lecture / Ecriture	BQ_FormatVir() As BanqueFormatVirement	Format de virement (cf. énumérateur BanqueFormatVirement).
Lecture / Ecriture	BQ_FormatVirInter() As BanqueFormatVirement	Format de virement international (cf. énumérateur BanqueFormatVirement).
Lecture / Ecriture	BQ_Intitule() As String	Intitulé.
Lecture / Ecriture	BQ_ModeleParam() As String	Modèle d'import paramétrable des extraits.
Lecture / Ecriture	BQ_ModeRemise() As BanqueModeRemise	Mode de remise (cf. énumérateur BanqueModeRemise).
Lecture / Ecriture	BQ_Remise() As BanqueRemise	Remise (cf.énumérateur BanqueRemise).
Lecture / Ecriture	BQ_TransEMailEnvoi() As String	Transfert : Adresse email d'envoi.
Lecture / Ecriture	BQ_TransSite() As String	Transfert : Site.
Lecture / Ecriture	BQ_VTAdresse() As Boolean *	Banques / Virement : Adresse.
Lecture / Ecriture	BQ_VTCodePostal() As Boolean *	Banques / Virement : Code postal.
Lecture / Ecriture	BQ_VTCodeService() As Boolean *	Banques / Virement : Code service.
Lecture / Ecriture	BQ_VTContrat() As Boolean	Banques / Virement : Référence du contrat de change.
Lecture / Ecriture	BQ_VTDateAchat() As Boolean *	Banques / Virement : Date d'achat.
Lecture / Ecriture	BQ_VTInstruction() As Boolean *	Banques / Virement : Instructions particulières.
Lecture / Ecriture	BQ_VTPays() As Boolean *	Banques / Virement : Pays.
Lecture / Ecriture	BQ_VTSiret() As Boolean *	Banques / Virement : N° SIRET.
Lecture / Ecriture	BQ_VTTauxChange() As Boolean *	Banques / Virement : Taux de change.
Lecture / Ecriture	BQ_VTVille() As Boolean *	Banques / Virement : Ville.
Lecture seule	FactoryBanque() As IBOBanqueFactory	Fabrique un objet Banque.
Lecture seule	FactoryBanqueRib() As IBOBanqueRibFactory	Fabrique un objet RIB Banque.
Lecture / Ecriture	Telecom() As ITelecom	Informations telecommunication.

* : Informations visibles sous Sage 100c Moyens de Paiement.

IBOBanqueContact

Contact de banque.

Interface héritée

Syntaxe	Description
IBIContact2	Cf. Interface IBIContact2 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	FactoryBanqueContact() As IBITypeObjectFactory	Fabrique un objet Contact de banque.
Lecture seule	Banque() As IBOBanque	Banque associée au contact.

IBOBanqueRIB

RIB Banque.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Adresse() As IAdresse	Adresse.
Lecture seule	Banque() As IBOBanque	Fabrique un objet Banque.
Lecture / Ecriture	EB_Abrege() As String	Abrégé.
Lecture / Ecriture	EB_Banque() As String	Code banque.
Lecture / Ecriture	EB_BenefAdresse() As IAdresse	Adresse bénéficiaire.
Lecture / Ecriture	EB_BenefRaisonSoc() As String	Raison sociale bénéficiaire.
Lecture / Ecriture	EB_BenefSiret() As String	N° SIRET bénéficiaire.
Lecture / Ecriture	EB_Bic() As String	Code BIC.
Lecture / Ecriture	EB_CalculIBAN() As Boolean	Calcul automatique de l'IBAN.
Lecture / Ecriture	EB_Cle() As String	Clé.
Lecture / Ecriture	EB_Commentaire() As String	Commentaire.
Lecture / Ecriture	EB_Compte() As String	Compte.
Lecture / Ecriture	EB_Emetteur(ByVal *sElt* As Integer) As String	Numéro d'émetteur (*sElt* de 1 à 3).
Lecture / Ecriture	EB_Guichet() As String	Guichet.
Lecture / Ecriture	EB_IBAN() As String	IBAN.
Lecture / Ecriture	EB_IntraGroupe() As Boolean	Compte Intra-groupe.

Accès	Syntaxe	Description
Lecture / Ecriture	EB_NomAgence() As String	Nom de l'agence.
Lecture / Ecriture	EB_PaysAgence() As String	Pays agence.
Lecture / Ecriture	EB_Struct() As RibType	Structure banque.
Lecture seule	FactoryBanqueRib() As IBOBanqueRibFactory	Fabrique un objet RIB Banque.
Lecture / Ecriture	Journal() As IBOJournal3	Journal banque.
Lecture / Ecriture	JournalEncaiss() As IBOJournal3	Journal encaissement.
Lecture / Ecriture	JournalEscompte() As IBOJournal3	Journal escompte.

IBOClient3

Client.

Interface héritée

Syntaxe	Description
IBOTiersPart3	Cf. Interface IBOTiersPart3 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	CatTarif() As IBPCategorieTarif	Catégorie tarifaire. *
Lecture/ Ecriture	CentraleAchat() As IBOClient3	Client centrale d'achat.
Lecture / Ecriture	CodeRisque() As IBPCodeRisque	Code risque.
Lecture / Ecriture	CT_Assurance() As Double	Plafon assurance crédit.
Lecture / Ecriture	CT_BLFact() As Boolean	1 BL/Facture.
Lecture / Ecriture	CT_ControlEnc() As ClientEncoursCtrlType	Contrôle de l'encours (Cf. énumérateur ClientEncoursCtrlType).
Lecture / Ecriture	CT_DelaiTransport() As Short	Délai de transport. *
Lecture / Ecriture	CT_LivrPartielle() As Boolean	Livraison partielle (True) ou non (False).
Lecture / Ecriture	CT_PrioriteLivr() As Short	Priorité de livraison.
Lecture seule	CT_Prospect() As Boolean	Client de type prospect.
Lecture / Ecriture	CT_Facture() As Short	Nombre de factures.
Lecture seule	FactoryClient() As IBOClientFactory3	Fabrique un objet Client.
Lecture seule	FactoryClientLivraison() As IBITypeObjectFactory	Fabrique un objet Lieu de livraison client. *
Lecture seule	FactoryClientProspect() As IBOClientProspectFactory	Fabrique un objet Client de type prospect ou non. *

Accès	Syntaxe	Description
Lecture seule	FactoryClientTarif() As IBOClientTarifFactory3	Fabrique un objet Tarif client. *
Lecture seule	FactoryClientTarifFamille() As IBOClientTarifFamilleFactory	Fabrique un objet tarif famille. *
Lecture seule	FactoryProspect() As IBOProspectFactory	Fabrique un objet Client de type prospect uniquement. *
Lecture seule	FactoryInfoComplement () As IBOFactoryInfoComplement	Fabrique un objet Infos complémentaire
Lecture / Ecriture	LivraisonPrincipal() As IBOClientLivraison3	Lieu de livraison principal. *
Lecture / Ecriture	Periodicite() As IBPPeriodicite	Périodicité. *
Lecture / Ecriture	Representant() As IBOCollaborateur	Représentant. *

Méthodes

Syntaxe	Description
TotalMarge(ByVal *dDebut* As Date, ByVal *dFin* As Date, ByVal *TypeDe* As DocumentType, ByVal *TypeA* As DocumentType) As Double	Marge réalisée pour une période donnée et une fourchette de types de documents donnée. *
RisqueReel() As Double	Montant du risque réel.

** : nécessite l'ouverture d'une base de Gestion Commerciale*

Traitements et initialisations par défaut

Méthode WriteDefault()

La méthode *WriteDefault()* créé une adresse de livraison principale (table F_LIVRAISON) à partir des informations renseignées sur l'adresse (propriété *Adresse*).

IBOClientLivraison3

Lieu de livraison client.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Client() As IBOClient3	Client auquel est associé le lieu de livraison client.
Lecture / Ecriture	ConditionLivraison() As IBPConditionLivraison	Condition de livraison. *
Lecture / Ecriture	Expedition() As IBPExpedition3	Mode d'expédition. *
Lecture seule	FactoryClientLivraison() As IBITypeObjectFactory	Fabrique un lieu de livraison client.
Lecture / Ecriture	Adresse() As IAdresse	Informations d'adresse.
Lecture / Ecriture	LI_Contact() As String	Contact.
Lecture / Ecriture	LI_Intitule() As String	Intitulé.
Lecture / Ecriture	Telecom() As ITelecom	Télécommunications.

** : nécessite l'ouverture d'une base de gestion Commerciale*

IBOCompteA3

Compte analytique.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Analyse() As IBPAnalyse	Section analytique.
Lecture / Ecriture	Analytique() As IBPAnalytique3	Plan analytique.
Lecture / Ecriture	CA_Achat() As Double	Objectif chiffre d'affaires achats
Lecture / Ecriture	CA_Classement() As String	Classement.
Lecture / Ecriture	CA_DateAcceptAffaire() As Date	Date d'acceptation de l'affaire.
Lecture seule	CA_DateCreate() As Date	Date de création.
Lecture / Ecriture	CA_DateCreationAffaire() As Date	Date de création de l'affaire.
Lecture / Ecriture	CA_DateDebutAffaire() As Date	Date de début d'affaire.
Lecture / Ecriture	CA_DateFinAffaire() As Date	Date de fin d'affaire.
Lecture / Ecriture	CA_Domaine() As CompteADomaineType	Domaine (cf. énumérateur CompteADomaineType).
Lecture / Ecriture	CA_Intitule() As String	Intitulé.

Accès	Syntaxe	Description
Lecture / Ecriture	CA_ModeFacturation() As CompteAModeFacturationType	Mode de facturation de l'affaire.
Lecture / Ecriture	CA_Num() As String	Numéro.
Lecture / Ecriture	CA_Raccourci() As String	Raccourci.
Lecture / Ecriture	CA_Report() As Boolean	Report à nouveau.
Lecture / Ecriture	CA_SautLigne() As Short	Saut de N lignes. Si N = 0 Alors Saut de page Sinon si N >= 1 Alors Saut de ligne
Lecture / Ecriture	CA_SautPage() As Boolean	Saut de page (True) ou saut de ligne (False).
Lecture / Ecriture	CA_Sommeil() As Boolean	En sommeil.
Lecture / Ecriture	CA_Statut() As CompteAStatutType	Statut du code affaire.
Lecture / Ecriture	CA_Type() As CompteAType	Type du compte analytique (cf. énumérateur CompteAType).
Lecture / Ecriture	CA_Vente() As Double	Objectif chiffre d'affaires ventes.
Lecture / Ecriture	Collaborateur() As IBOCollaborateur	Collaborateur chargé du code affaire.
Lecture seule	FactoryCompteA() As IBOCompteAFactory3	Fabrique un objet Compte analytique.
Lecture seule	FactoryCompteAContact() As IBOCompteAContact	Fabrique un objet contact de compte analytique.
Lecture seule	FactoryCompteAMedia() As IBOCompteAMedia	Fabrique un objet document rattaché au compte analytique.
Lecture seule	InfoLibre() As IBIValues	Valeur information libre.

Traitements et initialisations par défaut

Méthode SetDefault()

Si l'objet est non persistant, l'appel de la méthode *SetDefault()* initialise les propriétés suivantes :

Propriété	Valeur	Description
CA_Classement	Si CA_Classement = "" Alors CA_Classement = CA_Intitule	Affecte la valeur du champ Intitulé au champ Classement si celui-ci est vide.
CA_Intitule	Si CA_Intitule = "" Alors CA_Intitule = CA_Num	Affecte la valeur du champ Numéro de compte analytique au champ Intitulé si celui-ci est vide.

CA_Num		Ajustement du numéro de compte en fonction du paramétrage dans Paramètres société / Analytique.
CA_Type	Si CA_Type = CompteATypeTotal Alors Remise à zéro des champs inutilisés	Remet à zéro les champs inutilisés si le compte analytique est de type Total.

Si l'objet est persistant, l'appel de la méthode *SetDefault()* initialise les propriétés suivantes :

Propriété	Valeur	Description
CA_Classement	Si CA_Classement = "" Alors CA_Classement = CA_Intitule	Affecte la valeur du champ Intitulé au champ Classement si celui-ci est vide.
CA_Intitule	Si CA_Intitule = "" Alors CA_Intitule = CA_Num	Affecte la valeur du champ Numéro de compte analytique au champ Intitulé si celui-ci est vide.

Méthode Write()

La méthode *Write()* enregistre l'objet dans la base de données sans effectuer aucun traitement d'initialisation des propriétés de l'objet.

Méthode WriteDefault()

La méthode *WriteDefault()* enregistre l'objet dans la base de données sans effectuer aucun traitement d'initialisation des propriétés de l'objet.

IBOCompteAContact

Contacts des codes affaires.

Interface héritée

Syntaxe	Description
IBIContact2	Cf. Interface IBIContact2 pour les propriétés et méthodes héritées.

Propriété

Accès	Syntaxe	Description
Lecture seule	CompteA() As IBOCompteA3	Code affaire associé au contact.

IBOCompteAMedia

Document rattaché au code affaire.

Interface héritée

Syntaxe	Description
IBIMedia	Cf. Interface IBIMedia pour les propriétés et méthodes héritées.

Propriété

Accès	Syntaxe	Description
Lecture seule	CompteA() As IBOCompteA3	Code affaire associé au document rattaché.

IBOCompteG3

Compte général.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	CG_Analytique() As Boolean	Saisie analytique.
Lecture / Ecriture	CG_Classement() As String	Classement.
Lecture seule	CG_DateCreate() As Date	Date de création.
Lecture / Ecriture	CG_Devise() As Boolean	Saisie devise.
Lecture / Ecriture	CG_Echeance() As Boolean	Saisie de l'écheance.
Lecture / Ecriture	CG_Intitule() As String	Intitulé.
Lecture / Ecriture	CG_Lettrage() As Boolean	Lettrage automatique.
Lecture / Ecriture	CG_LettrageSaisie() As Boolean	Lettrage en saisie.
Lecture / Ecriture	CG_Num() As String	Numéro de compte.
Lecture / Ecriture	CG_Quantite() As Boolean	Saisie de la quantité.
Lecture / Ecriture	CG_Raccourci() As String	Raccourci.
Lecture / Ecriture	CG_Regroup() As Boolean	Regroupement.
Lecture / Ecriture	CG_Report() As CompteGReportType	Report à nouveau (cf. énumérateur CompteGReportType).

Accès	Syntaxe	Description
Lecture / Ecriture	CG_SautLigne() As Short	Saut de N lignes. Si N = 0 Alors Saut de page Sinon si N >= 1 Alors Saut de ligne
Lecture / Ecriture	CG_SautPage() As Boolean	Saut de page (True) ou saut de ligne (False).
Lecture / Ecriture	CG_Sommeil() Ad Boolean	Mise en sommeil.
Lecture / Ecriture	CG_Tiers() As Boolean	Saisie compte tiers.
Lecture / Ecriture	CG_Type() As CompteGType	Type du compte général (cf. énumérateur CompteGType).
Lecture / Ecriture	CompteR() As IBOCompteR	Compte reporting.
Lecture / Ecriture	Devise() As IBPDevise2	Devise.
Lecture seule	FactoryCompteG() As IBOCompteGFactory3	Fabrique un objet Compte général.
Lecture seule	FactoryCompteGTiers() As IBOCompteGTiersFactory3	Fabrique un objet Compte tiers rattaché.
Lecture seule	InfoLibre() As IBIValues	Valeur information libre.
Lecture seule	NatureCompte() As IBPNatureCompte	Nature de compte.
Lecture / Ecriture	Taxe() As IBOTaxe3	Taxe associée au compte général.

Traitements et initialisations par défaut

Méthode SetDefault()

Si l'objet est non persistant, l'appel de la méthode *SetDefault()* initialise les propriétés suivantes :

Propriété	Valeur	Description
NatureCompte() As IBPNatureCompte	Nature de compte = Nature de compte paramétrée dans Paramètres société / Comptes généraux / Nature de compte	Affectation de la Nature de compte paramétrée dans Paramètres société / Comptes généraux / Nature de compte correspondant au Numéro du compte général.
CG_Analytique() As Boolean	CG_Analytique = Saisie analytique de la Nature de compte	Affectation de l'option Saisie analytique de la Nature de compte à la propriété Saisie analytique du Compte général.
CG_Devise() As Boolean	CG_Devise = Saisie devise de la Nature de compte	Affectation de l'option Saisie devise de la Nature de compte à la propriété Saisie devise du Compte général.

Propriété	Valeur	Description
CG_Echeance() As Boolean	CG_Echeance = Saisie de l'échéance de la Nature de compte	Affectation de l'option Saisie de l'échéance de la Nature de compte à la propriété Saisie de l'échéance du Compte général.
CG_Lettrage() As Boolean	CG_Lettrage = Saisie lettrage de la Nature de compte	Affectation de l'option Saisie lettrage de la Nature de compte à la propriété Saisie lettrage du Compte général.
CG_Num() As String	-	Ajustement de la longueur du numéro de compte général en fonction du paramétrage dans Paramètres société / Comptes généraux / Nature de compte.
CG_Quantite() As Boolean	CG_Quantite = Saisie de la quantité de la Nature de compte	Affectation de l'option Saisie de la quantité de la Nature de compte à la propriété Saisie de la quantité du Compte général.
CG_Regroup() As Boolean	CG_Regroup = Regroupement de la Nature de compte	Affectation de l'option Regroupement de la Nature de compte à la propriété Regroupement du Compte général.
CG_Report() As CompteGReportType	CG_Report = Report à nouveau de la Nature de compte	Affectation de l'option Report à nouveau de la Nature de compte à la propriété Report à nouveau du Compte général.
CG_Tiers() As Boolean	CG_Tiers = Saisie compte tiers de la Nature de compte	Affectation de l'option Saisie compte tiers de la Nature de compte à la propriété Saisie compte tiers du Compte général.
CG_Type() As CompteGType	Si CG_Type = CompteGTypeTotal Alors Vidage des propriétés inutilisées	Si le Compte général est de type Total, les propriétés inutilisées sont vidées.
CT_Devise() As Boolean	Si Devise <> Nothing Alors CT_Devise = True	Si une devise est affectée au compte général, la propriété Saisie devise est positionnée à True.
Devise() As IBPDevise2	Devise = Devise de la Nature de compte	Affectation de l'option Devise de la Nature de compte à la propriété Devise du Compte général.

Si l'objet est persistant, l'appel de la méthode SetDefault() initialise les propriétés suivantes :

Propriété	Valeur	Description
CG_Classement() As String	Si CG_Classement = "" Alors CG_Classement = CG_Intitule	Affecte la propriété Intitulé du compte à la propriété Classement du compte si cette propriété est vide.
CG_Intitule() As String	Si CG_Intitule = "" Alors CG_Intitule = CG_Num	Affecte la propriété Numéro de compte à la propriété Intitulé du compte si cette propriété est vide.
CT_Devise() As Boolean	Si Devise <> Nothing Alors CT_Devise = True	Si une devise est affectée au compte général, la propriété Saisie devise est positionnée à True.

Méthode Write()

La méthode *Write()* enregistre l'objet dans la base de données sans effectuer aucun traitement d'initialisation des propriétés de l'objet.

Méthode WriteDefault()

La méthode *WriteDefault()* créé les enregistrements "éléments de taxe" (table F_ETAXE) associés au code taxe (propriété Taxe) du compte général avant d'enregistrer l'objet dans la base de données.

IBOCompteR

Compte reporting.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	CR_Num() As String	Numéro de compte reporting.
Lecture / Ecriture	CR_Intitule() As String	Intitulé du compte.
Lecture / Ecriture	CR_Type() As CompteRType	Type de compte.
Lecture / Ecriture	CR_Classement() As String	Classement du compte.
Lecture / Ecriture	CR_SautLigne() As Integer	Saut de ligne.
Lecture / Ecriture	CR_SautPage() As Boolean	Saut de page.

IBODossierContact3

Contact dossier.

Interface héritée

Syntaxe	Description
IBIContact2	Cf. Interface IBIContact2 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Adresse() As IAdresse	Informations d'adresse.
Lecture seule	FactoryDossierContact() As IBODossierContactFactory2	Fabrique un objet Contact dossier.

IBOEcriture3

Ecriture.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	CompteG() As IBOCompteG3	Compte général.
Lecture / Ecriture	CompteGContrepartie() As IBOCompteG3	Compte général de contrepartie.
Lecture / Ecriture	Date() As Date	Date.
Lecture / Ecriture	DateSaisie() As Date	Date de saisie.
Lecture / Ecriture	Devise() As IBPDevise2	Devise.
Lecture seule	EC_ANType() As .EcritureANType	Type à nouveau (Cf. énumérateur EcritureANType).
Lecture seule	EC_Cloture() As Boolean	Ecriture clôturée (True) ou non (False).
Lecture seule	EC_DateRappro() As Date	Date de rapprochement.
Lecture / Ecriture	EC_Devise() As Double	Montant en devise.
Lecture / Ecriture	EC_Echeance() As Date	Date d'échéance.
Lecture seule	EC_ExportExpert() As EcritureExpertType	Export expert-comptable (Cf. énumérateur EcritureExpertType).

Accès	Syntaxe	Description
Lecture seule	EC_Impression() As EcritureImpressionType	Etat d'impression écriture (Cf. énumérateur EcritureImpressionType).
Lecture / Ecriture	EC_Intitule() As String	Intitulé.
Lecture seule	EC_Lettrage() As String	Lettrage.
Lecture seule	EC_LettrageQ() As String	Lettre quantité.
Lecture / Ecriture	EC_Montant() As Double	Montant.
Lecture seule	EC_NO() As Long	Numéro interne de l'écriture.
Lecture seule	EC_NoCloture() As Long	Numéro interne de clôture
Lecture / Ecriture	EC_Norme() As EcritureNormeType	Norme (Cf. énumérateur EcritureNormeType).
Lecture / Ecriture	EC_Parite() As Double	Parité.
Lecture / Ecriture	EC_Piece() As String	Numéro de pièce.
Lecture seule	EC_Pointage() As String	Pointage.
Lecture / Ecriture	EC_Quantite() As Double	Quantité.
Lecture seule	EC_Rappel() As Short	Nombre de rappels.
Lecture / Ecriture	EC_Reference() As String	Référence pièce.
Lecture / Ecriture	EC_RefPiece() As String	Numéro de facture.
Lecture seule	EC_Remise() As Boolean	Remis en banque (True) ou non remis en banque (False).
Lecture seule	EC_Rtype() As Boolean	Ecriture révisée (True) ou non (False).
Lecture / Ecriture	EC_Sens() As EcritureSensType	Sens de l'écriture (Cf. énumérateur EcritureSensType).
Lecture seule	EC_TresoPiece() As String	Pièce de trésorerie.
Lecture seule	EC_Type() As EcritureType	Type d'écriture (Cf. énumérateur EcritureType).
Lecture seule	FactoryEcriture() As IBOEcritureFactory3	Fabrique un objet Ecriture.
Lecture seule	FactoryEcritureA() As IBOEcritureAFactory2	Fabrique un objet Ecriture analytique.
Lectrue seule	FactoryEcritureMedia() As IBOEcritureMedia	Fabrique un objet média écriture.
Lecture seule	HasLettrage() As Boolean	Ecriture lettrée (True) ou non (False).
Lecture seule	HasLettrageQ() As Boolean	Ecriture lettrée en quantité (True) ou non (False).
Lecture seule	HasPointage() As Boolean	Ecriture pointée (True) ou non (False).
Lecture / Ecriture	HasTa_Provenance() As Boolean	Renvoie *True* si la provenance de l'écriture est différente de *Aucune.*
Lecture seule	InfoLibre() As IBIValues	Valeur information libre.
Lecture / Ecriture	Journal() As IBOJournal3	Journal.

Accès	Syntaxe	Description
Lecture / Ecriture	Reglement() As IBPReglement3	Mode de règlement.
Lecture / Ecriture	TA_Provenance() As TaxeProvenanceType	Provenance (Cf. énumérateur TaxeProvenanceType).
Lecture / Ecriture	Taxe() As IBOTaxe3	Taxe.
Lecture / Ecriture	Tiers() As IBOTiers3	Tiers.
Lecture / Ecriture	TiersContrepartie() As IBOTiers3	Contrepartie tiers.
Lecture seule	RegistreRevision() As IRegistreRevision	Régularisation des charges et produits.
Ecriture	WriteDefaultRegistreTaxe()	Ecrit les registres de taxe. En modification, recalcule les taxes
Lecture	BOOL HasRegistreTaxes()	Indique si le Registre de taxes de la ligne d'écriture existe
Lecture	IREGISTRETAXES GetRegistreTaxes()	Donne le Registre de taxes lié
Lecture	IBOECRITURE GetEcritureRegistreTaxes()	Donne l'écriture où le Registre de taxes est lié

Traitements et initialisations par défaut

Ecriture de centralisation des journaux de trésorerie

Lors de la création, de la modification ou de la suppression d'écritures dans un journal de trésorerie en centralisation fin de mois (option *Contrepartie à chaque ligne* du journal non cochée), l'écriture de centralisation est automatiquement mise à jour.

Méthode SetDefault()

Lorsque l'écriture est non persistante, l'appel de la méthode *SetDefault()* initialise les propriétés suivantes :

Propriété	Valeur	Description
Date	Si Ecriture.Date= 0 Alors Ecriture.Date = Date du jour	Si la date de l'écriture n'est pas renseignée (ou contient une valeur par défaut), la date d'écriture correspond à la date du jour.
DateSaisie	Si Ecriture.DateSaisie = 0 Alors Ecriture.DateSaisie = Date du jour	Si la date de saisie n'est pas renseignée (ou contient une valeur par défaut), la date de saisie correspond à la date du jour.

Propriété	Valeur	Description
Taxe	Si Ecriture.Taxe = Nothing et CompteG.Taxe <> Nothing Alors Ecriture.Taxe = CompteG.Taxe	Si la taxe de l'écriture n'est pas renseignée et que le compte général de l'écriture possède une taxe, la taxe du compte général est affectée à la taxe de l'écriture.
Tiers	Si Ecriture.Tiers = Nothing Alors Ecriture.Tiers = 1er tiers dans F_COMPTETG de CompteG	Si le tiers de l'écriture n'est pas renseigné, le premier tiers de la table des tiers du compte général (table F_COMPTETG) est repris.
CompteG	Si Ecriture.CompteG = Nothing et Ecriture.Tiers<> Nothing Alors Ecriture.CompteG = Tiers.CompteGPrinc	Si l'écriture n'a pas de compte général mais qu'elle a un tiers renseigné, le compte général correspondra au compte général principal du tiers de l'écriture.
EC_Parite	Si Ecriture.Devise <> Nothing et Ecriture.Date <> 0 et Ecriture.EC_Parite = 0 Alors Ecriture.EC_Parite = Cours de la devise à la date de l'écriture	Si la devise et la date de l'écriture ne sont pas renseignées et que la parité est à zéro, la parité correspondra au cours de la devise à la date de l'écriture.
EC_Devise	Si Ecriture.Devise = Nothing Alors Ecriture.EC_Devise = 0	Si la devise de l'écriture n'est pas renseignée, le montant en devise sera égal à zéro.
EC_Parite	Si Ecriture.Devise = Nothing Alors Ecriture.EC_Parite = 0	Si la devise de l'écriture n'est pas renseignée, la parité sera égale à zéro.
EC_Montant	Si Ecriture.Devise <> Nothing et Ecriture.EC_Montant = 0 Alors Ecriture.EC_Montant = Conversion de EC_Devise en fonction de Ecriture.Devise et Ecriture.Date	Si la devise de l'écriture est renseignée mais que le montant de l'écriture en monnaie de tenue ne l'est pas, le montant de l'écriture en monnaie de tenue est calculé par conversion du montant en devise en fonction de la devise et de la date de l'écriture.

En fonction de la gestion ou nom de la norme IFRS, la colonne *Type norme* est automatiquement initialisée lors de l'appel de *SetDefault()* (Cf. *Manuel utilisateur Sage Comptabilité 100*).

Lorsque l'écriture est persistante, l'appel de la méthode *SetDefault()* initialise les propriétés suivantes :

Propriété	Valeur	Description
Date	Si Ecriture.Date= 0 Alors Ecriture.Date = Date du jour	Si la date de l'écriture n'est pas renseignée (ou contient une valeur par défaut), la date d'écriture correspond à la date du jour.
DateSaisie	Si Ecriture.DateSaisie = 0 Alors Ecriture.DateSaisie = Date du jour	Si la date de saisie n'est pas renseignée (ou contient une valeur par défaut), la date de saisie correspond à la date du jour.

Propriété	Valeur	Description
CompteG	Si Ecriture.CompteG = Nothing et Ecriture.Tiers<> Nothing Alors Ecriture.CompteG = Tiers.CompteGPrinc	Si l'écriture n'a pas de compte général mais qu'elle a un tiers renseigné, le compte général correspondra au compte général principal du tiers de l'écriture.
EC_Parite	Si Ecriture.Devise <> Nothing et Ecriture.Date <> 0 et Ecriture.EC_Parite = 0 Alors Ecriture.EC_Parite = Cours de la devise à la date de l'écriture	Si la devise et la date de l'écriture ne sont pas renseignées et que la parité est à zéro, la parité correspondra au cours de la devise à la date de l'écriture.
EC_Devise	Si Ecriture.Devise = Nothing Alors Ecriture.EC_Devise = 0	Si la devise de l'écriture n'est pas renseignée, le montant en devise sera égal à zéro.
EC_Parite	Si Ecriture.Devise = Nothing Alors Ecriture.EC_Parite = 0	Si la devise de l'écriture n'est pas renseignée, la parité sera égale à zéro.

En fonction de la gestion ou non de la norme IFRS, la colonne *Type norme* est automatiquement initialisée lors de l'appel de *SetDefault()* (Cf. *Manuel utilisateur Sage 100c Comptabilité*).

Méthode Write()

La méthode *Write()* enregistre l'objet dans la base de données après avoir initialisé la colonne *Type norme* en fonction de la gestion ou non de la norme IFRS (Cf. *Manuel utilisateur Sage 100c Comptabilité*).

Méthode WriteDefault()

- Si l'écriture est non persistante, génération des écritures analytiques si nécessaire.
- Si l'écriture est persistante, que son montant est modifié et qu'il existe des écritures analytiques associées, les valeurs des écritures analytiques sont recalculées au prorata.
- Comme la méthode *Write()*, la méthode *WriteDefault()* enregistre l'objet dans la base de données après avoir initialisé la colonne *Type norme* en fonction de la gestion ou non de la norme IFRS (Cf. *Manuel utilisateur Sage 100c Comptabilité*).
- Gestion du bon à payer en fonction du paramétrage défini sur le dossier comptable.

IBORegistreTaxesLignes

Lignes de Registres de taxes.

Accès	Syntaxe	Description
Lecture	short GetNumeroLigne()	Numéro de ligne de taxes du registre
Lecture / Ecriture	IBOCOMPTEG GetCompteG() SetCompteG(IBOCOMPTEG)	Compte général de Taxe
Lecture / Ecriture	TaxeProvenance GetTA_Provenance() SetTA_Provenance(TaxeProvenance)	Type de provenance Taxe
Lecture / Ecriture	TaxeTypeTaux GetTA_TTaux() SetTA_TTaux(TaxeTypeTaux)	Type de taux de taxe
Lecture / Ecriture	double GetTA_Taux() void SetTA_Taux(double)	Taux de la taxe
Lecture / Ecriture	double GetRT_Base() SetRT_Base(double)	Base de la taxe
Lecture / Ecriture	double GetRT_Montant () SetRT_Montant (double)	Montant de la taxe
Lecture / Ecriture	IBOTAXE GetTaxe () SetTaxe(IBOTAXE)	Taxe associée
Lecture / Ecriture	TypeLigneRegistre GetRT_TypeLigne() SetRT_TypeLigne(TypeLigneRegistre)	Type de ligne
Ecriture	Clear()	Marquage pour suppression de la ligne de taxe

IBORegistreTaxes

Registres de taxes.

Accès	Syntaxe	Description
Lecture	short GetMaxLignes()	Nombre de lignes de taxes d'un registre
Lecture	long GetRT_No()	Numéro de registre de taxes
Lecture / Ecriture	CDATE GetRT_DateReg() SetRT_No(CDATE)	Date du règlement
Lecture / Ecriture	CDATE GetRT_DatePiece() SetRT_DatePiece(CDATE)	Date de la pièce
Ecriture	Clear()	Marquage pour suppression du registre
Lecture / Ecriture	IREGISTRETAXESLIGNE GetLigne(short) SetLigne(short,IREGISTRETAXESLIGNE)	Propriété Ligne de registre de taxes : numéro de ligne entre **1 et GetMaxLignes()**

IBOEcritureA3

Ecriture analytique.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	CompteA() As IBOCompteA3	Compte analytique.
Lecture / Ecriture	EA_Montant() As Double	Montant.
Lecture / Ecriture	EA_Quantite() As Double	Quantité.
Lecture seule	Ecriture() As IBOEcriture3	Ecriture à laquelle est associée l'écriture analytique.
Lecture seule	FactoryEcritureA() As IBOEcritureAFactory2	Fabrique un objet Ecriture analytique.

IBOEcritureMedia

Médias des écritures générales.

Interface héritée

Syntaxe	Description
IBIMedia	Cf. Interface IBIMedia pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Ecriture() As IBOEcriture3	Ecriture générale associée au média.
Lecture seule	EcritureMediaFactory() As IBITypeObjetFactory	Fabrique un objet média.

IBOEcritureOD

Ecriture d'od analytique.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Journal() As IBOJournal3	Journal analytique.
Lecture / Ecriture	Date() As Date	Date de l'écriture.
Lecture / Ecriture	ER_Piece() As String	Numéro de pièce.
Lecture / Ecriture	ER_RefPiece() As String	Référence pièce.
Lecture / Ecriture	JA_IFRS() As Boolean	Ecriture IFRS.
Lecture / Ecriture	CompteG() As IBOCompteG3	Compte général.
Lecture / Ecriture	ER_Intitule() As String	Intitulé de l'écriture.
Lecture / Ecriture	ER_Sens() As EcritureSensType	Sens de l'écriture.
Lecture seule	ER_Type() As EcritureODType	Type de l'écriture.
Lecture / Ecriture	ER_Norme() As EcritureNormeType	Norme de l'écriture.
Lecture / Ecriture	CompteA() As IBOCompteA3	Section analytique.
Lecture / Ecriture	ER_MontantA() As Double	Montant de l'écriture.
Lecture / Ecriture	ER_QuantiteA() As Double	Quantité de l'écriture.

IBOFournisseur3

Fournisseur.

Interface héritée

Syntaxe	Description
IBOTiersPart3	Cf. Interface IBOTiersPart3 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	CT_DelaiAppro() As Short	Délai d'approvisionnement. *

Accès	Syntaxe	Description
Lecture / Ecriture	ConditionLivraison() As IBPConditionLivraison	Condition de livraison. *
Lecture / Ecriture	Expedition() As IBPExpedition3	Mode d'expédition. *
Lecture seule	FactoryFournisseur() As IBOFournisseurFactory3	Fabrique un objet Fournisseur.
Lecture seule	FactoryFournisseurTarif() As IBOFournisseurTarifFactory3	Fabrique un objet Tarif fournisseur. *
Lecture seule	FactoryFournisseurTarifFamille() As IBOFournisseurTarifFamilleFactory	Fabrique un objet Tarif fournisseur par famille. *
Lecture / Ecriture	Representant() As IBOCollaborateur	Représentant.

** : nécessite l'ouverture d'une base de Gestion Commerciale*

IBOJournal3

Journal.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	CompteG() As IBOCompteG3	Compte général.
Lecture seule	FactoryJournal() As IBOJournalFactory3	Fabrique un objet Journal.
Lecture / Ecriture	JO_Contrepartie() As Boolean	Contrepartie à chaque ligne (True) ou centralisation en fin de mois (False).
Lecture / Ecriture	JO_IFRS() As Boolean	Réservé IFRS (True) ou non (False).
Lecture / Ecriture	JO_Intitule() As String	Intitulé.
Lecture / Ecriture	JO_LettrageSaisie() As Boolean	Lettrage en saisie.
Lecture / Ecriture	JO_NotCalcTot() As Boolean	Masquer les totaux (True) ou afficher les totaux (False).
Lecture / Ecriture	JO_Num() As String	Code journal.
Lecture / Ecriture	JO_NumPiece() As JournalNumPieceType	Type numérotation pièce (Cf. énumérateur JournalNumPieceType).
Lecture / Ecriture	JO_Rappro() As JournalRapproType	Type rapprochement (Cf. énumérateur JournalRapproType).
Lecture/ Ecriture	JO_Reglement() As Boolean	Règlement définitif.
Lecture / Ecriture	JO_SaisAnal() As Boolean	Saisie analytique (True) ou aucune saisie analytique (False).

Accès	Syntaxe	Description
Lecture / Ecriture	JO_Sommeil() As Boolean	Mis en sommeil (True) ou actif (False).
Lecture / Ecriture	JO_Type() As JournalType	Type du journal (Cf. énumérateur JournalType).
Lecture seule	NextEC_Piece(ByVal DatePeriode As Date) As String	Prochain numéro de pièce d'écriture.
Lecture seule	TypePeriodCloture(ByVal DatePeriode As Date) As PeriodClotureType	Retourne le type de clôture pour la période passée en paramètre (cf. énumérateur PeriodClotureType).

IBOJournalA

Journal analytique.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	JA_Num() As String	Code journal analytique.
Lecture / Ecriture	JA_Intitule() As String	Intitulé.
Lecture / Ecriture	JA_Sommeil() As Boolean	Mis en sommeil.
Lecture / Ecriture	JA_IFRS() As Boolean	Réservé IFRS (True) ou non (False).

IBOModeleEcriture3

Entête de modèle de saisie.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	FactoryModeleEcriture() As IBOModeleEcritureFactory2	Fabrique un objet Entête de modèle de saisie.

Accès	Syntaxe	Description
Lecture seule	FactoryModeleEcritureLigne() As IBITypeObjectFactory	Fabrique un objet Ligne de modèle de saisie d'écritures générales.
Lecture / Ecriture	Journal() As IBOJournal3	Journal du modèle de saisie.
Lecture / Ecriture	JO_Type() As JournalType	Type du journal (Cf. énumérateur JournalType).
Lecture / Ecriture	PI_Intitule() As String	Intitulé.
Lecture / Ecriture	PI_Raccourci() As String	Raccourci.

IBOModeleEcritureLigne3

Ligne de modèle de saisie d'écritures générales.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	CalculDevise() As Boolean	Si le champ Devise est positionné sur Calcul alors : **True** Si le champ Devise est positionné sur Aucune alors : **False**
Lecture / Ecriture	CalculReglement() As Boolean	Si le champ Reglement est positionné sur Calcul alors : **True** Si le champ Reglement est positionné sur Aucune alors : **False**
Lecture / Ecriture	CalculTA_Provenance() As Boolean	Si le champ Provenance est positionné sur Calcul alors : **True** Si le champ Provenance est positionné sur Aucune alors : **False**
Lecture / Ecriture	CG_Num() As String	Numéro de compte général ou fonction : **=MacroSaisir()** **=MacroRepeter()** **=MacroCalculer()**
Lecture / Ecriture	CG_NumCont() As String	Numéro de compte général de contrepartie ou fonction : **=MacroSaisir()** **=MacroRepeter()** **=MacroCalculer()**
Lecture / Ecriture	CT_Num() As String	Numéro de compte tiers ou fonction :

Accès	Syntaxe	Description
		=MacroSaisir() **=MacroRepeter()** **=MacroCalculer()**
Lecture / Ecriture	CT_NumCont() As String	Numéro de compte tiers de contrepartie ou fonction : **=MacroSaisir()** **=MacroRepeter()** **=MacroCalculer()**
Lecture / Ecriture	Devise() As IBPDevise2	Devise.
Lecture seule	FactoryModeleEcritureLigne() As IBITypeObjectFactory	Fabrique un objet Ligne d'écriture général de modèle de saisie.
Lecture seule	FactoryModeleEcritureLigneA() As IBITypeObjectFactory	Fabrique un objet Ligne d'écriture analytique de modèle de saisie.
Lecture / Ecriture	HasTA_Provenance() As Boolean	Si la ligne a une provenance taxe : **True** Sinon : **False**
Lecture / Ecriture	InfoLibre(ByVal *sElt* As Short) As String	Retourne la Nieme information libre (*Short* passé en paramètre) sous forme d'une chaîne de caractères.
Lecture seule	ModeleEcriture() As IBOModeleEcriture3	Entête du modèle de saisie auquel est associée cette ligne de modèle de saisie d'écritures générales.
Lecture / Ecriture	PG_Devise() As String	Montant en Devise ou fonction : **=MacroSaisir()** **=MacroEquilibrer()**
Lecture / Ecriture	PG_Echeance() As String	Echéance ou fonction : **= MacroSaisir()** **=MacroRepeter()** **=MacroCalculer()**
Lecture / Ecriture	PG_Intitule() As String	Intitulé ou fonction : **=MacroSaisir()** **=MacroRepeter()**
Lecture / Ecriture	PG_Jour() As String	Jour ou fonction : **=MacroSaisir()** **=MacroRepeter()**
Lecture / Ecriture	PG_Montant() As String	Montant ou fonction : **=MacroSaisir()** **=MacroEquilibrer()** **=MacroCalculer()**
Lecture / Ecriture	PG_Parite() As String	Parité ou fonction :

Accès	Syntaxe	Description
		=MacroSaisir() **=MacroRepeter()** **=MacroCalculer()**
Lecture / Ecriture	PG_Piece() As String	Pièce ou fonction : **=MacroSaisir()** **=MacroRepeter()** **=MacroIncrementer()** **=MacroCalculer()**
Lecture / Ecriture	PG_Quantite() As String	Quantité ou fonction : **=MacroSaisir()** **=MacroEquilibrer()**
Lecture/ Ecriture	PG_Reference() As String	Référence ou fonction : **=MacroSaisir()** **=MacroRepeter()** **=MacroIncrementer()** **=MacroCalculer()**
Lecture / Ecriture	PG_RefPiece() As String	Référence pièce ou fonction : **=MacroSaisir()** **=MacroRepeter()** **=MacroIncrementer()** **=MacroCalculer()**
Lecture / Ecriture	PG_Sens() As EcritureSensType	Sens (Cf. énumérateur EcritureSensType).
Lecture / Ecriture	Reglement() As IBPReglement3	Règlement.
Lecture / Ecriture	TA_Code() As String	Code taxe ou fonction : **=MacroSaisir()** **=MacroRepeter()** **=MacroCalculer()**
Lecture / Ecriture	TA_Provenance() As TaxeProvenanceType	Provenance taxe (Cf. énumérateur TaxeProvenanceType).

Particularités : les commandes prédéfinies et les fonctions

Depuis Sage 100c Comptabilité, les commandes prédéfinies du modèle de saisie sont représentées sous la forme d'icônes.

Les Objets Métiers proposent des fonctions équivalentes :

Commandes prédéfinies	Fonctions Objets Métier	Description
	=MacroSaisir()	Saisie manuelle
	=MacroRepeter()	Répétition
	=MacroCalculer()	Calcul
	=MacroEquilibrer()	Equilibre
	=MacroIncrementer()	Incrémentation

Les Objets Métiers mettent à disposition les fonctions suivantes pour l'ensemble des champs :

Syntaxe	Description
=Date_E()	Date d'écriture
=Date_S()	Date de saisie
=Piece()	N° pièce
=Ref_P	Référence pièce
=Cpte_G()	N° compte général
=Cpte_T()	N° compte tiers
=Contre_G	Contrepartie général
=Contre_T	Contrepartie tiers
=Code_Taxe	Code taxe
=Provenance()	Provenance
=Libelle()	Libellé écriture
=Echeance()	Date d'échéance
=Mode()	Intitulé mode de règlement
=Devise()	Intitulé devise
=Parite()	Parité
=Qte()	Quantité
=Mont_Dev()	Montant devise
=Debit()	Montant débit
=Credit()	Montant crédit

Syntaxe	Description
=Montant()	Montant débit ou crédit
=Int_T()	Intitulé du compte tiers
=Int_G()	Intitulé du compte général
=CDText()	Transformation d'une date en texte

IBOModeleEcritureLigneA3

Ligne de modèle de saisie d'écritures analytiques.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Analytique() As IBPAnalytique3	Plan analytique.
Lecture / Ecriture	CA_Num() As String	Section analytique ou fonction : =MacroSaisir()
Lecture seule	FactoryModeleEcritureLigneA() As IBITypeObjectFactory	Fabrique un objet Ligne de saisie d'écriture analytique de modèle d'enregistrement.
Lecture seule	ModeleEcritureLigne() As IBOModeleEcritureLigne3	Modèle de saisie auquel est associée la ligne de saisie d'écritures analytiques.
Lecture / Ecriture	PA_Montant() As String	Montant ou fonction : =MacroSaisir() =MacroSaisirPourcentage() =MacroEquilibrer()
Lecture / Ecriture	PA_Quantite() As String	Quantité ou fonction : =MacroSaisir() =MacroSaisirPourcentage() =MacroEquilibrer()

IBOModeleGrille

Modèle de grille.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	FactoryModeleGrille() As IBOModeleGrilleFactory	Fabrique un objet modèle de grille.
Lecture seule	FactoryModeleGrilleLigne() As IBOModeleGrilleLigneFactory	Fabrique un objet ligne de modèle de grille.
Lecture/ Ecriture	Intitule() As String	Intitulé du modèle de grille.
Lecture/ Ecriture	Raccourci() As String	Raccourci du modèle de grille.
Lecture/ Ecriture	Type() As ModeleGrilleType	Type de modèle de grille (cf. énumérateur ModeleGrilleType).

IBOModeleGrilleLigne

Ligne de modèle de grille.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture/ Ecriture	CompteA() As IBOCompteA3	Section analytique.
Lecture/ Ecriture	CompteG() As IBOCompteG3	Compte général.
Lecture/ Ecriture	EG_TRepart() As RepartitionType	Type de répartition (cf. énumérateur RepartitionType).
Lecture/ Ecriture	EG_VRepart() As Double	Valeur de répartition.
Lecture seule	FactoryModeleGrilleLigne() As IBOModeleGrilleLigneFactory	Fabrique un objet ligne de modèle de grille.
Lecture/ Ecriture	ModeleGrille() As IBOModeleGrille	Modèle de grille.

IBOModeleReglement

Modèle de règlement.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	FactoryModeleReglement() As IBOModeleReglementFactory	Fabrique un objet modèle de règlement.
Lecture seule	FactoryModeleReglementLigne() As IBITypeObjectFactory	Fabrique un objet ligne de modèle de règlement.
Lecture/ Ecriture	Intitule() As String	Intitulé du modèle de règlement.

IBOModeleReglementLigne

Ligne de modèle de règlement.

Interface héritée

Syntaxe	Description
IBIReglement	Cf. Interface IBIReglement pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	FactoryModeleReglementLigne() As IBITypeObjectFactory	Fabrique un objet ligne de modèle de règlement.
Lecture seule	ModeleReglement() As IBOModeleReglement	Modèle de règlement.

IBOPays

Pays.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Intitule() As String	Intitulé du pays.

Accès	Syntaxe	Description
Lecture / Ecriture	Code() As String	Code du pays.
Lecture / Ecriture	CodeEdi() As String	Code EDI.
Lecture / Ecriture	Assurance() As Double	Frais assurance.
Lecture / Ecriture	Transport() As Double	Frais de transport.
Lecture / Ecriture	CodeISO2() As String	Code ISO2.
Lecture / Ecriture	IsSEPA() As Boolean	Zone SEPA.

IBOTaxe3

Taxe.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	CompteG() As IBOCompteG3	Compte général de taxe.
Lecture seule	FactoryTaxe() As IBOTaxeFactory3	Fabrique un objet Taxe.
Lecture / Ecriture	TA_Assujet() As Double	Assujettissement (%).
Lecture / Ecriture	TA_Code() As String	Code taxe.
Lecture / Ecriture	TA_GrilleBase() As String	Grille de base (spécifique à la Comptabilité Belge).
Lecture / Ecriture	TA_GrilleTaxe() As String	Grille de taxe (spécifique à la Comptabilité Belge).
Lecture / Ecriture	TA_Intitule() As String	Intitulé.
Lecture / Ecriture	TA_NP() As Boolean	Taxe non perçue (True) ou taxe perçue (False).
Lecture / Ecriture	TA_Provenance() As TaxeProvenanceType	Provenance (Cf. énumérateur TaxeProvenanceType).
Lecture / Ecriture	TA_Regroup() As String	Code regroupement.
Lecture / Ecriture	TA_Sens() As TaxeSensType	Sens (Cf. énumérateur TaxeSensType).
Lecture / Ecriture	TA_Taux() As Double	Taux.
Lecture / Ecriture	TA_TTaux() As TaxeTauxType	Type taux (Cf. énumérateur TaxeTauxType).
Lecture / Ecriture	TA_Type() As TaxeType	Type taxe (Cf. énumérateur TaxeType).

Traitements et initialisations par défaut

Méthode SetDefault()

Si TA_Assujet est null, alors TA_Assujet = 100.

IBOTiers3

Tiers.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	ActivateGDPR() As Boolean	Protection des données personnelles.
Lecture / Ecriture	Adresse() As IAdresse	Informations d'adresse.
Lecture / Ecriture	BanquePrincipale() As IBOTiersBanque3	Banque principale.
Lecture / Ecriture	CompteAIFRS() As IBOCompteA3	Compte analytique IFRS.
Lecture / Ecriture	CompteGPrinc() As IBOCompteG3	Compte général principal.
Lecture / Ecriture	CT_Ape() As String	Code APE (NAF).
Lecture / Ecriture	CT_Classement() As String	Classement.
Lecture / Ecriture	CT_Commentaire() As String	Commentaire.
Lecture / Ecriture	CT_Contact() As String	Contact.
Lecture seule	CT_DateCreate() As Date	Date de création.
Lecture / Ecriture	CT_EdiCodeType() As EdiCodeType	Type de code EDI.
Lecture / Ecriture	CT_EdiCode() As String	Code EDI.
Lecture / Ecriture	CT_EMail() As String	E-Mail.
Lecture / Ecriture	CT_Identifiant() As String	Numéro identifiant.
Lecture / Ecriture	CT_Intitule() As String	Intitulé.
Lecture / Ecriture	CT_Langue() As LangueType	Langue (Cf. énumérateur LangueType).
Lecture / Ecriture	CT_Lettrage() As Boolean	Lettrage automatique (True) ou non (False).
Lecture / Ecriture	CT_NotRappel() As Boolean	Hors rappel / relevé (True) ou non (False).

Accès	Syntaxe	Description
Lecture / Ecriture	CT_Num() As String	Numéro de compte.
Lecture / Ecriture	CT_Qualite() As String	Qualité.
Lecture / Ecriture	CT_Raccourci() As String	Raccourci.
Lecture / Ecriture	CT_SautLigne() As Short	Saut de ligne (nombre de lignes).
Lecture / Ecriture	CT_SautPage() As Boolean	Saut de page (True) ou saut de ligne (False).
Lecture / Ecriture	CT_Siret() As String	Numéro de Siret.
Lecture / Ecriture	CT_Sommeil() As Boolean	Mis en sommeil (True) ou actif (False).
Lecture / Ecriture	CT_Stat (ByVal *sElt* As Short) As String	Enuméré correspondant au N ieme champ statistique.
Lecture seule	CT_Type() As TiersType	Type tiers (Cf. énumérateur TiersType).
Lecture / Ecriture	CT_ValidEch() As Boolean	Validation automatique des règlements (True) ou non (False).
Lecture / Ecriture	Devise() As IBPDevise2	Devise.
Lecture / Ecriture	ExclureMarketing() As Boolean	Exclure des traitements à des fins marketing.
Lecture seule	FactoryTiers() As IBOTiersFactory3	Fabrique un objet Tiers.
Lecture seule	FactoryTiersBanque() As IBITypeObjectFactory	Fabrique un objet Banque tiers.
Lecture seule	FactoryTiersCompteG() As IBOTiersCompteGFactory3	Fabrique un objet Compte général tiers.
Lecture seule	FactoryTiersContact() As IBITypeObjectFactory	Fabrique un objet Contact tiers.
Lecture seule	FactoryTiersMedia() As IBITypeObjectFactory	Fabrique un objet Média tiers.
Lecture seule	FactoryTiersReglement() As IBITypeObjectFactory	Fabrique un objet règlement tiers.
Lecture seule	InfoLibre() As IBIValues	Valeur information libre.
Lecture / Ecriture	ModeReglement() As IBOModeleReglement	Modèle de règlement.
Lecture / Ecriture	Telecom() As ITelecom	Télécommunications.
Lecture / Ecriture	CT_Facebook() As String	Compte Facebook.
Lecture / Ecriture	CT_LinkedIn() As String	Compte LinkedIn.

Méthodes

Syntaxe	Description
CleanPersonalData()	Suppression des données personlles.

Syntaxe	Description
MvtCredit(ByVal *pCompteG* As IBOCompteG3, ByVal *pJournal* As IBOJournal3, ByVal *dDebut* As Date, ByVal *dFin* As Date) As Double	Montant total des mouvements créditeurs du tiers pour un compte général, un journal et une période donnés.
MvtDebit(ByVal *pCompteG* As IBOCompteG3, ByVal *pJournal* As IBOJournal3, ByVal *dDebut* As Date, ByVal *dFin* As Date) As Double	Montant total des mouvements débiteurs du tiers pour un compte général, un journal et une période donnés.
MvtDevCredit(ByVal *pCompteG* As IBOCompteG3, ByVal *pJournal* As IBOJournal3, ByVal *dDebut* As Date, ByVal *dFin* As Date) As Double	Montant total des mouvements créditeurs en devise du tiers pour un compte général, un journal et une période donnés.
MvtDevDebit(ByVal *pCompteG* As IBOCompteG3, ByVal *pJournal* As IBOJournal3, ByVal *dDebut* As Date, ByVal *dFin* As Date) As Double	Montant total des mouvements débiteurs en devise du tiers pour un compte général, un journal et une période donnés.
MvtDevSolde(ByVal *pCompteG* As IBOCompteG3, ByVal *pJournal* As IBOJournal3, ByVal *dDebut* As Date, ByVal *dFin* As Date) As Double	Montant total du solde des mouvements en devise du tiers pour un compte général, un journal et une période donnés.
MvtSolde(ByVal *pCompteG* As IBOCompteG3, ByVal *pJournal* As IBOJournal3, ByVal *dDebut* As Date, ByVal *dFin* As Date) As Double	Montant total du solde des mouvements du tiers pour un compte général, un journal et une période donnés.
Solde() As Double	Solde.

Traitements et initialisations par défaut

Méthode SetDefault()

L'appel de la méthode *SetDefault()* initialise les propriétés suivantes :

Propriété	Valeur	Description
CT_DateCreation() As Date	Si CT_DateCreation="" Alors CT_DateCreation=Date du jour	Affectation de la date de création si celle-ci n'est pas renseignée.
CT_Intitule() As String	Si CT_Intitule = "" Alors CT_Intitule = CT_Num	Affectation de la propriété Numéro du tiers à la propriété Intitulé du tiers si celle-ci n'est pas renseignée.
TiersPayeur() As IBOTiers3	Si TiersPayeur = Nothing Alors TiersPayeur = Tiers	Affectation de l'objet Tiers à la propriété TiersPayeur si celle-ci n'est pas renseignée.
CompteGPrinc() As IBOCompteG3	Si CompteGPrinc = Nothing Alors CompteGPrinc = Compte collectif par défaut	Affectation du compte collectif par défaut (situé dans Paramètres société / Tiers / Compte collectif) à la propriété Compte général principal si celle-ci n'est pas renseignée.

Méthode Write()

La méthode *Write()* enregistre l'objet dans la base de données après avoir créé le Compte général principal s'il n'existe pas dans la table F_COMPTEG.

Méthode WriteDefault()

La méthode *WriteDefault()* enregistre l'objet dans la base de données après avoir créé le Compte général principal s'il n'existe pas dans la table F_COMPTEG.

IBOTiersBanque3

Banque tiers.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Adresse() As IAdresse	Informations d'adresse.
Lecture / Ecriture	BT_Banque() As String	Code banque.
Lecture / Ecriture	BT_Bic() As String	Code BIC.
Lecture / Ecriture	BT_CalculIBAN() As Boolean	Calcul automatique de l'IBAN.
Lecture / Ecriture	BT_Cle() As String	Clé RIB.
Lecture / Ecriture	BT_Commentaire() As String	Commentaire.
Lecture / Ecriture	BT_Compte() As String	Numéro de compte bancaire.
Lecture / Ecriture	BT_Guichet() As String	Code guichet.
Lecture / Ecriture	BT_IBAN() As String	IBAN.
Lecture / Ecriture	BT_Intitule() As String	Intitulé.
Lecture / Ecriture	BT_MandatDate() As Date [1]	Date de mandat.
Lecture / Ecriture	BT_MandatRef() As String [1]	Référence du mandat.
Lecture / Ecriture	BT_NomAgence() As String	Nom de l'agence.
Lecture / Ecriture	BT_PaysAgence() As String	Pays agence.
Lecture / Ecriture	BT_Struct() As RibType	Structure banque (Cf. énumérateur RibType).
Lecture / Ecriture	Devise() As IBPDevise2	Devise.
Lecture seule	FactoryTiersBanque() As IBITypeObjectFactory	Fabrique un objet Banque tiers.
Lecture seule	Tiers() As IBOTiers3	Tiers auquel est associée la banque tiers.

[1] : Propriétés dépréciées. Elles sont conservées pour assurer la compatibilité avec les développements existants, cependant la valeur retournée est vide.

IBOTiersContact3

Contact tiers.

Interface héritée

Syntaxe	Description
IBIContact2	Cf. Interface IBIContact2 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	FactoryTiersContact() As IBITypeObjectFactory	Fabrique un objet Contact tiers.
Lecture seule	Tiers() As IBOTiers3	Tiers auquel est associé le contact.

IBOTiersMedia3

Média tiers.

Interface héritée

Syntaxe	Description
IBIMedia	Cf. Interface IBIMedia pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	FactoryTiersMedia() As IBITypeObjectFactory	Fabrique un objet Média tiers.
Lecture seule	Tiers() As IBOTiers3	Tiers auquel est associé le média tiers.

IBOTiersPart3

Tiers de type Client ou Fournisseur.

Interface héritée

Syntaxe	Description
IBOTiers3	Cf. Interface IBOTiers3 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	CategorieCompta() As IBICategorieCompta	Catégorie comptable. *
Lecture / Ecriture	CompteA() As IBOCompteA3	Compte analytique.
Lecture / Ecriture	CT_Coface() As String	Numéro Coface SCRL.
Lecture / Ecriture	CT_DateFermetureDebut() As Date	Date de début de fermeture
Lecture / Ecriture	CT_DateFermetureFin() As Date	Date de fin de fermeture
Lecture / Ecriture	CT_Encours() As Double	Encours maximum autorisé.
Lecture / Ecriture	CT_Surveillance() As Boolean	Placé sous surveillance (True) ou non (False).
Lecture seule	CT_SvCA() As Double	Chiffre d'affaires.
Lecture seule	CT_SvCotation() As String	Cotation de la solvabilité.
Lecture seule	CT_SvDateBilan() As Date	Date d'arrêté de bilan.
Lecture seule	CT_SvDateCreate() As Date	Date de création de la société.
Lecture seule	CT_SvDateIncid() As Date	Date du dernier incident de paiement.
Lecture seule	CT_SvDateMaj() As Date	Date de la dernière mise à jour.
Lecture seule	CT_SvEffectif() As String	Effectif.
Lecture seule	CT_SvFormeJuri() As String	Forme juridique.
Lecture seule	CT_SvIncident() As Short	Incidents de paiement.
Lecture seule	CT_SvNbMoisBilan() As Short	Nombre de mois du bilan.
Lecture seule	CT_SvObjetMaj() As String	Objet de la dernière mise à jour.
Lecture seule	CT_SvPrivil() As Short	Privilèges.
Lecture seule	CT_SvRegul() As String	Régularité des paiements.
Lecture seule	CT_SvResultat() As Double	Résultat net.
Lecture / Ecriture	Depot() As IBODepot3	Dépôt. *
Lecture / Ecriture	TauxEscompte() As Double	Taux escompte.
Lecture / Ecriture	TauxRemise() As Double	Taux remise.
Lecture / Ecriture	TauxReleve() As Double	Taux relevé.
Lecture / Ecriture	TauxRFA() As Double	Taux RFA.

Accès	Syntaxe	Description
Lecture / Ecriture	TiersPayeur() As IBOTiers3	Tiers Payeur.

** : nécessite l'ouverture d'une base de Gestion Commerciale*

Méthodes

Syntaxe	Description
CA_HTBrut(ByVal *dDebut* As Date, ByVal *dFin* As Date, ByVal *TypeDe* As DocumentType, ByVal *TypeA* As DocumentType) As Double	Chiffre d'affaires HT brut pour une période donnée et une plage de types de document. *
CA_HTNet(ByVal *dDebut* As Date, ByVal *dFin* As Date, ByVal *TypeDe* As DocumentType, ByVal *TypeA* As DocumentType) As Double	Chiffre d'affaires HT net pour une période donnée et une plage de types de document. *
CA_TTC(ByVal *dDebut* As Date, ByVal *dFin* As Date, ByVal *TypeDe* As DocumentType, ByVal *TypeA* As DocumentType) As Double	Chiffre d'affaires TTC pour une période donnée et une plage de types de document. *
DepassementEncours() As Double	Montant du dépassement de l'encours.
DepassementEncours2() As Double	Montant du dépassement de l'encours en tenant compte du portefeuille BL, FA et des règlements.
DerniereFacture() As Date	Date de la dernière facture.
DernierReglement() As Date	Date du dernier règlement.
Portefeuille() As Double	Portefeuille BL & FA. *
PortefeuilleCommande() As Double	Portefeuille BC. *
SoldeEchus1() As Double	Solde échu à 1 mois.
SoldeEchus2() As Double	Solde échu à 2 mois.
SoldeEchus3() As Double	Solde échu à 3 mois.
SoldeNonEchus() As Double	Solde non échus.
TauxRemiseMoyen(ByVal *dDebut* As Date, ByVal *dFin* As Date, ByVal *TypeDe* As DocumentType, ByVal *TypeA* As DocumentType) As Double	Taux de remise moyen pour une période donnée et une plage de types de document. *
TotalPoidsBrut(ByVal *dDebut* As Date, ByVal *dFin* As Date, ByVal *TypeDe* As DocumentType, ByVal *TypeA* As DocumentType) As Double	Total poids brut pour une période donnée et une plage de types de document. *
TotalPoidsNet(ByVal *dDebut* As Date, ByVal *dFin* As Date, ByVal *TypeDe* As DocumentType, ByVal *TypeA* As DocumentType) As Double	Total poids net pour une période donnée et une plage de types de document. *
TotalPrixRevient(ByVal *dDebut* As Date, ByVal *dFin* As Date, ByVal *TypeDe* As DocumentType, ByVal *TypeA* As DocumentType) As Double	Total prix de revient pour une période donnée et une plage de types de document. *
TotalQtes(ByVal *dDebut* As Date, ByVal *dFin* As Date, ByVal *TypeDe* As DocumentType, ByVal *TypeA* As DocumentType) As Double	Total quantités pour une période donnée et une plage de types de document. *

Syntaxe	Description
TotalQtesColises(ByVal *dDebut* As Date, ByVal *dFin* As Date, ByVal *TypeDe* As DocumentType, ByVal *TypeA* As DocumentType) As Double	Total quantités colisées pour une période donnée et une plage de types de document. *
TotalReglement() As Double	Total règlements. *

** : nécessite l'ouverture d'une base de Gestion Commerciale*

IBOTiersReglement3

Règlement tiers.

Interface héritée

Syntaxe	Description
IBIReglement	Cf. Interface IBIReglement pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	FactoryTiersReglement() As IBITypeObjectFactory	Fabrique un objet Règlement tiers.
Lecture seule	Tiers() As IBOTiers3	Tiers auquel est associé le Règlement tiers.

Interfaces Application Commerciale : Objets100c.CIAL

Classe d'application / BSCIALApplication100c

Accès à la base commeriale.

Interfaces héritées

Syntaxe	Description
IBILoggable	Cf. Interface IBILoggable pour les propriétés et méthodes héritées.
IBIPersistStream	Cf. Interface IBIPersistStream pour les propriétés et méthodes héritées.
IBSCIALApplication100c	Cf. Interface IBSCIALApplication100c pour les propriétés et méthodes héritées.

Interface Stream / IBSCIALApplication100c

Base de données commerciale.

Interface héritée

Syntaxe	Description
IBIPersistStream	Cf. Interface IBIPersistStream pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Companies() As ICompanies	Liste des bases Sage 100c.
Lecture / Ecriture	CompanyServer() As String	Nom du serveur/instance SQL.
Lecture / Ecriture	CompanyDatabaseName() As String	Nom de la base SQL.
Lecture / Ecriture	CptaApplication() As BSCPTAApplication100c	Base de données comptable.
Lecture seule	DatabaseInfo() As IDatabaseInfo	Informations sur la base de données.
Lecture / Ecriture	DepotPrincipal() As IBODepot2	Dépôt principal.
Lecture seule	FactoryAgenda() As IBOAgendaFactory	Fabrique un objet Agenda.
Lecture seule	FactoryAgendaConfig As IBPAgendaFactory	Fabrique un objet configuration Agenda.
Lecture seule	FactoryArrondi() As IBITypeObjectFactory	Fabrique un objet Arrondi.
Lecture seule	FactoryArticle() As IBOArticleFactory3	Fabrique un objet Article.
Lecture seule	FactoryArticleDepotLot() As IBOArticleDepotLotAllFactory	Accès aux lots sans passer par l'article et le dépôt.
Lecture seule	FactoryArticleStat() As IBPArticleStatFactory	Fabrique un objet Statistique article.
Lecture seule	FactoryBaremeCommision() As IBOBaremeCommissionFactory	Fabrique un objet Commission.
Lecture seule	FactoryBaremeRabais() As IBOBaremeRabaisFactory	Fabrique un objet Rabais, remise et ristournes.
Lecture seule	FactoryBaremeRabaisFrs() As IBOBaremeRabaisFactory	Fabrique un objet Rabais, remise et ristournes fournisseur.
Lecture seule	FactoryBaremeSolde() As IBOBaremeSoldeFactory	Fabrique un objet Soldes et promotions.
Lecture seule	FactoryBaremeSoldeFrs() As IBOBaremeSoldeFactory	Fabrique un objet Soldes et promotions fournisseur.
Lecture seule	FactoryCategorieComptaAchat() As IBPCategorieComptaAchatFactory	Fabrique un objet Catégorie comptable des achats.
Lecture seule	FactoryCategorieComptaStock() As IBPCategorieComptaStockFactory	Fabrique un objet Catégorie comptable des stocks.

Accès	Syntaxe	Description
Lecture seule	FactoryCategorieComptaVente() As IBPCategorieComptaVenteFactory	Fabrique un objet Catégorie comptable des ventes.
Lecture seule	FactoryCategorieTarif() As IBPCategorieTarifFactory	Fabrique un objet Catégorie tarifaire.
Lecture seule	FactoryConditionLivraison() As IBPConditionLivraisonFactory	Fabrique un objet Condition de livraison.
Lecture seule	FactoryConditionnement() As IBPConditionnementFactory	Fabrique un objet Conditionnement.
Lecture seule	FactoryDepot() As IBODepotFactory2	Fabrique un objet Dépôt de stockage.
Lecture seule	FactoryDocument() As IBODocumentFactory	Fabrique un objet Document.
Lecture seule	FactoryDocumentAchat() As IBODocumentAchatFactory3	Fabrique un objet Document des achats.
Lecture seule	FactoryDocumentInterne() As IBODocumentInterneFactory3	Fabrique un objet Document interne.
Lecture seule	FactoryDocumentLigne() As IBODocumentLigneAllFactory	Fabrique un objet ligne de document..
Lecture seule	FactoryDocumentReglement() As IBODocumentReglement	Fabrique un objet règlement de document..
Lecture seule	FactoryDocumentStock() As IBODocumentStockFactory3	Fabrique un objet Document des stocks.
Lecture seule	FactoryDocumentVente() As IBODocumentVenteFactory3	Fabrique un objet Document des ventes.
Lecture seule	FactoryDossier() As IBITypeObjectFactory	Fabrique un objet Dossier commercial.
Lecture seule	FactoryDossierParam() As IBITypeObjectFactory	Fabrique un objet un objet Paramétrage dossier commercial.
Lecture seule	FactoryExpedition() As IBPExpeditionFactory3	Fabrique un objet Mode d'expédition.
Lecture seule	FactoryFamille() As IBOFamilleFactory2	Fabrique un objet Famille.
Lecture seule	FactoryGamme() As IBPGammeFactory	Fabrique un objet Gamme.
Lecture seule	FactoryGlossaire() As IBOGlossaireFactory	Fabrique un objet Glossaire.
Lecture seule	FactoryModele() As IBOModeleFactory	Fabrique un objet Modèle d'enregistrement.
Lecture seule	FactoryParamDocAchat() As IBIParamDocFactory	Fabrique un objet paramétrage des documents d’achat.
Lecture seule	FactoryParamDocInterne() As IBIParamDocFactory	Fabrique un objet paramétrage des documents de vente.
Lecture seule	FactoryParamDocStock() As IBIParamDocFactory	Fabrique un objet paramétrage des documents de stock.
Lecture seule	FactoryParamDocVente() As IBIParamDocFactory	Fabrique un objet paramétrage des documents interne.

Accès	Syntaxe	Description
Lecture seule	FactoryPeriodicite() As IBPPeriodiciteFactory	Fabrique un objet Périodicité.
Lecture seule	FactoryProduit() As IBPProduitFactory2	Fabrique un objet catalogue de produits.
Lecture seule	FactoryRessource() As IBORessourceFactory	Fabrique un objet Ressource.
Lecture seule	FactoryRessourceBase() As IBIRessourceFactory	Fabrique un objet Ressource de base.
Lecture seule	FactoryRessourceCentre() As IBORessourceCentreFactory	Fabrique un objet Ressource centralisatrice.
Lecture seule	FactorySoucheAchat() As IBISoucheFactory	Fabrique un objet souche de document de d'achat.
Lecture seule	FactorySoucheInterne() As IBISoucheFactory	Fabrique un objet souche de document de interne.
Lecture seule	FactorySoucheStock() As IBISoucheFactory	Fabrique un objet souche de document de stock.
Lecture seule	FactorySoucheVente() As IBISoucheFactory	Fabrique un objet souche de document de vente.
Lecture seule	FactoryUnite() As IBPUniteFactory	Fabrique un objet Unité d'achat et de vente.
Lecture seule	Licence() As ILicence	Licence.
Lecture seule	Loggable() As IBILoggable	Autorisations d'accès à la base de données.
Lecture / Ecriture	Name() As String	Chemin et nom de la base de données.
Lecture seule	Transformation() As ITransformation	Accès au processus de transformation de document d'achat ou vente.

Méthodes

Syntaxe	Description
ControlQualiteStat (ByRef pArticle As IBOArticle3, ByRef pFournisseur As IBOFournisseur3, ByVal DateDe As Date, ByVal DateA As Date) As IControleQualiteStat	Retourne les statistiques du contrôle qualité pour l'article, le fournisseur et la période passés en paramètres.
ControlQualiteStatDoubleGamme (ByRef pArtGammeEnum1 As IBOArticleGammeEnum3, ByRef pArtGammeEnum2 As IBOArticleGammeEnum3, ByRef pFournisseur As IBOFournisseur3, ByVal DateDe As Date, ByVal DateA As Date) As IControleQualiteStat	Retourne les statistiques du contrôle qualité pour les énumérés de gamme, le fournisseur et la période passés en paramètres.

Syntaxe	Description
ControlQualiteStatMonoGamme (ByRef pArtGammeEnum As IBOArticleGammeEnum3, ByRef pFournisseur As IBOFournisseur3, ByVal DateDe As Date, ByVal DateA As Date) As IControleQualiteStat	Retourne les statistiques du contrôle qualité pour l'énuméré de gamme, le fournisseur et la période passés en paramètres.
CreateProcess_AppliquerBareme(ByRef pDoc As IBODocumentPart3) As IPMAppliquerBareme	Créé un processus d'application des barêmes.
CreateProcess_Coliser() As IPMColiser	Créé un processus de colisage.
CreateProcess_ControleQualite() As IPMControlerQualite	Créé un processus de contrôle qualité.
CreateProcess_DocTransferer() As IPMDocTransferer	Créé un processus de transfert d'un article d'un dépôt/emplacement vers un autre.
CreateProcess_Document(ByVal DO_Type As DocumentType) As IPMDocument	Créé un processus Document.
CreateProcess_PreleverLot() As IPMPreleverLot	Créé un processus de prélèvement de série/lot.
CreateProcess_SousTotal(ByRef pDoc As IBODocumentPart3) As IPMDocInsererSousTotal	Créé un processus d'insertion de ligne de sous-total.
CreateProcess_RecalculPrixCompose(ByRef pDoc As IBODocument3) As IPMDocRecalculPrixCompose	Créé un processus de recalcul de prix de revient d'une nomenclature fabrication.
CreateProcess_SortirLots() As IPMSortirLots	Créé un processus de sortie d'un article géré par lot.
CreateProcess_ConversionClient(*pClient* As IBOClient3) As IPMConversionClient	Créé un processus de transformation d'un prospect en client.
CreateProcess_ReglerEcheances () As IPMReglerEcheances	Crée un processus de règlement d'échéances. Demande un Règlement existant. Associe plusieurs échéances d'un même document pour le même client. Possibilité d'ajouter les échéances avec un montant autre que celui de l'échéance.
CreateProcess_Inventaire () As IPMInventaire	Crée un processus d'Inventaire à une date. Permet de corriger la quantté et la valeur d'articles pour un dépôt et un emplacement

Interfaces métiers (Ixxx, IBIxxx et IDOCxxx)

IArrondi

Arrondi la valeur d'un tarif de vente (Cf. ITarif Vente).

Propriétés

Accès	Syntaxe	Description
Lecture seule	AR_Type() As ArrondiType	Type de l'arrondi (cf. énumérateur ArrondiType).
Lecture seule	AR_Valeur() As Double	Valeur de l'arrondi.

Méthodes

Syntaxe	Description
Arrondir(ByVal dVal As double) As Double	Retourne la valeur passée en paramètre après l'avoir arrondie.

IBIArticleStock3

Informations de stock article.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Article() As IBOArticle3	Article auxquelles sont associées les informations de stock.
Lecture seule	CMUP() As Double	CMUP du stock.
Lecture / Ecriture	EmplacementControle() As IBODepotEmplacement	Emplacement de contrôle par défaut.
Lecture / Ecriture	EmplacementPrincipal() As IBODepotEmplacement	Emplacement principal par défaut.
Lecture seule	FactoryArticleStockEmpl() As IBIArticleStockEmplFactory	Fabrique un objet emplacement de stock.
Lecture seule	Montant() As Double	Valeur du stock.

Méthodes

Syntaxe	Description
StockAControler() As Double	Stock sur emplacement de contrôle
StockATerme() As Double	Stock à terme.
StockCommande() As Double	Stock commandé.
StockCommandeContremarque() As Double	Stock commandé contremarque.
StockDispo() As Double	Stock disponible.

Syntaxe	Description
StockPrepare() As Double	Stock préparé.
StockQte(ByVal DateStock As Date) As Double	Quantité en stock à la date indiquée.
StockReel() As Double	Stock réel.
StockReserve() As Double	Stock réservé.
StockReserveContremarque() As Double	Stock réservé contremarque.
StockValeur(ByVal DateStock As Date) As Double	Valeur du stock à la date indiquée.

IBIArticleStockEmpl

Stock article par emplacement.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	ArticleStock() As IBIArticleStock3	Stock article de l'emplacement.
Lecture seule	Emplacement() As IBODepotEmplacement	Emplacement.

Méthodes

Syntaxe	Description
StockQte(ByVal DateStock As Date) As Double	Quantité en stock à la date indiquée.
StockReel() As Double	Stock réel.
StockATerme() As Double	Stock à terme.
StockPrepare() As Double	Stock préparé.
StockDispo() As Double	Stock disponible.

IBIArticleTarif3

Tarif de l'article.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Article() As IBOArticle3	Article auquel est associé le tarif.
Lecture / Ecriture	Devise() As IBPDevise2	Devise.
Lecture seule	FactoryArticleTarifGamme() As IBITypeObjectFactory	Fabrique un objet tarif par énuméré de gamme.
Lecture seule	FactoryArticleTarifQte() As IBITypeObjectFactory	Fabrique un objet tarif par quantité, montant ou prix net.
Lecture / Ecriture	GammeRemise() As IBPGamme	Gamme de remise.
Lecture / Ecriture	IsHorsRemise() As Boolean	Hors remise.
Lecture / Ecriture	Prix() As Double	Prix.
Lecture / Ecriture	PrixDevise() As Double	Prix en devise.
Lecture / Ecriture	Remise() As IRemise2	Remise.

Traitements et initialisations par défaut

Méthode SetDefault()

Si l'article possède une gamme de remise (propriété *GammeRemise()* différente de *Nothing*), l'appel de la méthode *SetDefault()* initialise les propriétés suivantes :

Propriété	Valeur	Description
IsHorsRemise() As Boolean	False	Tarif remisé.
Remise() As IRemise2	Nothing	Pas de remise hors gamme.

Méthode Write()

La méthode *Write()* enregistre l'objet dans la base de données sans effectuer aucun traitement d'initialisation des propriétés de l'objet.

Méthode WriteDefault()

La méthode WriteDefault() contrôle s'il existe une gamme de remise (propriété *GammeRemise()* <> Nothing).

Si c'est la cas, la méthode *WriteDefault()* créé les sous-objets suivants après avoir enregistré l'objet dans la base de données :

Type sous-objet	Description
IBOArticleTarifQteCategorie3	Tarif par quantité ou montant pour une catégorie tarifaire.
IBOArticleTarifQteClient3	Tarif par quantité ou montant pour un client.
IBOArticleTarifQteFournisseur3	Tarif par quantité ou montant pour un fournisseur.

IBIArticleTarifCond3

Tarif article par conditionnement.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Article() As IBOArticle3	Article auquel est associé le tarif par conditionnement.
Lecture / Ecriture	ArticleCond() As IBOArticleCond3	Conditionnement.
Lecture seule	ArticleTarif() As IBIArticleTarif3	Tarif de l'article
Lecture / Ecriture	Prix() As Double	Prix unitaire.

IBIArticleTarifGamme3

Tarif article par énuméré de gamme.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Article() As IBOArticle3	Article auquel est associé le tarif par éuméré de gamme.
Lecture / Ecriture	ArticleGammeEnum1() As IBOArticleGammeEnum3	Enuméré de gamme 1.

Accès	Syntaxe	Description
Lecture / Ecriture	ArticleGammeEnum2() As IBOArticleGammeEnum3	Enuméré de gamme 2.
Lecture seule	ArticleTarif() As IBIArticleTarif3	Tarif article.
Lecture / Ecriture	Prix() As Double	Prix unitaire.

IBIArticleTarifQte3

Tarif de l'article par quantité, montant ou prix net.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Article() As.IBOArticle3	Article auquel est associé le tarif par quantité, montant ou prix net.
Lecture seule	ArticleTarif() As IBIArticleTarif3	Tarif Article.
Lecture / Ecriture	BorneSup() As Double	Borne supérieure.
Lecture / Ecriture	PrixNet() As Double	Prix Net.
Lecture / Ecriture	Remise() As IRemise2	Remise.

IBIArticleTarifVente3

Tarif de vente de l'article.

Interface héritée

Syntaxe	Description
IBIArticleTarif3	Cf. Interface IBIArticleTarif3 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Arrondi() As IBPArrondi	Mode d'arrondi.
Lecture / Ecriture	Calcul() As Boolean	Calcul du prix de vente en fontion du prix de revient (True) ou non (False).

Accès	Syntaxe	Description
Lecture / Ecriture	Coefficient() As Double	Coefficient (Prix d'achat x Coefficient = Prix de vente).
Lecture seule	FactoryArticleTarifCond() As IBITypeObjectFactory	Fabrique un objet tarif par conditionnement.
Lecture / Ecriture	PrixTTC() As Boolean	Prix TTC (True) ou Prix HT (False).

IBICategorieCompta

Catégorie comptable.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Domaine() As DomaineType	Domaine de la catégorie comptable (cf. énumérateur DomaineType).
Lecture / Ecriture	Intitule() As String	Intitulé de la catégorie comptable.

IBIParamCompta3

Paramétrage comptable.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	CategorieCompta() As IBICategorieCompta	Catégorie comptable à laquelle est associé le paramétrage comptable.
Lecture / Ecriture	CompteA() As IBOCompteA3	Compte analytique.
Lecture / Ecriture	CompteG() As IBOCompteG3	Compte général.

Accès	Syntaxe	Description
Lecture / Ecriture	DateApplication(ByVal sElt As Short) As Date	Date d'application de la taxe (sElt de 1 à 3).
Lecture / Ecriture	OldTaxe(ByVal sElt As Short) As IBOTaxe3	Ancien code taxe (sElt de 1 à 3).
Lecture / Ecriture	Taxe(ByVal sElt As Short) As IBOTaxe3	Taxe (sElt de 1 à 3).
Lecture / Ecriture	TaxeDateApplication(ByVal sElt As Short, ByVal cDateApp As Date) As IBOTaxe3	Retourne la taxe à utiliser pour une date donnée (peut retourner nul).

IControleQualiteStat

Statistiques de contrôle qualité.

Propriétés

Accès	Syntaxe	Description
Lecture seule	QteAControler() As Double	Quantité à contrôler.
Lecture seule	QteControlee() As Double	Quantité contrôlée.
Lecture seule	QteRebut() As Double	Quantité mise au rebut.
Lecture seule	QteRetour() As Double	Quantité retournée.
Lecture seule	QteValidee() As Double	Quantité validée.

IDOCLigneValorisation

Valorisation de ligne de document.

Propriétés

Accès	Syntaxe	Description
Lecture seule	HT() As Double	Montant HT.
Lecture seule	HTBrut() As Double	Montant HT Brut.
Lecture seule	HTDev() As Double	Montant HT en devise.
Lecture seule	HTNet() As Double	Montant HT Net.
Lecture seule	HTNetDev() As Double	Montant HT Net en devise.
Lecture seule	Marge() As Double	Montant de la marge.
Lecture seule	PoidsBrut() As Double	Poids brut.
Lecture seule	PoidsNet() As Double	Poids net.

Accès	Syntaxe	Description
Lecture seule	PrixRevient() As Double	Prix de revient.
Lecture seule	PU_Net	Prix unitaire net.
Lecture seule	PU_NetDev	Prix unitaire net en devise.
Lecture seule	Qte() As Double	Quantité.
Lecture seule	QteColisee() As Double	Quantité colisée.
Lecture seule	Taxes() As IDOCValoTaxes	Informations sur les taxes du document (cf.IDOCValoTaxes)
Lecture seule	TotalTaxes() As Double	Montant total des taxes.
Lecture seule	TotalTaxesDev() As Double	Montant total des taxes en devise.
Lecture seule	TTC() As Double	Montant TTC.
Lecture seule	TTC_Brut() As Double	Montant TTC Brut.
Lecture seule	TTC_HorsEscompte() As Double	Montant TTC Hors escompte.
Lecture seule	TTCDev() As Double	Montant TTC en devise.

IDOCValoEscompte

Informations sur l'escompte défini sur un document (accessible uniquement pour les documents de vente et d'achat).

Propriétés

Accès	Syntaxe	Description
Lecture seule	BaseCalcul() As Double	Base de calcul de l'escompte.
Lecture seule	Montant() As Double	Montant de l'escompte.
Lecture seule	Taux() As Double	Taux d'escompte.

IDOCValorisation

Informations sur la valorisation d'un document (accessible uniquement pour les documents de vente et d'achat).

Propriétés

Accès	Syntaxe	Description
Lecture seule	Escompte() As IDOCValoEscompte	Informations sur l'escompte du document (cf. IDOCValoEscompte).
Lecture seule	NetAPayer() As Double	Montant net à payer.

Accès	Syntaxe	Description
Lecture seule	NetAPayerDev() As Double	Montant net à payer en devise.
Lecture seule	Taxes() As IDOCValoTaxes	Informations sur les taxes du document (cf.IDOCValoTaxes).
Lecture seule	TotalAcomptes() As Double	Montant total des acomptes.
Lecture seule	TotalAcomptesDev() As Double	Montant total des acomptes en devise.
Lecture seule	TotalBonsAchat() As Double	Montant total des bons d'achat.
Lecture seule	TotalBonsAchatDev() As Double	Montant total des bons d'achat en devise.
Lecture seule	TotalEcheance() As Double	Montant total des échéances.
Lecture seule	TotalEcheanceDev() As Double	Montant total des échéances en devise.
Lecture seule	TotalHT() As Double	Montant total HT.
Lecture seule	TotalHTBrut() As Double	Montant total HT Brut.
Lecture seule	TotalHTDev() As Double	Montant total HT en devise.
Lecture seule	TotalHTNet() As Double	Montant total HT Net.
Lecture seule	TotalHTNetDev() As Double	Montant total HT Net en devise.
Lecture seule	TotalMarge() As Double	Montant total de la marge.
Lecture seule	TotalPoidsBrut() As Double	Montant total poids brut.
Lecture seule	TotalPoidsNet() As Double	Montant total poids net.
Lecture seule	TotalPrixRevient() As Double	Montant total Prix de revient.
Lecture seule	TotalQte() As Double	Montant total des quantités.
Lecture seule	TotalQteColisee() As Double	Montant total des quantités colisées.
Lecture seule	TotalTaxes() As Double	Montant total des taxes.
Lecture seule	TotalTaxesDev() As Double	Montant total des taxes en devise.
Lecture seule	TotalTTC() As Double	Montant total TTC.
Lecture seule	TotalTTC_Brut() As Double	Montant total TTC Brut.
Lecture seule	TotalTTC_HorsEscompte() As Double	Montant total TTC Hors escompte.
Lecture seule	TotalTTCDev() As Double	Montant total TTC en devise.

IDOCValoTaxe

Elément de taxe d'un document (accessible uniquement pour les documents de vente et d'achat).

Propriétés

Accès	Syntaxe	Description
Lecture seule	BaseCalcul() As Double	Base de calcul de la taxe.
Lecture seule	BaseCalculDev() As Double	Base de calcul de la taxe en devise.

Accès	Syntaxe	Description
Lecture seule	Code() As String	Code taxe.
Lecture seule	Montant() As Double	Montant de la taxe.
Lecture seule	MontantDev() As Double	Montant de la taxe en devise.
Lecture seule	Taux() As Double	Taux de la taxe.

IDOCValoTaxes

Collection de taxes d'un document (accessible uniquement pour les documents de vente et d'achat).

Propriétés

Accès	Syntaxe	Description
Lecture seule	Count() As Long	Nombre d'élément de taxe.
Lecture seule	Item(ByVal iIndex As Long) As IDOCValoTaxe	Elément de taxe.

IFrais2

Frais des articles (Cf. IBOArticle3 et IBOFamille3).

Cf. : *Exemples avancés – Gestion des coûts des articles*.

Propriété

Accès	Syntaxe	Description
Lecture / Ecriture	Frais(ByVal *sElt* As Short) As IFraisElem	Retourne le N ieme (1 à 3) élément de frais.

Méthode

Syntaxe	Description
Calcul(ByVal *dMontantUnitaire* As Double, ByVal dQte as double, ByVal *NbDecimalResult* As Short) As Double	Calcul du montant des frais de l'article en fonction de la valeur *dMontantUnitaire* et de la quantité *dQte* passées en paramètres. Le résultat est arrondi au nombre de décimales *NbDecimalResult*.

IFraisElem2

Elément de frais (Cf. IFrais2).

Cf. : *Exemples avancés – Gestion des coûts des articles*.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	FR_Denomination() As String	Intitulé du frais.
Lecture / Ecriture	Remise() As IRemise	Coût.

Méthode

Syntaxe	Description
Calcul(ByVal *dMontantUnitaire* As Double, ByVal *dQte* As Double, ByVal *NbDecimalResult* As Short) As Double	Calcul du montant des frais en fonction de la valeur *dMontantUnitaire* et de la quantité *dQte* passées en paramètres. Le résultat est arrondi au nombre de décimales *NbDecimalResult*.

IRemise2

Remise (tarifs) / Coût (frais).

- Voir « Exemples avancés – Gestion des coûts des articles ».

Propriété

Accès	Syntaxe	Description
Lecture / Ecriture	Remise(ByVal sElt As Short) As IRemiseElem2	Nieme remise / coût.

Méthodes

Syntaxe	Description
Calcul(ByVal *dVal* As Double, ByVal *dQte* As Double, ByVal *NbDecimalResult* As Short) As Double	Calcul de la remise / du coût en fonction de la valeur *dVal* et de la quantité *dQte* passées en paramètres. Le résultat est arrondi au nombre de décimales *NbDecimalResult*.
CalculDevise(ByVal *dVal* As Double, ByVal *dQte* As Double, ByVal *sDate* As Date, ByVal *pDevise* As IBPDevise2) As Double	Calcul de la remise / du coût en devise en fonction de la valeur *dVal* ,de la quantité *dQte,* de la date *sDate* et devise *pDevise* passées en paramètres. Le résultat est arrondi au nombre de décimales *NbDecimalResult*.

Syntaxe	Description
FromString(ByVal *bVal* As String)	Remise / coût exprimé sous forme d'une chaîne de caractères. Exemple : Article.AR_Frais.Frais(2).Remise.FromString("1 %+5U+100F")
ToString() As String	Retourne une remise / un coût exprimé sous forme d'une chaîne de caractères.

IRemiseElem2

Eléments de remise (tarifs) / de coût (frais).

- Voir « Exemples avancés – Gestion des coûts des articles ».

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	REM_Type() As RemiseType	Type de remise / de coût (Cf. énumérateur RemiseType).
Lecture / Ecriture	REM_Valeur() As Double	Valeur de la remise / du coût.

Méthodes

Syntaxe	Description
Calcul(ByVal *dVal* As Double, ByVal *dQte* As Double, ByVal *NbDecimalResult* As Short) As Double	Calcul de la remise / du coût en fonction de la valeur *dVal* et de la quantité *dQte* passées en paramètres. Le résultat est arrondi au nombre de décimales *NbDecimalResult*.
CalculDevise(ByVal *dVal* As Double, ByVal *dQte* As Double, ByVal *sDate* As Date, ByVal *pDevise* As IBPDevise2) As Double	Calcul de la remise / du coût en devise en fonction de la valeur *dVal*, de la quantité *dQte*, de la date *sDate* et devise *pDevise* passées en paramètres. Le résultat est arrondi au nombre de décimales *NbDecimalResult*.
FromString(ByVal *sVal* As String)	Remise / coût exprimé sous forme d'une chaîne de caractères.
ToString() As String	Retourne une remise / un coût exprimé sous forme d'une chaîne de caractères.

ITarif2

Tarif.

Propriétés

Accès	Syntaxe	Description
Lecture seule	CodeBarre() As String	Code barre.
Lecture seule	Devise() As IDevise	Devise.
Lecture seule	IsHorsRemise() As Boolean	Tarif hors remise (True) ou non (False).
Lecture seule	Prix() As Double	Prix.
Lecture seule	PrixDevise() As Double	Prix en devise.
Lecture seule	Reference() As String	Référence.
Lecture seule	Remise() As IRemise2	Remise.

ITarifAchat2

Tarif d'achat.

Interface héritée

Syntaxe	Description
ITarif2	Cf. Interface ITarif2 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Colisage() As Double	Colisage.
Lecture seule	DelaiAppro() As Short	Délai d'approvisionnement (en jours).
Lecture seule	Garantie() As Short	Garantie (en mois).
Lecture seule	QteMini() As Double	Quantité minimale.
Lecture seule	QteReelle() As Double	Quantité réelle.

ITarifVente2

Tarif de vente.

Interface héritée

Syntaxe	Description
ITarif2	Cf. Interface ITarif2 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Arrondi() As IArrondi	Arrondi.
Lecture seule	Calcul() As Boolean	Calcul du prix de vente en fonction du prix de revient.
Lecture seule	Coefficient() As Double	Coefficient.
Lecture seule	PrixTTC() As Boolean	Prix TTC.

IBIRessource

Ressource de base.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	ArticleDefault() As IBOArticle3	Article par défaut.
Lecture / Ecriture	Depot() As IBODepot3	Dépôt.
Lecture seule	FactoryRessourceArticle() As IBIRessourceArticleFactory	Fabrique un objet ressource article.
Lecture seule	FactoryRessourceBase() As IBIRessourceFactory	Fabrique un objet ressource base.
Lecture seule	FactoryRessourceRessource() As IBIRessourceRessourceFactory	Fabrique un objet centre ou ressource selon le type de l'objet parent.
Lecture / Ecriture	Horaire1(ByVal *iDayOfWeek* As Integer) As IDateTimePeriod	Plage horaire 1 (de 1 à 7, 1 correspondant à Lundi).
Lecture / Ecriture	Horaire2(ByVal *iDayOfWeek* As Integer) As IDateTimePeriod	Plage horaire 2 (de 1 à 7, 1 correspondant à Lundi).
Lecture / Ecriture	RessourceCentrale() As IBIRessource	Ressource centralisation.

Accès	Syntaxe	Description
Lecture / Ecriture	RP_Capacite() As Long	Capacité.
Lecture / Ecriture	RP_Code() As String	Code ressource.
Lecture / Ecriture	RP_Commentaire() As String	Commentaire.
Lecture / Ecriture	RP_Complement() As String	Complément.
Lecture / Ecriture	RP_Continue() As Boolean	Utilisation continue.
Lecture / Ecriture	RP_CoutStd() As Double	Coût standard.
Lecture seule	RP_DateCreation() As Date	Date de création.
Lecture / Ecriture	RP_Intitule() As String	Intitulé.
Lecture / Ecriture	RP_Sommeil() As Boolean	Mise en sommeil.
Lecture / Ecriture	RP_Temps() As Long	Temps d'utilisation.
Lecture / Ecriture	RP_Type() As RessourceProdType	Type de ressource produit (centre ou ressource). Cf. énumérateur RessourceProdType.
Lecture / Ecriture	RP_TypeRess() As RessourceType	Type de ressource (Cf. énumérateur RessourceType).
Lecture / Ecriture	RP_Visite() As Date	Date prochaine visite.
Lecture / Ecriture	Unite() As IBPUnite	Unité.
Lecture / Ecriture	RP_Facebook() As String	Compte Facebook.
Lecture / Ecriture	RP_LinkedIn() As String	Compte LinkedIn.
Lecture / Ecriture	RP_Skype() As String	Compte Skype.

Méthode SetDefault()

Si l'objet est non persistant, l'appel de la méthode *SetDefault()* initialise la propriété suivante :

Propriété	Valeur	Description
RP_DateCreation	Si RP_DateCreation = "" Alors RP_DateCreation = Date du jour	Affecte la valeur du champ date de création si celui-ci est vide.

IBISouche

Souche et numérotation.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Domaine() As DomaineType	Domaine de la souche (Vente, Achat, Stock, Interne).
Lecture / Ecriture	Intitule() As String	Intitulé de la souche.
Lecture / Ecriture	IsValide() As Boolean	Souche valide.

Méthodes

Syntaxe	Description
ReadCurrentPiece(ByVal *DO_Type* As DocumentType, ByVal *DO_Provenance* As DocumentProvenanceType) As String	Retourne le prochain numéro de pièce à utiliser pour le type et la provenance du document passés en paramètres.
WriteCurrentPiece(ByVal *DO_Type* As DocumentType, ByVal *DO_Provenance* As DocumentProvenanceType, ByVal *sVal* As String)	Enregistre le numéro de pièce pour le type et la provenance du document passés en paramètres.

IBIParamDoc

Paramétrage de documents (Paramètres société / Documents).

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	DefaultSouche() As IBISouche	Souche par défaut.
Lecture seule	DO_Provenance() As DocumentProvenanceType	Provenance du document (cf. énumérateur DocumentProvenanceType).
Lecture seule	DO_Type() As DocumentType	Type du document (cf. énumérateur DocumentType).

Accès	Syntaxe	Description
Lecture seule	Domaine() As DomaineType	Domaine du document (Vente, Achat, Stock, Interne). Cf. énumérateur DomaineType.
Lecture / Ecriture	Intitule() As String	Intitulé de la souche.
Lecture / Ecriture	Validation(ByVal sElt As integer) As Boolean	Circuit de validation.
Lecture / Ecriture	ValidationStr(ByVal sElt As Short) As String	Permet de paramétrer les chaînes des statuts. sElt = 1 : Saisi sElt = 2 : Confirmé sElt = 3 : Validé Ne s'applique pas aux documents de stocks.

Méthode

Syntaxe	Description
ReadCurrentPiece(*pSouche* As IBISouche) As String	Retourne le prochain numéro de pièce à utiliser pour la souche passée en paramètre.

IBIReglement

Modèle de règlement.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Condition() As ReglementConditionType	Condition de règlement (cf. énumérateur ReglementConditionType).
Lecture / Ecriture	JourTb(ByVal *sElt* As Integer) As Integer	Jour de tombée.
Lecture / Ecriture	NbJour() As Integer	Nombre de jours.
Lecture / Ecriture	Reglement() As IBPReglement3	Mode de règlement.
Lecture / Ecriture	TRepart() As ReglementRepartitionType	Type de répartition (cf. énumérateur ReglementRepartitionType).
Lecture / Ecriture	VRepart() As Double	Valeur du règlement.

Méthode

Syntaxe	Description
Echeance(ByVal dDate As Date) As Date	Retourne la date d'échéance calculée pour la date passée en paramètre.

IBIFamilleTarif

Tarifs famille.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Devise() As IBPDevise2	Devise du tarif.
Lecture seule	FactoryFamilleTarifQte() As IBITypeObjectFactory	Fabrique un objet famille tarif par quantité.
Lecture / Ecriture	Famille() As IBOFamille3	Famille.
Lecture / Ecriture	GammeRemise() As IBPGamme	Gamme de remise.
Lecture / Ecriture	IsHorsRemise() As Boolean	Tarif hors remise.
Lecture / Ecriture	Remise() As IRemise2	Elément remise.

IBIFamilleTarifVente

Tarifs de vente par famille.

Interface héritée

Syntaxe	Description
IBIFamilleTarif	Cf. Interface IBIFamilleTarif pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Arrondi() As IBPArrondi	Elément d'arrondi.
Lecture / Ecriture	Calcul() As Boolean	Calcul du prix en fonction du prix de revient (True) ou non (False).
Lecture / Ecriture	Coefficient() As Double	Coefficient.
Lecture / Ecriture	PrixTTC() As Boolean	Prix TTC (True) ou HT (False).

IBIFamilleTarifQte

Tarifs famille par quantité.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	BorneSup() As Double	Borne supérieure.
Lecture seule	Famille() As IBOFamille3	Famille.
Lecture seule	FamilleTarif() As IBIFamille	Tarif famille.
Lecture / Ecriture	Remise() As IRemise2	Elément remise.

IBILot

Lots.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	ArticleDepot() As IBIArticleStock3	Dépôt de stockage article.
Lecture seule	IsEntamed() As Boolean	Lot entamé.
Lecture seule	IsEpuised() As Boolean	Lot épuisé.
Lecture seule	LigneEntree() As IBODocumentLigne3	Ligne d'entrée.

Méthodes

Syntaxe	Description
StockATerme() As Double	Stock à terme.
StockReel() As Double	Stock réel.

IUserColis

Elément colis d'une ligne de préparation de livraison ajoutée dans le processus de colisage.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Qte() As Double	Quantité du colis.
Lecture / Ecriture	NumColis() As String	Numéro du colis.

Méthode

Syntaxe	Description
Set(ByVal sNumColis As String, ByVal dQteColis As Double)	Ajoute un colis en une seule fois.

IUserLot

Elément série/lot d'une ligne de document ajoutée dans un processus de transformation de documents, ou dans un processus de prélèvement des série/lot.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Lot() As IBOArticleDepotLot	Objet article dépôt lot.
Lecture / Ecriture	LS_ComplementOut() As String	Complément du lot.
Lecture / Ecriture	QteToUse() As Double	Quantité affectée au lot.

Méthode

Syntaxe	Description
Set(*pLot* As IBOArticleDepotLot, ByVal *dQteToUse* As Double, ByVal *sLS_Complement* As String)	Ajoute un lot en une seule fois.

IUserEmplacement

Elément emplacement d'une ligne de document ajoutée au processus de création de mouvement de transfert.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Emplacement() As IBODepotEmplacement	Objet emplacement.
Lecture / Ecriture	QteToUse() As Double	Quantité affectée à l'emplacement.

Méthode

Syntaxe	Description
Set(*pEmpl* As IBODepotEmplacement, ByVal *dQteToUse* As Double)	Ajoute l'emplacement en une seule fois.

ITransformation

Interface permettant l'accès aux processus de transformation d'achat ou vente.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Achat() As ITransformationAchat	Accès au processus de transformation des documents/lignes d'achat.
Lecture seule	Vente() As ITransformationVente	Accès au processus de transformation des documents/lignes de vente.

ITransformationAchat

Interface permettant l'accès aux processus de transformation des documents/lignes d'achat.

Méthodes

Syntaxe	Description
CreateProcess_Commander() As IPMDocTransformer	Création d'un processus de transformation de documents/lignes d'achat vers un bon de commande d'achat.
CreateProcess_Receptionner() As IPMDocTransformer	Création d'un processus de transformation de documents/lignes d'achat vers un bon de livraison d'achat.
CreateProcess_Facturer() As IPMDocTransformer	Création d'un processus de transformation de documents/lignes d'achat vers une facture d'achat.

ITransformationVente

Interface permettant l'accès aux processus de transformation des documents/lignes de vente.

Méthodes

Syntaxe	Description
CreateProcess_Commander() As IPMDocTransformer	Création d'un processus de transformation de documents/lignes de vente vers un bon de commande de vente.
CreateProcess_Livrer() As IPMDocTransformer	Création d'un processus de transformation de documents/lignes de vente vers un bon de livraison de vente.
CreateProcess_Facturer() As IPMDocTransformer	Création d'un processus de transformation de documents/lignes de vente vers une facture de vente.
CreateProcess_Preparer() As IPMDocTransformer	Création d'un processus de transformation de documents/lignes de vente vers une préparation de livraison de vente.

Interfaces factory paramètres (IBIxxxFactory, IBPxxxFactory)

IBPAgendaFactory

Fabrique un objet agenda.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *sIntitule* As String) As Boolean	Teste l'existence de l'objet agenda correspondant à l'intitulé passé en paramètre. Retourne *True* s'il existe, sinon *False*.
ReadIntitule(ByVal *sIntitule* As String) As IBPArticleStat	Retourne l'objet agenda correspondant à l'intitulé passé en paramètre.

IBPArticleStatFactory

Fabrique un objet statistique article.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *sIntitule* As String) As Boolean	Teste l'existence de l'objet statistique article correspondant à l'intitulé passé en paramètre. Retourne *True* s'il existe, sinon *False*.
ReadIntitule(ByVal *sIntitule* As String) As IBPArticleStat	Retourne l'objet statistique article correspondant à l'intitulé passé en paramètre.

IBPCategorieComptaAchatFactory

Fabrique un objet Catégorie comptable d'achat.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *sIntitule* As String) As Boolean	Test l'existence de l'objet Catégorie comptable achat correspondant à l'intitulé passé en paramètre. Retourne *True* s'il existe, sinon *False*.
ReadIntitule(ByVal *sIntitule* As String) As IBPCategorieComptaAchat	Retourne l'objet Catégorie comptable achat correspondant à l'intitulé passé en paramètre.

IBPCategorieComptaStockFactory

Fabrique un objet Catégorie comptable de stock.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *sIntitule* As String) As Boolean	Test l'existence de l'objet Catégorie comptable stock correspondant à l'intitulé passé en paramètre. Retourne *True* s'il existe, sinon *False*.
ReadIntitule(ByVal *sIntitule* As String) As IBPCategorieComptaStock	Retourne l'objet Catégorie comptable stock correspondant à l'intitulé passé en paramètre.

IBPCategorieComptaVenteFactory

Fabrique un objet Catégorie comptable de vente.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *sIntitule* As String) As Boolean	Test l'existence de l'objet Catégorie comptable vente correspondant à l'intitulé passé en paramètre. Retourne *True* s'il existe, sinon *False*.
ReadIntitule(ByVal *sIntitule* As String) As IBPCategorieComptaVente	Retourne l'objet Catégorie comptable vente correspondant à l'intitulé passé en paramètre.

IBPCategorieTarifFactory

Fabrique un objet Catégorie tarifaire.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *sIntitule* As String) As Boolean	Test l'existence de l'objet Catégorie tarifaire correspondant à l'intitulé passé en paramètre. Retourne True s'il existe, sinon False.

Syntaxe	Description
ReadIntitule(ByVal *sIntitule* As String) As IBPCategorieTarif	Retourne l'objet Catégorie tarifaire correspondant à l'intitulé passé en paramètre.

IBPConditionLivraisonFactory

Fabrique un objet Condition de livraison.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *sIntitule* As String) As Boolean	Test l'existence de l'objet Condition de livraison correspondant à l'intitulé passé en paramètre. Retourne *True* s'il existe, sion *False*.
ReadIntitule(ByVal *sIntitule* As String) As IBPConditionLivraison	Retourne l'objet Condition de livraison correspondant à l'intitulé passé en paramètre.

IBPConditionnementFactory

Fabrique un objet Conditionnement.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *sIntitule* As String) As Boolean	Test l'existence de l'objet Conditionnement correspondant à l'intitulé passé en paramètre. Retourne *True* s'il existe, sinon *False*.
Function ReadIntitule(ByVal *sIntitule* As String) As IBPConditionnement	Retourne l'objet Conditionnement correspondant à l'intitulé passé en paramètre.

IBPExpeditionFactory3

Fabrique un objet Mode d'expédition.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *sIntitule* As String) As Boolean	Test l'existence de l'objet Mode d'expédition correspondant à l'intitulé passé en paramètre.
ReadIntitule(ByVal *sIntitule* As String) As IBPExpedition3	Retourne l'objet Mode d'expédition correspondant à l'intitulé passé en paramètre.

IBPGammeFactory

Fabrique un objet Gamme.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *sIntitule* As String) As Boolean	Test l'existence de l'objet Gamme correspondant à l'intitulé passé en paramètre.
ReadIntitule(ByVal *sIntitule* As String) As IBPGamme	Retourne l'objet Gamme correspondant à l'intitulé passé en paramètre.

IBPPeriodiciteFactory

Fabrique un objet Périodicité.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *sIntitule* As String) As Boolean	Test l'existence de l'objet Périodicité correspondant à l'intitulé passé en paramètre.
ReadIntitule(ByVal *sIntitule* As String) As IBPPeriodicite	Retourne l'objet Périodicité correspondant à l'intitulé passé en paramètre.

IBPProduitFactory2

Fabrique un objet Catalogue article.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
Function ExistIntitule(ByVal *sIntitule* As String) As Boolean	Test l'existence de l'objet Catégorie de produit correspondant à l'intitulé passé en paramètre.
ReadIntitule(ByVal *sIntitule* As String) As IBPProduit2	Retourne l'objet catalogue article correspondant à l'intitulé passé en paramètre.

IBPUniteFactory

Fabrique un objet Unité d'achat et de vente.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *sIntitule* As String) As Boolean	Test l'existence de l'objet Unité d'achat et de vente correspondant à l'intitulé passé en paramètre.
ReadIntitule(ByVal *sIntitule* As String) As IBPUnite	Retourne l'objet Unité d'achat et de vente correspondant à l'intitulé passé en paramètre.

IBPMotifDevisFactory

Fabrique un objet Motif devis perdu.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *sIntitule* As String) As Boolean	Test l'existence de l'objet Motif devis perdu correspondant à l'intitulé passé en paramètre.
ReadIntitule(ByVal *sIntitule* As String) As IBPMotifDevis	Retourne l'objet motif devis perdu correspondant à l'intitulé passé en paramètre.

IBIArticleStockEmplFactory

Fabrique un objet emplacement stock article.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistEmplacement(*pDepotEmpl* As IBODepotEmplacement) As Boolean	Teste l'existence de l'objet emplacement dépôt pour l'emplacement passé en paramètre. Retourne *True* s'il existe sinon *False*.
ReadEmplacement(*pDepotEmpl* As IBODepotEmplacement) As IBIArticleStockEmpl	Retourne l'objet emplacement dépôt correspondant à l'emplacement passé en paramètre.

IBIParamDocFactory

Fabrique un objet paramétrage de document (organisation des documents).

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *sIntitule* As String) As Boolean	Test l'existence de l'objet paramétrage de document correspondant à l'intitulé passé en paramètre. Retourne *True* s'il existe sinon *False*.
ReadIntitule(ByVal *sIntitule* As String) As IBIParamDoc	Retourne l'objet paramétrage de document correspondant à l'intitulé passé en paramètre.
ReadTypeDocument(ByVal *DO_Type* As DocumentType, [ByVal *DO_Provenance* As DocumentProvenanceType]) As IBIParamDoc	Retourne le paramétrage de document pour le type et la provenance (paramètre optionnel) passés en paramètres.

IBISoucheFactory

Fabrique un objet souche de document.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *sIntitule* As String) As Boolean	Test l'existence de l'objet souche de document correspondant à l'intitulé passé en paramètre. Retourne *True* s'il existe sinon *False*.
QuerySoucheValide() As IBICollection	Retourne une collection de souches de document valides.
ReadIntitule(ByVal *sIntitule* As String) As IBISouche	Retourne l'objet souche de document correspondant à l'intitulé passé en paramètre.

IBIRessourceFactory

Fabrique un objet ressource de base.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	InfoLibreFields() As IBIFields	Informations libres.
Lecture seule	ListOrderDepot() As IBICollection	Retourne la liste des ressources triées par dépôt.
Lecture seule	ListOrderIntitule() As IBICollection	Retourne la liste des ressources triées par intitulé.
Lecture seule	ListOrderTypeRessource() As IBICollection	Retourne la liste des ressources triées par type de ressource.

Méthodes

Syntaxe	Description
QueryTypeOrderCode(ByVal *sType* As RessourceType) As IBICollection	Retourne une collection de ressources triées par code pour le type de ressource passé en paramètre.
QueryTypeOrderDepot(ByVal *sType* As RessourceType) As IBICollection	Retourne une collection de ressources triées par dépôt pour le type de ressource passé en paramètre.
QueryTypeOrderIntitule(ByVal *sType* As RessourceType) As IBICollection	Retourne une collection de ressources triées par intitulé pour le type de ressource passé en paramètre.
QueryTypeOrderTypeRessource(ByVal sType As Ressourcetype) As IBICollection	Retourne une collection de ressources triées par type de ressource produit pour le type de ressource passé en paramètre.

IBIRessourceArticleFactory

Fabrique un objet ressource article de base.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
AddArticle(*pArticle* As IBOArticle3)	Ajoute à la ressource l'article passé en paramètre.
RmvArticle(*pArticle* As IBOArticle3)	Supprime de la ressource l'article passé en paramètre.

IBIRessourceRessourceFactory

Fabrique un objet ressource.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
AddRessource(*pRessource* AS IBIRessource)	Ajoute la ressource passée en paramètre.
RmvRessource(*pRessource* As IBIRessource)	Supprime la ressource passée en paramètre.

Interfaces factory métiers (IBOxxxFactory)

IBOAgendaFactory

Fabrique un objet événement agenda.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
QueryNonEchu() As IBICollection	Retourne la collection des événements agendas non échus.
QueryNonEchuType(ByVal *stype* As AgendaTypeInteresse) As IBICollection	Retourne la collection des événements agendas non échus pour le type passé en paramètre.
QueryType(ByVal *sType* As AgendaTypeInteresse) As IBICollection	Retourne la collection des événements agendas pour le type passé en paramètre.

IBOArticleCondFactory

Fabrique un objet conditionnement article.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistEnumere(ByVal *sEnum* As String, ByVal *dQte* As Double) As Boolean	Test l'existence d'un énuméré de conditionnement correspondant à l'énuméré et la quantité passés en paramètres. Retourne *True* si l'article existe, sinon *False*.
ReadEnumere(ByVal *sEnum* As String, ByVal *dQte* As Double) As IBOArticleCond3	Retourne l'énuméré de conditionnement correspondant à l'énuméré et la quantité passés en paramètres.
ExistReference(ByVal *sReference* As String) As Boolean	Test l'existence de l'énuméré de conditionnement correspondant à la référence de conditionnement passée en paramètre. Retourne *True* si l'article existe, sinon *False*.
ReadReference(ByVal *sReference* As String) As IBOArticleCond3	Retourne l'énuméré de conditionnement correspondant à la référence de conditionnement passée en paramètre.
ExistPrincipal() As Boolean	Teste l'existence d'un énuméré de conditionnement principal .
ReadPrincipal() As IBOArticleCond3	Retourne l'énuméré de conditionnement principal .

IBOArticleFactory3

Fabrique un objet article ou une collection d'objets articles.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	InfoLibreFields() As IBIFields	Collection de champs (informations libres).
Lecture seule	ListOrderDesignation() As IBICollection	Retourne une collection d'articles classée par désignations.
Lecture seule	ListOrderFamille() As IBICollection	Retourne une collection d'articles classée par familles.
Lecture seule	ListOrderReference() As IBICollection	Retourne une collection d'articles classée par référence articles.

Méthodes

Syntaxe	Description
ExistCodeBarre(ByVal *sCodeBarre* As String) As Boolean	Test l'existence d'un article correspondant au code barre passé en paramètre. Retourne *True* si l'article existe, sinon *False*.
ExistReference(ByVal *sReference* As String) As Boolean	Test l'existence de l'article correspondant à la référence article passée en paramètre. Retourne *True* si l'article existe, sinon *False*.
QueryActifOrderReference() As IBICollection	Retourne une collection d'articles actifs (pas en sommeil) classée par référence article.
QueryBaremeSoldeApplicable(ByVal *pSolde* AsIBOBaremeSolde) As IBICollection	Retourne une collection d'articles sur lesquels le barême/solde passé en paramètre s'applique.
QueryCatalogue(*pProduit* As IBPProduit2) As IBICollection	Retourne une collection d'articles appartenant au catalogue article passé en paramètre.
QueryDepotEmplacement(ByVal *pEmpl* As IBODepotEmplacement) As IBICollection	Retourne une collection d'articles passés au moins une fois par l'emplacement passé en paramètre.
QueryFamille(*pFamille* As IBOFamille3) As IBICollection	Retourne une collection d'articles appartenant à la famille passée en paramètre, classée par famille.
QueryMouvementeOrderReference() As IBICollection	Retourne une collection d'articles mouvementés classée par référence article.
QueryPublieOrderReference() As IBICollection	Retourne une collection d'articles publiés sur le site marchand classée par référence.
QueryReferenceOrderReference(ByVal *AR_RefDe* As String, ByVal *AR_RefA* As String) As IBICollection	Retourne une collection d'articles compris entre les 2 références articles (*AR_RefDe* et *AR_RefA*) passées en paramètres.
QueryReplicate(ByVal *lIndice* As Integer) As IBICollection	Retourne une collection d'articles dont le nombre de réplications correspond à *lIndice* passé en paramètre.
ReadCodeBarre(ByVal *sCodeBarre* As String) As IBOArticle3	Retourne un objet article correspondant au code barre passé en paramètre.
ReadReference(ByVal *sReference* As String) As IBOArticle3	Retourne un objet article correspondant à la référence article passée en paramètre.

IBOArticleDepotFactory

Fabrique un objet dépôt article.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistDepot(*pDepot* As IBODepot3) As Boolean	Test l'existence de stocks sur l'article pour le dépôt passé en paramètre.
ReadDepot(*pDepot* As IBODepot3) As IBOArticleDepot	Retourne un objet dépôt article pour le dépôt passé en paramètre.

IBOArticleDepotGammeFactory

Fabrique un objet dépôt de stockage pour les gammes article.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistMonoGamme(*pEnum* As IBOArticleGammeEnum3) As Boolean	Test l'existence d'un dépôt de stockage pour l'énuméré de gamme passé en paramètre.
ReadMonoGamme(*pEnum* As IBOArticleGammeEnum3) As IBOArticleDepotGamme3	Retourne l'objet dépôt de stockage pour l'énuméré de gamme article passé en paramètre.
ExistDoubleGamme(*pEnum1* As IBOArticleGammeEnum3, *pEnum2* As IBOArticleGammeEnum3) As Boolean	Test l'existence d'un dépôt de stockage pour les énumérés de gamme passé en paramètre.
ReadDoubleGamme(*pEnum1* As IBOArticleGammeEnum3, *pEnum2* As IBOArticleGammeEnum3) As IBOArticleDepotGamme3	Retourne l'objet dépôt de stockage pour les énumérés des gamme passés en paramètres.

IBOArticleDepotLotFactory

Fabrique un objet lot article.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	InfoLibreFields() As IBIFields	Collection de champs (informations libres).
Lecture seule	NextNoSerie(ByVal *sNumSerie* As String) As String	Calcul d'un numéro de série/lot en retournant la chaîne passée en paramètre incrémentée de 1. La méthode de calcul employée est la même que celle utilisée par la gestion commerciale.

Méthodes

Syntaxe	Description
ExistNoSerie(ByVal *sNoSerie* As String) As Boolean	Teste l'existence de l'objet lot correspondant au numéro série/lot passé en paramètre. Retourne *True* si l'article existe, sinon *False*.
QueryNonEpuise() As IBICollection	Retourne une collection d'objets série/lot non épuisés.
ReadNoSerie(ByVal *sNoSerie* As String) As IBOArticleDepotLot	Retourne l'objet lot pour le numéro série/lot passé en paramètre.
ExistNoSerieEmpl(ByVal *sNoSerie* As String, ByRef *pEmpl* As IBODepotEmplacement) As Boolean	Teste l'existence de l'objet lot correspondant au niméro série/lot et emplacement passes en paramètres.
ReadNoSerieEmpl(ByVal sNoSerie As String, ByRef pEmpl As IBODepotEmplacement) As IBOArticleDepotLot	Retourne l'objet lot pour le numéro série/lot et l'emplacement passés en paramètres.
QueryNoSerie(ByVal *sNoSerie* As String) As IBICollection	Retourne la collection des numéros série/lot pour le numéro passé en paramètre.
QueryNonEpuiseNoSerie(ByVal sNoSerie As String) As IBICollection	Retourne la collection des numéros série/lot non épuisés pour le numéro passé en paramètre.

IBOArticleDepotLotAllFactory

Permet d'accéder à une collection de lots pour tous articles et dépôts confondus.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	InfoLibreFields() As IBIFields	Collection de champs (informations libres).

IBOArticleGammeEnumFactory

Fabrique un objet énuméré de gamme.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistEnumere(ByVal *sEnum* As String) As Boolean	Teste l'existence de l'énuméré de gamme passé en paramètre. Retourne *True* si l'article existe, sinon *False*.
ReadEnumere(ByVal *sEnum* As String) As IBOArticleGammeEnum3	Retourne un objet énuméré de gamme article.
ExistCompteur (int AG_No) As boolean	Teste l'existence d'un énuméré de gamme par son compteur
ReadCompteur (int AG_No) As IBOArticleGammeEnum3*	Lecture d'un énuméré de gamme par son compteur

IBOArticleGlossaireFactory2

Fabrique un objet glossaire article.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthode

Syntaxe	Description
AddGlossaire(ByVal *pGlossaire* As IBOGlossaire2)	Ajoute à l'article l'objet Glossaire passé en paramètre.
RemoveGlossaire(ByVal *pGlossaire* As IBOGlossaire2)	Supprime le lien Article Glossaire

IBOArticleParamComptaFactory3

Fabrique un paramétrage comptable article ou une collection de paramétrages comptables article.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistCategorieCompta(*pCatCompta* As IBICategorieCompta) As Boolean	Test l'existence du paramétrage d'une catégorie comptable. Retourne *True* si la catégorie comptable existe, sinon *False*.
QueryAchat() As IBICollection	Retourne une collection de paramétrages comptables de l'article appartenant au domaine Achats.
QueryStock() As IBICollection	Retourne une collection de paramétrages comptables de l'article appartenant au domaine Stocks.
QueryVente() As IBICollection	Retourne une collection de paramétrages comptables de l'article appartenant au domaine Ventes.
ReadCategorieCompta(*pCatCompta* As IBICategorieCompta) As IBOArticleParamCompta3	Retourne le paramétrage article pour la catégorie comptable passée en paramètre.

IBOArticleRessourceFactory

Fabrique une ressource article.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
AddRessource(*pRessource* As IBIRessource)	Ajoute à l'article la ressource passée en paramètre.
RmvRessource(*pRessource* As IBIRessource)	Supprime de l'article la ressource passée en paramètre.

IBOBaremeCommissionFactory

Fabrique un objet commission.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *sIntitule* As String) As Boolean	Teste l'existence d'un objet commission pour l'intitulé passé en paramètre. Retourne *True* s'il existe, sinon *False*.
QueryApplicableArticleDate(ByVal *pArticle* As IBOArticle3, ByVal *dVal* As Date) As IBICollection	Retourne une collection de commissions applicables pour l'article et la date passés en paramètres.
QueryApplicableDate(ByVal *dDate* As Date) As IBICollection	Retourne une collection de commissions applicables pour la date passée en paramètre.
ReadIntitule(ByVal *sintitule* As String) As IBOBaremeCommission	Retourne l'objet commission correspondant à l'intitulé passé en paramètre.

IBOBaremeRabaisFactory

Fabrique un objet rabais, remises et ristournes.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *sIntitule* As String) As Boolean	Teste l'existence d'un objet rabais, remises et ristournes pour l'intitulé passé en paramètre. Retourne *True* s'il existe, sinon *False*.
QueryApplicableArticleDate(ByVal *pArticle* As IBOArticle3, ByVal *dVal* As Date) As IBICollection	Retourne une collection de rabais, remises et ristournes applicables pour l'article et la date passés en paramètres.
QueryApplicableDate(ByVal *dDate* As Date) As IBICollection	Retourne une collection de rabais, remises et ristournes applicables pour la date passée en paramètre.
ReadIntitule(ByVal *sintitule* As String) As IBOBaremeRabais	Retourne l'objet rabais, remises et ristournes correspondant à l'intitulé passé en paramètre.

IBOBaremeSoldeFactory

Fabrique un objet soldes et promotions.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *sIntitule* As String) As Boolean	Teste l'existence d'un objet soldes et promotions pour l'intitulé passé en paramètre. Retourne *True* s'il existe, sinon *False*.
QueryApplicableArticleDate(ByVal *pArticle* As IBOArticle3, ByVal *dDate* As Date) As IBICollection	Retourne une collection de soldes et promotions applicables pour l'article et la date passés en paramètres.

Syntaxe	Description
QueryApplicableDate(ByVal *dDate* As Date) As IBICollection	Retourne une collection de soldes et promotions applicables pour la date passée en paramètre.
ReadIntitule(ByVal *sintitule* As String) As IBOBaremeSolde	Retourne l'objet soldes et promotions correspondant à l'intitulé passé en paramètre.

IBOClientTarifFamilleFactory

Fabrique un objet tarif famille client.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistFamille(*pFamille* As IBOFamille3) As Boolean	Teste l'existence d'un objet tarif famille client pour la famille passée en paramètre. Retourne *True* s'il existe, sinon *False*.
ReadFamille(*pFamille* As IBOFamille3) As IBOFamilleTarifClient	Retourne l'objet tarif famille client correspondant à la famille passée en paramètre.

IBODepotFactory2

Fabrique un objet dépôt de stockage.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *sNum* As String) As Boolean	Test l'existence d'un objet dépôt correspondant à l'intitulé passé en paramètre. Retourne *True* s'il existe, sinon *False*.
ReadIntitule(ByVal *sNum* As String) As IBODepot3	Retourne l'objet dépôt correspondant à l'intitulé passé en paramètre.

Syntaxe	Description
ExistCompteur (int DE_No) As boolean	Teste l'existence d'un dépôt par son compteur
ReadCompteur (int DE_No) As IBODepot3*	Lecture d'un dépôt par son compteur

IBODepotEmplacementFactory

Fabrique un objet dépôt par emplacement.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistCode(ByVal *sDP_Code* As String) As Boolean	Teste l'existence d'un objet emplacement correspondant au code passé en paramètre. Retourne *True* s'il existe, sinon *False*.
ReadCode(ByVal *sDP_Code* As String) As IBODepotEmplacement	Retourne l'objet emplacement correspondant au code passé en paramètre.

IBODocumentFactory

Fabrique un objet document.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Propriété

Accès	Syntaxe	Description
Lecture seule	InfoLibreFields() As IBIFields	Collection de champs (Informations libres).

IBODocumentAchatFactory3

Fabrique un document d'achat.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
CreateFacture(ByVal *DO_Provenance* As DocumentProvenanceType) As IBODocumentAchat3	Crée une facture d'achat en fonction de la provenance passée en paramètre (cf. énumérateur DocumentProvenanceType).
CreateType(ByVal *DO_Type* As DocumentType) As IBODocumentAchat3	Cree un document d'achat correspondant au type passé au paramètre.
ExistPiece(ByVal *DO_Type* As DocumentType, ByVal *sDO_Piece* As String) As Boolean	Si la pièce (dont le type et le numéro sont passés en paramètres) existe : *True* sinon : *False*.
QueryCollaborateurType(*Collab* As IBOCollaborateur, ByVal *DO_Type* As DocumentType) As IBICollection	Retourne une collection de documents du collaborateur et du type passés en paramètres.
QueryTiersOrderDate(*Fourniss* As IBOFournisseur3) As IBICollection	Retourne une collection de documents d'achat triés par date pour le fournisseur passé en paramètre.
QueryTiersTypeOrderDate(*Fourniss* As IBOFournisseur3, ByVal *DO_Type* As DocumentType) As IBICollection	Retourne une collection de documents d'achats triés par date pour le fournisseur et le type passés en paramètres.
QueryTypeDateOrderPiece(ByVal *DO_Type* As DocumentType, ByVal *DateDeb* As Date, ByVal *DateFin* As Date) As IBICollection	Retourne une collection de documents d'achats triée par numéro de pièce, pour le type de document et la période passés en paramètres.
QueryTypeOrderPiece(ByVal *DO_Type* As DocumentType) As IBICollection	Retourne une collection de documents d'achat triés par numéro de pièce en fonction d'un type de document passé en paramètre.
QueryTypePieceOrderPiece(ByVal *DO_Type* As DocumentType, ByVal *sDO_PieceDe* As String, ByVal *sDO_PieceA* As String) As IBICollection	Retourne une collection de documents d'achat triés par numéro de pièce en fonction d'un type de document et d'une plage de numéros de pièces passés en paramètres.
ReadPiece(ByVal *DO_Type* As DocumentType, ByVal *sDO_Piece* As String) As IBODocumentAchat3	Retourne le document d'achat correspondant au type et au numéro de pièce passés en paramètre.

IBODocumentInterneFactory3

Fabrique un objet Document interne.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
CreateType(ByVal *DO_Type* As DocumentType) As IBODocumentInterne3	Créé un document interne correspondant au type passé en paramètre.
ExistPiece(ByVal *DO_Type* As DocumentType, ByVal *sDO_Piece* As String) As Boolean	Si le document interne existe (type et numéro de pièce passés en paramètres) retourne *True* sinon retourne *False*.
QueryCollaborateurType(*Collab* As IBOCollaborateur, ByVal *DO_Type* As DocumentType) As IBICollection	Retourne une collection de documents du collaborateur et du type passés en paramètres.
QueryTiersOrderDate(*Client* As IBOClient3) As IBICollection	Retourne une collection de documents internes triés par date en fonction d'un client passé en paramètre.
QueryTiersTypeOrderDate(ByVal *Client* As IBOClient3, ByVal *DO_Type* As DocumentType) As IBICollection	Retourne une collection de documents internes triés par date correspondant au client et au type passés en paramètres.
QueryTypeDateOrderPiece(ByVal *DO_Type* As DocumentType, ByVal *DateDeb* As Date, ByVal *DateFin* As Date) As IBICollection	Retourne une collection de documents internes triée par numéro de pièce, pour le type de document et la période passés en paramètres.
QueryTypeOrderPiece(ByVal *DO_Type* As DocumentType) As IBICollection	Retourne une collection de documents internes correspondant au type passé en paramètre.
QueryTypePieceOrderPiece(ByVal *DO_Type* As DocumentType, ByVal *sDO_PieceDe* As String, ByVal *sDO_PieceA* As String) As IBICollection	Retourne une collection de documents internes (type et plage de numéros de pièce passés en paramètre).
ReadPiece(ByVal *DO_Type* As DocumentType, ByVal *sDO_Piece* As String) As IBODocumentInterne3	Retourne le document interne correspondant au type et au numéro de pièce passés en paramètre.

IBODocumentLigneFactory

Fabrique un objet Ligne de document.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Propriété

Accès	Syntaxe	Description
Lecture seule	InfoLibreFields() As IBIFields	Collection de champs (Informations libres).

Méthode

Syntaxe	Description
QueryPied() As IBICollection	Retourne la collection des lignes de pied du document.

IBODocumentLigneAllFactory

Fabrique un objet Ligne de document sans passer par l'entête de document.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	InfoLibreFields() As IBIFields	Collection des champs Informations libres.
Lecture / Ecriture	AutoSet_FraisLigne() As Boolean	Permet de conditionner l'affectation des frais sur les lignes de documents. Valeur par défaut à False. Si AutoSet_FraisLigne = True Alors l'affectation de la propriété DL_Frais entraîne l'affectation automatique des propriétés DL_PrixRU et DL_CMUP.
Lecture / Ecriture	AutoSet_PrixLigne() As Boolean	Permet de gérer le comportement d'affectation des propriétés DL_Prixunitaire, DL_PUDevise et DL_PUTTC. Valeur par défaut à False. Si AutoSet_PrixLigne = True Alors l'afffectation d'une des propriétés DL_Prixunitaire, DL_PUDevise ou DL_PUTTC entraîne le recalcul des deux autres. Si AutoSet_PrixLigne = False Alors l'afffectation d'une des propriétés DL_Prixunitaire, DL_PUDevise ou DL_PUTTC ne recalcule pas les deux autres.

Méthodes

Syntaxe	Description
QueryArticle(*pArticle* As IBOArticle3, ByVal *dValDe* As Date, ByVal *dValA* As Date, ByVal *TypeDe* As DocumentType, ByVal *TypeA* As DocumentType) As IBICollection	Retourne la collection des lignes de document pour l'article, la période et les types de documents passés en paramètres.
QueryArticleDoubleGamme(*pArtGamEnum1* As IBOArticleGammeEnum3, *pArtGamEnum2* As IBOArticleGammeEnum3, ByVal *dValA* As Date, ByVal *TypeDe* As DocumentType, ByVal *TypeA* As DocumentType) As IBICollection	Retourne la collection des lignes de document pour les énumérés de gamme, la période et les types de documents passés en paramètres.
QueryArticleMonoGamme(*pArtGamEnum* As IBOArticleGammeEnum3, ByVal *dValA* As Date, ByVal *TypeDe* As DocumentType, ByVal *TypeA* As DocumentType) As IBICollection	Retourne la collection des lignes de document pour l'énuméré de gamme, la période et les types de documents passés en paramètres.
QueryDepotLot(*pLot* As IBOArticleDepotLot) As IBICollection	Retourne la collection des lignes de document pour le lot article passé en paramètre.
QueryMouvementStockArticle(*pArticle* As IBOArticle3, *pDepot* As IBODepot3, ByVal *dDateDe* As Date, ByVal *dDateA* As Date) As IBICollection	Retourne la collection des lignes de document mouvementant le stock pour l'article, le dépôt et la période passés en paramètres.
QueryMouvementStockArticleDoubleGamme(*pArtGamEnum1* As IBOArticleGammeEnum3, *pArtGamEnum2* As IBOArticleGammeEnum3, *pDepot* As IBODepot3, ByVal *dDateDe* As Date, ByVal *dDateA* As Date) As IBICollection	Retourne la collection des lignes de document mouvementant le stock pour les énumérés de gamme, le dépôt et la période passés en paramètres.
QueryMouvementStockArticleMonoGamme(*pArtGamEnum* As IBOArticleGammeEnum3, *pDepot* As IBODepot3, ByVal *dDateDe* As Date, ByVal *dDateA* As Date) As IBICollection	Retourne la collection des lignes de document mouvementant le stock pour l'énuméré de gamme, le dépôt et la période passés en paramètres.
QueryMouvementStockPreviArticle(*pArticle* As IBOArticle3, *pDepot* As IBODepot3, ByVal *dDateDe* As Date, ByVal *dDateA* As Date, ByVal *bLigneWithoutDateLivraison* As Boolean, ByVal *bIncludePL* As Boolean) As IBICollection	Retourne la collection des lignes de document qui mouvementeront le stock pour l'article, le dépôt et la période passés en paramètres. Les critères permettent de tenir compte des lignes sans date de livraison (*bLigneWithoutDateLivraison* à Vrai) et des préparations de livraisons (*bIncludePL* à Vrai). Les données retournées sont équivalentes à la fonction d'interrogation de stock prévisionnel de Sage 100c Gestion commerciale.

Syntaxe	Description
QueryMouvementStockPreviArticleDoubleGamme(*pArtGamEnum1* As IBOArticleGammeEnum3, *pArtGamEnum2* As IBOArticleGammeEnum3, *pDepot* As IBODepot3, ByVal *dDateDe* As Date, ByVal *dDateA* As Date, ByVal *bLigneWithoutDateLivraison* As Boolean, ByVal *bIncludePL* As Boolean) As IBICollection	Retourne la collection des lignes de document qui mouvementeront le stock pour les énumérés de gamme, le dépôt et la période passés en paramètres. Les critères permettent de tenir compte des lignes sans date de livraison (*bLigneWithoutDateLivraison* à Vrai) et des préparations de livraisons (*bIncludePL* à Vrai). Les données retournées sont équivalentes à la fonction d'interrogation de stock prévisionnel de Sage 100c Gestion commerciale.
QueryMouvementStockPreviArticleMonoGamme(*pArtGamEnum* As IBOArticleGammeEnum3, *pDepot* As IBODepot3, ByVal *dDateDe* As Date, ByVal *dDateA* As Date, ByVal *bLigneWithoutDateLivraison* As Boolean, ByVal *bIncludePL* As Boolean) As IBICollection	Retourne la collection des lignes de document qui mouvementeront le stock pour l'énuméré de gamme, le dépôt et la période passés en paramètres. Les critères permettent de tenir compte des lignes sans date de livraison (*bLigneWithoutDateLivraison* à Vrai) et des préparations de livraisons (*bIncludePL* à Vrai). Les données retournées sont équivalentes à la fonction d'interrogation de stock prévisionnel de Sage 100c Gestion commerciale.
QueryTiers(*pTiers* As IBOTiers3, ByVal *dDateDe* As Date, ByVal *dDateA* As Date, ByVal *TypeDe* As DocumentType, ByVal *TypeA* As DocumentType) As IBICollection	Retourne la collection des lignes de document d'articles pour le tiers, la période et les types de documents passés en paramètres.
QueryTiersArticle(*pTiers* As IBOTiers3, *pArticle* As IBOArticle3, ByVal *dDateDe* As Date, ByVal *dDateA* As Date, ByVal *TypeDe* As DocumentType, ByVal *TypeA* As DocumentType) As IBICollection	Retourne la collection des lignes de document pour l'article, le tiers, la période et les types de documents passés en paramètres.
QueryType(ByVal *TypeDe* As DocumentType, ByVal *TypeA* As DocumentType) As IBICollection	Retourne la collection des lignes de document pour la fourchette de type de document passée en paramètre.
ExistLigne (int DL_No) As Boolean	Teste l'existence de l'identifiant Numéro de ligne DL_No
ReadLigne (int DL_No) As IBODocumentLigne3	Lecture de la ligne document suivant l'identifiant DL_No

IBODocumentReglementFactory

Fabrique un objet Règlement de document

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
QueryTiersOrderDate(pTiers As IBOTiersPart3) As IBICollection	Retourne une collection de règlements de documents pour le client/fournisseur passé en paramètre.
QueryTiersNonImputeOrderDate (pTiers As IBOTiersPart3) As IBICollection	Retourne une collection de règlements de documents non totalement imputés pour le client/fournisseur passé en paramètre.
QueryClientOrderDate() As IBICollection	Retourne une collection de tous les règlements des clients.
QueryFournisseurOrderDate() As IBICollection	Retourne une collection de tous les règlements des fournisseurs.

IBODocumentStockFactory3

Fabrique un objet Document de stock.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
CreateType(ByVal *DO_Type* As DocumentType) As IBODocumentStock3	Créé un nouveau document de stock dont le type est passé en paramètre (Cf. énumérateur DocumentType).
ExistPiece(ByVal *DO_Type* As DocumentType, ByVal *sDO_Piece* As String) As Boolean	Test l'existence d'un document de stock dont le type et le numéro sont passés en paramètres (Cf. énumérateur DocumentType).
QueryDepotOrderType(Depot As IBODepot) As IBICollection	Retourne une collection de documents de stock pour le dépôt passé en paramètre.

Syntaxe	Description
QueryTypeDateOrderPiece(ByVal *DO_Type* As DocumentType, ByVal *DateDeb* As Date, ByVal *DateFin* As Date) As IBICollection	Retourne une collection de documents de stock triée par numéro de pièce, pour le type de document et la période passés en paramètres.
QueryTypeOrderPiece(ByVal *DO_Type* As DocumentType) As IBICollection	Retourne une collection de documents de stock dont le type est passé en paramètre (Cf. énumérateur DocumentType).
QueryTypePieceOrderPiece(ByVal *DO_Type* As DocumentType, ByVal *sDO_PieceDe* As String, ByVal *sDO_PieceA* As String) As IBICollection	Retourne une collection de documents internes (type et plage de numéros de pièce passés en paramètre).
ReadPiece(ByVal *DO_Type* As DocumentType, ByVal *sDO_Piece* As String) As IBODocumentStock3	Retourne un document de stock correspondant au type et au numéro de pièce passés en paramètres (Cf. énumérateur DocumentType).

IBODocumentVenteFactory3

Fabrique un objet Document de vente.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
CreateFacture(ByVal *DO_Provenance* As DocumentProvenanceType) As IBODocumentVente3	Crée une facture de vente en fonction de la provenance passée en paramètre (cf. énumérateur DocumentTypeProvenance).
CreateType(ByVal *DO_Type* As DocumentType) As IBODocumentVente3	Créé un nouveau document de vente dont le type est passé en paramètre (Cf. énumérateur DocumentType).
ExistPiece(ByVal *DO_Type* As DocumentType, ByVal *sDO_Piece* As String) As Boolean	Test l'existence d'un document dont le type et le numéro de pièce sont passés en paramètres (Cf. énumérateur DocumentType).
QueryCollaborateurType(*Collab* As IBOCollaborateur, ByVal *DO_Type* As DocumentType) As IBICollection	Retourne une collection de documents de vente pour le collaborateur et le type passés en paramètres.
QueryTiersOrderDate(*Client* As IBOClient3) As IBICollection	Retourne une collection de documents de vente triés par date en focntion du client passé en paramètre.
QueryTiersTypeOrderDate(*Client* As IBOClient3, ByVal *DO_Type* As DocumentType) As IBICollection	Retourne une collection de documents de vente triée par date en fonction du client et du type passés en paramètres.

Syntaxe	Description
QueryTypeDateOrderPiece(ByVal *DO_Type* As DocumentType, ByVal *DateDeb* As Date, ByVal *DateFin* As Date) As IBICollection	Retourne une collection de documents de vente triée par numéro de pièce, pour le type de document et la période passés en paramètres.
QueryTypeOrderPiece(ByVal DO_Type As DocumentType) As IBICollection	Retourne une collection de documents (triée par numéro de pièce) dont le type est passé en paramètre (Cf. énumérateur DocumentType).
QueryTypePieceOrderPiece(ByVal *DO_Type* As DocumentType, ByVal *sDO_PieceDe* As String, ByVal *sDO_PieceA* As String) As IBICollection	Retourne une collection de documents (triée par numéro de pièce) dont le type et une fourchette de numéros de pièce sont passés en paramètres (Cf. énumérateur DocumentType).
ReadPiece(ByVal *DO_Type* As DocumentType, ByVal *sDO_Piece* As String) As IBODocumentVente3	Retourne un document de vente dont le type et le numéro de pièce sont passés en paramètres (Cf. énumérateur DocumentType).

IBOFamilleFactory2

Fabrique un objet famille.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistCode(ByVal *fType* As FamilleType, ByVal *sNum* As String) As Boolean	Test l'existence d'un objet famille correspondant au type et numéro passés en paramètres. Retourne *True* s'il existe, sinon *False*.
QueryCatalogue(*pProduit* As IBPProduit2) As IBICollection	Retourne une collection d'éléments catalogue correspondant au catalogue passé en paramètre.
ReadCode(ByVal *fType* as FamilleType, ByVal *sNum* As String) As IBOFamille3	Retourne l'objet famille correspondant au type et numéro passés en paramètres.

IBOFournisseurTarifFamilleFactory

Fabrique un objet tarif famille fournisseur.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistFamille(*pFamille* As IBOFamille3) As Boolean	Teste l'existence d'un objet tarif famille fournisseur pour la famille passée en paramètre. Retourne *True* s'il existe, sinon *False*.
ReadFamille(*pFamille* As IBOFamille3) As IBOFamilleTarifFournisseur	Retourne l'objet tarif famille fournisseur correspondant à la famille passée en paramètre.

IBOGlossaireFactory

Fabrique un objet glossaire.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistIntitule(ByVal *GL_Domain* As GlossaireDomaineType, ByVal *sNum* As String) As Boolean	Teste l'existence de l'objet glossaire correspondant au domaine et l'intitulé passés en paramètre. Retourne *True* s'il existe, sinon *False*.
ReadIntitule(ByVal *GL_Domain* As GlossaireDomaineType, ByVal *sIntitule* As String) As IBOGlossaire2	Retourne l'objet glossaire correspondant au domaine et l'intitulé passés en paramètres.

IBORessourceFactory

Fabrique un objet ressource.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	ListOrderDepot() As IBICollection	Retourne la collection des ressources triées par dépôt.
Lecture seule	ListOrderIntitule() As IBICollection	Retourne la collection des ressources triées par intitulé.
Lecture seule	ListOrderType() As IBICollection	Retourne la collection des ressources triées par type.

Méthodes

Syntaxe	Description
ExistNumero(ByVal *sNum* As String) As Boolean	Teste l'existence de la ressource pour l'intitulé passé en parameter.
ReadNumero(ByVal *sNum* As String) As IBORessource	Retourne l'objet ressource correspondant à l'intitulé passé en paramètre.

IBORessourceCentreFactory

Fabrique un centre de charges.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	ListOrderDepot() As IBICollection	Retourne la collection des centres de charges triés par dépôt.
Lecture seule	ListOrderIntitule() As IBICollection	Retourne la collection des centres de charges triés par intitulé.
Lecture seule	ListOrderType() As IBICollection	Retourne la collection des centres de charges triés par type.

Méthodes

Syntaxe	Description
ExistNumero(ByVal *sNum* As String) As Boolean	Teste l'existence du centre de charges pour l'intitulé passé en parameter.
ReadNumero(ByVal *sNum* As String) As IBORessource	Retourne l'objet centre de charges correspondant à l'intitulé passé en paramètre.

IBOInfoComplementClient

Fabrique un objet Information Complement Client.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistCode (ByVal sCode As String) As Boolean	Teste l'existence de l'objet InfoComplementClient du client correspondant au Code passé en paramètre. Retourne *True* s'il existe, sinon *False*.
ReadCode(ByVal *sCode* As String) As IBOInfoComplementClient	Retourne l'objet InfoComplementClient correspondant au code passé en paramètre.

IBOInfoComplementEntete

Fabrique un objet Information Complement Document Vente Entête.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistCode (ByVal sCode As String) As Boolean	Teste l'existence de l'objet InfoComplementEntete du Document correspondant au Code passé en paramètre. Retourne *True* s'il existe, sinon *False*.
ReadCode(ByVal *sCode* As String) As InfoComplementEntete	Retourne l'objet InfoComplementEntete correspondant au code passé en paramètre.

IBOInfoComplementDocligne

Fabrique un objet Information Complement Document ligne de Vente.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
ExistCode (ByVal sCode As String) As Boolean	Teste l'existence de l'objet InfoComplementDocligne de la ligne document correspondant au Code passé en paramètre. Retourne *True* s'il existe, sinon *False*.
ReadCode(ByVal *sCode* As String) As InfoComplementDocligne	Retourne l'objet InfoComplementDocligne correspondant au code passé en paramètre.

Interfaces objets paramètres (IBPxxx)

Correspondance des objets

Ce tableau permet de faire la correspondance entre les objets et les fonctions et tables des applications Sage 100c.

Type objet	Description	Correspondance dans l'application	Table
IBPAgenda	Agenda	Gestion Commerciale / Menu Fichier / Paramètres société / Agenda	P_AGENDA

Type objet	Description	Correspondance dans l'application	Table
IBPArrondi	Mode d'arrondi	Gestion Commerciale / Menu Fichier / Paramètres société / Articles / Mode d'arrondi	P_ARRONDI
IBPArticleStat	Champ statistique articles	Gestion Commerciale / Menu Fichier / Paramètres société / Articles / Champ statistique	P_INTSTATART
IBPCategorieComptaAchat	Catégorie comptable d'achat	Gestion Commerciale / Menu Fichier / Paramètres société / Comptabilisation / Catégorie comptable / Domaine Achat	P_CATCOMPTA
IBPCategorieComptaStock	Catégorie comptable de stock	Gestion Commerciale / Menu Fichier / Paramètres société / Comptabilisation / Catégorie comptable / Domaine Stock	P_CATCOMPTA
IBPCategorieComptaVente	Catégorie comptable de vente	Gestion Commerciale / Menu Fichier / Paramètres société / Comptabilisation / Catégorie comptable / Domaine Vente	P_CATCOMPTA
IBPCategorieTarif	Catégorie tarifaire	Gestion Commerciale / Menu Fichier / Paramètres société / Tiers / Catégorie tarifaire	P_CATTARIF
IBPConditionLivraison	Condition de livraison	Gestion Commerciale / Menu Fichier / Paramètres société / Logistique / Condition de livraison	P_CONDLIVR
IBPConditionnement	Conditionnement	Gestion Commerciale / Menu Fichier / Paramètres société / Articles / Conditionnement	P_CONDITIONNEMENT
IBPDossierCial	Dossier commercial	Gestion Commerciale / Menu Fichier / Paramètres société / Identification / Monnaie & formats	P_DOSSIERCIAL
IBPDossierParamCial	Paramétrage dossier commercial	Gestion Commerciale / Menu Fichier / Paramètres société / Logistique	P_PARAMETRECIAL
IBPExpedition3	Mode d'expédition	Gestion Commerciale / Menu Fichier / Paramètres société / logistique / Mode d'expédition	P_EXPEDITION
IBPGamme	Gamme	Gestion Commerciale / Menu Fichier / Paramètres société / Articles / Gamme	P_GAMME
IBPParamDocInterne	Paramétrage des documents internes	Gestion Commerciale / Menu Fichier / Paramètres société / Documents / Type de documents Internes	P_INTERNE
IBPPeriodicite	Périodicité	Gestion Commerciale / Menu Fichier / Paramètres société / Tiers / Périodicité de livraison	P_PERIOD

Type objet	Description	Correspondance dans l'application	Table
IBPProduit2	Catégorie de produits	Gestion Commerciale / Menu Fichier / Paramètres société / Article / Catalogue articles	P_PRODUIT
IBPSoucheAchat	Souche des documents d'achat	Gestion Commerciale / Menu Fichier / Paramètres société / Documents / Souche et circuit de validation / Documents des achats	P_SOUCHEACHAT
IBPSoucheInterne	Souche des documents interne	Gestion Commerciale / Menu Fichier / Paramètres société / Documents / Souche et circuit de validation / Documents internes	P_SOUCHEINTERNE
IBPSoucheVente	Souche des documents de vente	Gestion Commerciale / Menu Fichier / Paramètres société / Documents / Souche et circuit de validation / Documents des ventes	P_SOUCHEVENTE
IBPUnite	Unité d'achat et de vente	Gestion Commerciale / Menu Fichier / Paramètres société / Articles / Unité d'achat et de vente	P_UNITE
IBPMotifDevis	Motifs devis perdus	Gestion Commerciale / Menu Fichier / Paramètres société / Documents / Motifs devis perdus	P_MOTIFDEVIS

Liens entre les objets

Le tableau ci-dessous permet pour chaque type d'objet, de retrouver la propriété *Factory* à partir de laquelle l'objet peut être fabriqué. Ces propriétés sont toutes issues de l'objet maître *BSCIALApplication100c.*

Exemple :

Type d'objet IBPCategorieTarif :

- *L'objet **IBPCategorieTarif** (Catégorie tarifaire) est issu d'un objet maître de type **BSCIALApplication100c** ;*
- *L'objet maître fabrique un objet **IBPCategorieTarif** par appel de la propriété **FactoryCategorieTarif()**.*

Interface objet	Propriété factory objet	Description objet
IBPAgenda	FactoryAgendaConfig()	Agenda
IBPArrondi	FactoryArrondi()	Mode d'arrondi
IBPArticleStat	FactoryArticleStat()	Champ statistique articles

Interface objet	Propriété factory objet	Description objet
IBPCategorieComptaAchat	FactoryCategorieComptaAchat()	Catégorie comptable d'achat
IBPCategorieComptaStock	FactoryCategorieComptaStock()	Catégorie comptable de stock
IBPCategorieComptaVente	FactoryCategorieComptaVente()	Catégorie comptable de vente
IBPCategorieTarif	FactoryCategorieTarif()	Catégorie tarifaire
IBPConditionLivraison	FactoryConditionLivraison()	Condition de livraison
IBPConditionnement	FactoryConditionnement()	Conditionnement
IBPDossierCial	FactoryDossier()	Dossier commercial
IBPDossierParamCial	FactoryDossierParam()	Paramétrage dossier commercial.
IBPExpedition3	FactoryExpedition()	Mode d'expédition
IBPGamme	FactoryGamme()	Gamme
IBPParamDocInterne	FactoryParamDocInterne()	Paramétrage des documents internes
IBPPeriodicite	FactoryPeriodicite()	Périodicité
IBPProduit2	FactoryProduit()	Catégorie de produit
IBPSoucheAchat	FactorySoucheAchat()	Souche des documents d'achat
IBPSoucheInterne	FactorySoucheInterne()	Souche des documents interne
IBPSoucheVente	FactorySoucheVente()	Souche des documents de vente
IBPUnite	FactoryUnite()	Unité d'achat et de vente
IBP_MotifDevis	FactoryMotifDevis()	Motifd devis perdus

IBPAgenda

Agenda.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecure seule	Enums() As IBIValuesInserttable	Enumérés de l'agenda.
Lecture seule	FactoryAgenda() As IBPAgendaFactory	Fabrique un objet Agenda.
Lecture / Ecriture	Indisponible() As Boolean	Rend la ressource indisponible (valable uniquement pour les agendas de type ressource).

Accès	Syntaxe	Description
Lecture / Ecriture	Intitule() As String	Intitulé.
Lecture/ Ecriture	Type() As AgendaTypeInteresse	Type d'agenda (cf. énumérateur AgendaTypeInteresse).

IBPArrondi

Mode d'arrondi.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	AR_Type() As ArrondiType	Type d'arrondi (cf. énumérateur ArrondiType).
Lecture / Ecriture	AR_Valeur() As Double	Valeur.
Lecture seule	FactoryArrondi() As IBITypeObjectFactory	Fabrique un Objet Arrondi.

Méthode

Syntaxe	Description
Arrondir(ByVal dVal As Double) As Double	Arrondi la valeur passée en paramètre et retourne la valeur arrondie.

IBPCategorieComptaAchat

Catégorie comptable achat.

Interface héritée

Syntaxe	Description
IBICategorieCompta	Cf. Interface IBICategorieCompta pour les propriétés et méthodes héritées.

Propriété

Accès	Syntaxe	Description
Lecture seule	FactoryCategorieComptaAchat() As IBPCategorieComptaAchatFactory	Fabrique un objet Catégorie comptable achat.

IBPCategorieComptaStock

Catégorie comptable de stock.

Interface héritée

Syntaxe	Description
IBICategorieCompta	Cf. Interface IBICategorieCompta pour les propriétés et méthodes héritées.

Propriété

Accès	Syntaxe	Description
Lecture seule	FactoryCategorieComptaStock() As IBPCategorieComptaStockFactory	Fabrique un objet Catégorie comptable stock.

IBPCategorieComptaVente

Catégorie comptable de vente.

Interface héritée

Syntaxe	Description
IBICategorieCompta	Cf. Interface IBICategorieCompta pour les propriétés et méthodes héritées.

Propriété

Accès	Syntaxe	Description
Lecture seule	FactoryCategorieComptaVente() As IBPCategorieComptaVenteFactory	Fabrique un objet catégorie comptable de vente.

IBPCategorieTarif

Catégorie tarifaire.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	CT_Intitule() As String	Intitulé.
Lecture / Ecriture	CT_PrixTTC() As Boolean	Type tarif : True : TTC False : HT
Lecture seule	FactoryCategorieTarif() As IBPCategorieTarifFactory	Fabrique un objet Catégorie tarifaire.

IBPConditionLivraison

Condition de livraison.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	C_Intitule() As String	Intitulé.
Lecture / Ecriture	C_Mode() As String	Code condition de livraison.
Lecture seule	FactoryConditionLivraison() As IBPConditionLivraisonFactory	Fabrique un objet Condition de livraison.

IBPConditionnement

Conditionnement.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	FactoryConditionnement() As IBPConditionnementFactory	Fabrique un objet Conditionnement.

Accès	Syntaxe	Description
Lecture seule	FactoryConditionnementEnum() As IBPConditionnementEnumFactory	Fabrique un objet énuméré de conditionnement.
Lecture / Ecriture	Intitule() As String	Intitulé.

IBPConditionnementEnum

Enumérés de conditionnement.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Conditionnement() As IBPConditionnement	Conditionnement.
Lecture /Ecriture	EC_EDICode	Code EDI énuméré de conditionnement.
Lecture / Ecriture	Intitulé	Intitulé de l'énuméré de conditionnement.
Lecture / Ecriture	Quantite() As Double	Quantité.

Il n'est pas possible de modifier directement un énuméré de conditionnement. Ainsi, pour modifier l'intitulé ou la quantité d'un énuméré, il convient de supprimer puis recréer l'enuméré.

IBPDossierCial

Dossier commercial.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	D_ArchivePeriod() As Date	Date de dernier archivage.

Accès	Syntaxe	Description
Lecture seule	D_DerniereCloture() As Date	Date de dernière clôture.
Lecture / Ecriture	D_FormatPrix() As String	Format prix.
Lecture / Ecriture	D_FormatQte() As String	Format quantité.
Lecture / Ecriture	D_RaisonS() As String	Raison sociale.
Lecture / Ecriture	D_Validite Period() As Integer	Période de validité de saisie de documents.
Lecture / Ecriture	DeviseCompte() As IBPDevise2	Devise de tenue de compte.
Lecture / Ecriture	DeviseEquivalence() As IBPDevise2	Devise d'équivalence.
Lecture seule	FactoryDossier() As IBITypeObjectFactory	Fabrique un objet dossier commercial.

IBPDossierParamCial

Dossier commercial.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	AnalytiqueAffaire() As IBPAnalytique3	Plan analytique affaire.
Lecture / Ecriture	AnalytiqueArticle() As IBPAnalytique3	Plan analytique article.
Lecture seule	DebutExercice() As Date	Date de début d'exercice commercial pour l'année courante.
Lecture seule	DebutExerciceFirst() As Date	Date de début du premier exercice commercial (exercice contenant le document commercial le plus ancien).
Lecture / Ecriture	Emplacement_CanBeControle() As Boolean	Gestion des emplacements de contrôle.
Lecture / Ecriture	Emplacement_IsMulti() As Boolean	Gestion multi-emplacements par dépôt.
Lecture / Ecriture	Emplacement_PrioriteDestockage() As DossierparamEmplacementPriorite	Priorité de déstockage (cf. énumérateur DossierParamEmplacementPriorite).
Lecture seule	FactoryDossierParam() As IBITypeObjectFactory	Fabrique un objet paramétrage dossier commercial.
Lecture seule	FinExercice() As Date	Date de fin d'exercice commercial pour l'année courante (correspond à DebutExercice + 12 mois – 1 jour).

Accès	Syntaxe	Description
Lecture seule	FinExerciceFirst() As Date	Date de fin du premier exercice commercial (DebutExerciceFirst + 12 mois – 1 jour)
Lecture seule	IsInterditSommeil() As Boolean	Interdire l'utilisation des éléments mis en sommeil.

IBPExpedition3

Mode d'expédition.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	ArticleFraisExpedition() As IBOArticle3	Article de frais d'expédition.
Lecture / Ecriture	E_Intitule() As String	Intitulé.
Lecture / Ecriture	E_Mode() As String	Code mode de transport.
Lecture / Ecriture	E_TypeFrais() As DocumentFraisType	Type des frais d'expédition (cf. énumérateur DocumentFraisType).
Lecture / Ecriture	E_TypeFraisTTC() As Boolean	Frais d'expédition TTC (True) ou HT (False).
Lecture / Ecriture	E_TypeFranco() As DocumentFraisType	Type du franco de port (cf. énumérateur DocumentFraisType).
Lecture / Ecriture	E_TypeFrancoTTC() As Boolean	Franco TTC (True) ou HT (False).
Lecture / Ecriture	E_ValFrais() As Double	Valeur des frais d'expédition.
Lecture / Ecriture	E_ValFranco() As Double	Valeur du franco d'expédition.
Lecture seule	FactoryExpedition() As IBPExpeditionFactory3	Fabrique un objet Mode d'expédition.

IBPGamme

Gamme.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Enums() As IBIValuesInsertable	Enumérés de la gamme.
Lecture seule	FactoryGamme() As IBPGammeFactory	Fabrique un objet Gamme.
Lecture / Ecriture	G_Intitule() As String	Intitulé.
Lecture / Ecriture	G_Type() As GammeType	Type (cf. énumérateur GammeType).

IBPPeriodicite

Périodicité.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	FactoryPeriodicite() As IBPPeriodiciteFactory	Fabrique un objet Périodicité.
Lecture / Ecriture	P_Intitule() As String	Intitulé.

IBPProduit2

Catalogue produit.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	CatalogueParent() As IBPProduit2	Catalogue parent.
Lecture / Ecriture	CL_Code() As String	Code du catalogue produit.
Lecture / Ecriture	CP_Intitule() As String	Intitulé.
Lecture / Ecriture	CP_Stock() As Boolean	Suivi en gestion de stock : True : Oui False : Non
Lecture seule	FactoryProduit() As IBPProduitFactory2	Fabrique un objet Catégorie de produit.
Lecture seule	FactorySousCatalogue() As IBPProduitFactory2	Fabrique un objet sous catalogue du catalogue produit.

IBPUnite

Unité d'achat et de vente.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	CorrespondancePlanning() As Boolean	Unité ayant une correspondance avec le planning.
Lecture seule	FactoryUnite() As IBPUniteFactory	Fabrique un objet Unité d'achat et de vente.
Lecture / Ecriture	Intitule() As String	Intitulé.
Lecture / Ecriture	NbUnite() As Integer	Nombre d’unité.
Lecture / Ecriture	UniteTemps() As UnitePlanning	Unité de temps.

IBPParamDocInterne

Paramétrage des documents internes.

Interface héritée

Syntaxe	Description
IBIParamDoc	Cf. Interface IBIParamDoc pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	D_MvtStock() As DocumentInterneMvtType	Type de mouvement de stock.
Lecture / Ecriture	DefaultDepot() As IBODepot3	Dépôt par défaut.

IBPSoucheAchat

Souche et numérotation des documents d'achat.

Interface héritée

Syntaxe	Description
IBISouche	Cf. Interface IBISouche pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Journal() As IBOJournal3	Journal d'achat.
Lecture / Ecriture	JournalSituation() As IBOJournal3	Journal de situation.

IBPSoucheVente

Souche et numérotation des documents de vente.

Interface héritée

Syntaxe	Description
IBISouche	Cf. Interface IBISouche pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Journal() As IBOJournal3	Journal de vente.
Lecture / Ecriture	JournalSituation() As IBOJournal3	Journal de situation.

IBPArticleStat

Champ statistique article.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Enums() As IBIValuesInsertable	Fabrique un objet champ statistique article.
Lecture seule	FactoryArticleStat() As IBPArticleStatFactory	Fabrique un objet champ statistique article.
Lecture / Ecriture	Intitule() As String	Intitulé du champ statistique.

IBPMotifDevis

Motifs devis perdus.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	FactoryMotifDevis() As IBPMotifDevisFactory	Fabrique un objet Motif devis perdu
Lecture / Ecriture	Intitule() As String	Intitulé.

Interfaces objets métiers (IBOxxx)

Correspondance des objets

Ce tableau permet de faire la correspondance entre les objets et les fonctions et tables des applications Sage 100c.

Objet	Description	Correspondance dans l'application	Table correspondante
IBOAgenda	Evénements agenda	Gestion Commerciale / Menu Traitement / Evénements agenda	F_AGENDA

Objet	Description	Correspondance dans l'application	Table correspondante
IBOArticle3	Article	Gestion Commerciale / Menu Structure / Articles / Fenêtre Article	F_ARTICLE
IBOArticleCond3	Conditionnement article	Gestion Commerciale / Menu Structure / Articles / Fenêtre Article / Onglet Conditionnement	F_CONDITION
IBOArticleDepot3	Dépôt de stockage article	Gestion Commerciale / Menu Structure / Articles / Fenêtre Article / Onglet Paramètres / Dépôt	F_ARTSTOCK
IBOArticleDepotGamme3	Dépôt de stockage pour un article à gamme	Gestion Commerciale / Menu Structure / Articles / Fenêtre Article / Onglet Paramètres / Dépôt	F_GAMSTOCK
IBOArticleDepotLot	Numéros de Lot/Série par article et par dépôt	Gestion Commerciale / Menu Structure / Articles / Fenêtre Article / Fonction Interroger	F_LOTSERIE
IBOArticleGammeEnum3	Enuméré de gamme article	Gestion Commerciale / Menu Structure / Articles / Fenêtre Article / Onglet Gamme	F_ARTGAMME
IBOArticleGammeEnumRef3	Référence énuméré de gamme article	Gestion Commerciale / Menu Structure / Articles / Fenêtre Article / Onglet Gamme	F_ARTENUMREF F_ARTPRIX (propriété AR_PUNet)
IBOArticleMedia3	Information multimédia de l'article (photo/document)	Gestion Commerciale / Menu Structure / Articles / Fenêtre Article / Onglet Champs libres / Documents rattachés	F_ARTICLEMEDIA
IBOArticleNomenclature3	Composant de nomenclature article	Gestion Commerciale / Menu Structure / Nomenclatures / Fenêtre Nomenclature	F_NOMENCLAT
IBOArticleParamCompta3	Paramétrage comptable de l'article	Gestion Commerciale / Menu Structure / Articles / Fenêtre Article / Onglet Paramètres / Comptabilité	F_ARTCOMPTA
IBOArticleTarifCategorie3	Tarif par catégorie tarifaire article	Gestion Commerciale / Menu Structure / Articles / Fenêtre Article / Onglet Identification / Onglet Catégories tarifaires	F_ARTCLIENT
IBOArticleTarifClient3	Tarif client	Gestion Commerciale / Menu Structure / Articles / Fenêtre Article / Onglet Identification / Onglet Tarifs clients	F_ARTCLIENT
IBOArticleTarifCondCategorie3	Tarif par conditionnement d'une catégorie tarifaire article	Pour un article à conditionnement uniquement : Gestion Commerciale / Menu Structure / Articles / Fenêtre Article / Onglet Catégories tarifaires / Sélection d'une Catégorie tarifaire / Tarif par énuméré de conditionnement	F_TARIFCOND
IBOArticleTarifCondClient3	Tarif par conditionnement pour	Pour un article à conditionnement uniquement :	F_TARIFCOND

Objet	Description	Correspondance dans l'application	Table correspondante
	un tarif client d'un article	Gestion Commerciale / Menu Structure / Articles / Fenêtre Article / Onglet Tarifs clients / Sélection d'un Tarif client / Tarif par énuméré de conditionnement	
IBOArticleTarifFournisseur3	Tarif fournisseur d'un article	Gestion Commerciale / Menu Structure / Articles / Fenêtre Article / Onglet Fournisseurs	F_ARTFOURNISS
IBOArticleTarifGammeCategorie3	Tarif pour une catégorie tarifaire d'un article à gamme	Pour un article à gamme uniquement : Gestion Commerciale / Menu Structure / Articles / Fenêtre Article / Onglet Catégories tarifaires / Sélection d'une Catégorie tarifaire / Tarif par énuméré de gamme	F_TARIFGAM
IBOArticleTarifGammeClient3	Tarif pour un client d'un article à gamme	Pour un article à gamme uniquement : Gestion Commerciale / Menu Structure / Articles / Fenêtre Article / Onglet Tarifs clients / Sélection d'un Tarif client / Tarif par énuméré de gamme	F_TARIFGAM
IBOArticleTarifGammeFournisseur3	Tarif pour un fournisseur d'un article à gamme	Pour un article à gamme uniquement : Gestion Commerciale / Menu Structure / Articles / Fenêtre Article / Onglet Fournisseurs / Sélection d'un fournisseur / Tarif par énuméré de gamme	F_TARIFGAM
IBOArticleTarifQteCategorie3	Tarif par quantité ou montant pour une catégorie tarifaire	Gestion Commerciale / Menu Structure / Articles / Fenêtre Article / Onglet Catégories tarifaires / Sélection d'une catégorie tarifaire / Remise	F_TARIFQTE
IBOArticleTarifQteClient3	Tarif par quantité ou montant pour pour un client	Gestion Commerciale / Menu Structure / Articles / Fenêtre Article / Onglet Tarifs clients / Sélection d'un tarif client / Remise	F_TARIFQTE
IBOArticleTarifQteFournisseur3	Tarif par quantité ou montant pour pour un fournisseur	Gestion Commerciale / Menu Structure / Articles / Fenêtre Article / Onglet Fournisseurs / Sélection d'un fournisseur / Remise	F_TARIFQTE
IBOBaremeCommission	Commissions	Gestion Commerciale / Menu Structure / Barèmes / Commissions	F_TARIF
IBOBaremeCommissionQte	Gamme de remise commissions	Gestion Commerciale / Menu Structure / Barèmes / Commissions / Fenêtre commissions	F_TARIFREMISE
IBOBaremeRabais	Rabais, remises et ristournes	Gestion Commerciale / Menu Structure / Barèmes / Rabais, remises et ristournes	F_TARIF

Objet	Description	Correspondance dans l'application	Table correspondante
IBOBaremeRabaisQte	Gamme de remise rabais, remises et ristournes	Gestion Commerciale / Menu Structure / Barèmes / Rabais, remises et ristournes / Fenêtre Rabais, remises et ristournes	F_TARIFREMISE
IBOBaremeSolde	Soldes et promotions	Gestion Commerciale / Menu Structure / Barèmes / Soldes et promotions	F_TARIF
IBODepot3	Dépôt de stockage	Gestion Commerciale / Menu Structure / Dépôts de stockage / Fenêtre Dépôt	F_DEPOT
IBODepotContact3	Contact du dépôt de stockage	Gestion Commerciale / Menu Structure / Dépôts de stockage / Fenêtre Dépôt / Onglet Contacts	F_DEPOTCONTACT
IBODepotEmplacement	Emplacement dépôt	Gestion Commerciale / Menu Structure / Dépôts de stockage / Fenêtre Dépôt / Onglet Emplacement	F_DEPOTEMPL
IBODocument3	Document	Gestion Commerciale / Menu Traitement / Documents des ventes, des achats, des stocks et internes / Fenêtre Document	F_DOCENTETE
IBODocumentAchat3	Document des achats	Gestion Commerciale / Menu Traitement / Documents des achats / Fenêtre Document	F_DOCENTETE
IBODocumentAchatLigne3	Ligne de document des achats	Gestion Commerciale / Menu Traitement / Documents des achats / Fenêtre Document	F_DOCLIGNE
IBODocumentAcompte3	Acompte	Gestion Commerciale / Menu Traitement / Document des ventes / Fenêtre Document / Bouton fonction Pied / Acomptes	F_DOCREGL
IBODocumentEcheance3	Echéance de règlement	Gestion Commerciale / Menu Traitement / Document des ventes / Fenêtre Document / Bouton fonction Pied/ Echéances	F_DOCREGL
IBODocumentInterne3	Document interne	Gestion Commerciale / Menu Traitement / Documents internes / Fenêtre Document	F_DOCENTETE
IBODocumentInterneLigne3	Ligne de document interne	Gestion Commerciale / Menu Traitement / Documents internes / Fenêtre Document	F_DOCLIGNE
IBODocumentLigne3	Ligne de document	Gestion Commerciale / Menu Traitement / Documents des ventes, des achats, des stocks et internes / Fenêtre Document	F_DOCLIGNE
IBODocumentLigneEmplacement	Emplacements associés à la ligne	Gestion Commerciale / Menu Traitement / Documents des ventes, des achats, des stocks et internes /	F_DOCLIGNEEMPL

Objet	Description	Correspondance dans l'application	Table correspondante
		Sélection d'une ligne / Bouton Actions : Saisir les emplacements	
IBODocumentLigneLienCM	Liens contremarque associés à la ligne	Gestion Commerciale / Menu Traitement / Documents des ventes, des achats, des stocks et internes / Sélection d'une ligne / Bouton Actions : Voir les informations sur le document lié	F_CMLIEN
IBODocumentMedia	Documents média associés aux documents	Gestion commerciale / Menu Traitement / Documents des ventes, des achats, des stocks et internes / Bouton Fonctions : Documents liés	F_DOCENTETEMEDIA
IBODocumentPart3	Document faisant référence à un partenaire commercial (vente, achat et interne)	Gestion Commerciale / Menu Traitement / Documents des ventes, des achats et internes / Fenêtre Document	F_DOCENTETE
IBODocumentPartLigne3	Ligne de document faisant référence à un tiers de type client ou fournisseur (vente, achat et interne)	Gestion Commerciale / Menu Traitement / Documents des ventes, des achats et internes / Fenêtre Document	F_DOCLIGNE
IBODocumentReglement	Règlement client ou fournisseur	Gestion Commerciale / Menu Traitement / Gestion des règlements / Saisie des règlements clients ou fournisseurs	F_CREGLEMENT
IBODocumentStock3	Document des stocks	Gestion Commerciale / Menu Traitement / Documents des stocks / Fenêtre Document	F_DOCENTETE
IBODocumentStockLigne3	Ligne de document des stocks	Gestion Commerciale / Menu Traitement / Documents des stocks / Fenêtre Document	F_DOCLIGNE
IBODocumentVente3	Document des ventes	Gestion Commerciale / Menu Traitement / Documents des ventes / Fenêtre Document	F_DOCENTETE
IBODocumentVenteLigne3	Ligne des ventes	Gestion Commerciale / Menu Traitement / Documents des ventes / Fenêtre Document	F_DOCLIGNE
IBOFamille3	Famille d'article	Gestion Commerciale / Menu Structure / Famille / Fenêtre Famille	F_FAMILLE
IBOFamilleParamCompta3	Paramètres comptables de la famille	Gestion Commerciale / Menu Structure / Famille / Fenêtre Famille / Onglet Paramètres / Comptabilité	F_FAMCOMPTA
IBOFamilleTarifCategorie	Tarif famille catégorie	Gestion Commerciale / Menu Structure / Famille / Fenêtre Famille / Onglet Tarifs / Catégorie tarifaire	F_FAMTARIF

Objet	Description	Correspondance dans l'application	Table correspondante
IBOFamilleTarifClient	Tarif famille client	Gestion Commerciale / Menu Structure / Famille / Fenêtre Famille / Onglet Tarifs / Remise par client	F_FAMCLIENT
IBOFamilleTarifFournisseur	Tarif famille fournisseur	Gestion Commerciale / Menu Structure / Famille / Fenêtre Famille / Onglet Tarifs / Fournisseur	F_FAMFOURNISS
IBOFamilleTarifQteCategorie	Gamme de remise de tarif famille catégorie	Gestion Commerciale / Menu Structure / Famille / Fenêtre Famille / Onglet Tarifs / Catégorie tarifaire	F_FAMTARIFQTE
IBOFamilleTarifQteFournisseur	Gamme de remise de tarif fournisseur	Gestion Commerciale / Menu Structure / Famille / Fenêtre Famille / Onglet Tarifs / Fournisseur	F_FAMTARIFQTE
IBOGlossaire2	Glossaire	Gestion Commerciale / Menu Structure / Glossaires / Fenêtre Identification	F_GLOSSAIRE
IBOModele2	Modèle d'enregistrement	Gestion Commerciale / Menu Structure / Modèles d'enregistrement / Fenêtre Modèle	F_MODELE
IBORessource	Ressources	Gestion Commerciale / Menu Structure / Ressources	F_RESSOURCEPROD
IBORessourceCentre	Centres de charges	Gestion Commerciale / Menu Structure / Centres de charges	F_RESSOURCEPROD

Liens entre les objets

Le tableau ci-dessous permet pour chaque type d'objet de retrouver :

- L'interface de l'objet maître dont est issu l'objet ;
- la propriété *Factory* de l'objet maître permettant de fabriquer l'objet ;
- les différentes propriétés de l'objet permettant de fabriquer des sous-objets ;
- l'interface des sous-objets fabriqués.

Exemple :

IBOArticle3 :

- *L'objet **IBOArticle3** (Article) est issu d'un objet maître de type **BSCIALApplication100c** ;*
- *L'objet maître fabrique un objet **IBOArticle3** par appel de la propriété **FactoryArticle()** ;*
- *L'objet **IBOArticle3** propose différents Factory dont la propriété **FactoryArticleCond()** ;*
- *Les sous-objets fabriqués par appel de la propriété **FactoryArticleCond()** de l'objet **IBOArticle3** sont de type **IBOArticleCond3** (Conditionnement article).*

Chaque objet propose une propriété permettant de créer un objet de même type. Il ne s'agit pas d'un sous-objet, par conséquent cette propriété n'est pas listée dans la colonne « Propriété factory sous-objet ».

Exemple :

Les objets de type IBOArticle3 *proposent une propriété* FactoryArticle *permettant de créer un nouvel objet de type* IBOArticle3*. Cette propriété n'apparaît donc pas sous la colonne « Propriété factory sous-objet ».*

Interface objet *Interface objet maître* Propriété factory objet	**Description objet**	**Propriété factory sous-objet** *Interface sous-objet*	**Description sous-objet**
IBOAgenda *BSCIALApplication100c* FactoryAgenda()	Agenda	**-**	-
IBOArticle3 *BSCIALApplication100c* FactoryArticle()	Article	**FactoryArticleCond()** *IBOArticleCond3*	Conditionnement article
		FactoryArticleDepot() *IBOArticleDepot3*	Dépôt de stockage article
		FactoryArticleGammeEnum1() *IBOArticleGammeEnum3*	Enuméré de gamme 1 article
		FactoryArticleGammeEnum2() *IBOArticleGammeEnum3*	Enuméré de gamme 2 article
		FactoryArticleGlossaire() *IBOGlossaire2*	Glossaire
		FactoryArticleMedia() *IBOArticleMedia3*	Information multimédia de l'article (photo/document)
		FactoryArticleNomenclature() *IBOArticleNomenclature3*	Composant de nomenclature article
		FactoryArticleParamCompta() *IBOArticleParamCompta3*	Paramétrage comptable de l'article

Interface objet *Interface objet maître* Propriété factory objet	**Description objet**	**Propriété factory sous-objet** *Interface sous-objet*	**Description sous-objet**
		FactoryArticleTarifCategorie() *IBOArticleTarifCategorie3*	Tarif par catégorie tarifaire article
		FactoryArticleTarifClient() *IBOArticleTarifClient3*	Tarif client
		FactoryArticleTarifFournisseur() *IBOArticleTarifFournisseur3*	Tarif fournisseur d'un article
		FactoryAticleRessource() *IBOArticleRessource*	Ressource associée à l'article
IBOArticleCond3 *IBOArticle3* FactoryArticleCond()	Conditionnement article	-	-
IBOArticleDepot3 *IBOArticle3* FactoryArticleDepot()	Dépôt de stockage article	**FactoryArticleDepotGamme()** *IBOArticleDepotGamme2*	Dépôt de stockage article à gamme
IBOArticleDepotGamme3 *IBOArticleDepot3* FactoryArticleDepotGamme()	Dépôt de stockage article à gamme	-	-
IBOArticleDepotLot *IBOArticleDepot3* FactoryArticleDepotLot**()**	Lot/série sur dépôt de stockage	**-**	-
IBOArticleGammeEnum3 *IBOArticle3* FactoryArticleGammeEnum1() FactoryArticleGammeEnum2()	Enuméré de gamme article	**FactoryArticleGammeEnumRef()** *IBOArticleGammeEnumRef3*	Référence énuméré de gamme article

Interface objet *Interface objet maître* Propriété factory objet	**Description objet**	**Propriété factory sous-objet** *Interface sous-objet*	**Description sous-objet**
IBOArticleGammeEnumRef3 *IBOArticleGammeEnum3* FactoryArticleGammeEnumRef()	Référence énuméré de gamme article	-	-
IBOArticleMedia3 *IBOArticle3* FactoryArticleMedia()	Information multimédia de l'article (photo/document)	-	-
IBOArticleNomenclature3 *IBOArticle3* FactoryArticleNomenclature()	Composant de nomenclature article	-	-
IBOArticleParamCompta3 *IBOArticle3* FactoryArticleParamCompta()	Paramétrage comptable de l'article	-	-
IBOArticleTarifCategorie3 *IBOArticle3* FactoryArticleTarifCategorie()	Tarif par catégorie tarifaire article	**FactoryArticleTarifCond()** *IBOArticleTarifCondCategorie3*	Tarif pour une catégorie tarifaire d'un article à conditionnement
		FactoryArticleTarifGamme() *IBOArticleTarifGammeCategorie3*	Tarif pour une catégorie tarifaire d'un article à gamme
		FactoryArticleTarifQte() *IBOArticleTarifQteCategorie2*	Tarif par quantité, montant ou prix net pour une catégorie tarifaire d'un article
IBOArticleTarifClient3 *IBOArticle3* FactoryArticleTarifClient()	Tarif client d'un article	**FactoryArticleTarifCond()** *IBOArticleTarifCondClient3*	Tarif client pour un article à conditionnement
		FactoryArticleTarifGamme() *IBOArticleTarifGammeClient3*	Tarif client pour un article à gamme

Interface objet *Interface objet maître* Propriété factory objet	**Description objet**	**Propriété factory sous-objet** *Interface sous-objet*	**Description sous-objet**
		FactoryArticleTarifQte() *IBOArticleTarifQteClient3*	Tarif par quantité, montant ou prix net pour un client d'un article
IBOArticleTarifCondCategorie3 *IBOArticleTarifCategorie3* FactoryArticleTarifCond()	Tarif par conditionnement d'une catégorie tarifaire article	-	-
IBOArticleTarifCondClient3 *IBOArticleTarifClient3* FactoryArticleTarifCond()	Tarif par conditionnement pour un tarif client d'un article	-	-
IBOArticleTarifFournisseur2 *IBOArticle3* FactoryArticleTarifFournisseur()	Tarif fournisseur d'un article	**FactoryArticleTarifGamme()** *IBOArticleTarifGammeFournisseur3*	Tarif fournisseur pour un article à gamme
		FactoryArticleTarifQte() *IBOArticleTarifQteFournisseur3*	Tarif par quantité, montant ou prix net pour un fournisseur d'un article
IBOArticleTarifGammeCategorie3 *IBOArticleTarifCategorie3* FactoryArticleTarifGamme()	Tarif pour une catégorie tarifaire d'un article à gamme	-	-
IBOArticleTarifGammeClient3 *IBOArticleTarifClient3* FactoryArticleTarifGamme()	Tarif pour un client d'un article à gamme	-	-
IBOArticleTarifGammeFournisseur3 *IBOArticleTarifFournisseur3* FactoryArticleTarifGamme()	Tarif pour un fournisseur d'un article à gamme	-	-

Interface objet *Interface objet maître* Propriété factory objet	**Description objet**	**Propriété factory sous-objet** *Interface sous-objet*	**Description sous-objet**
IBOArticleTarifQteCategorie3 *IBOArticleTarifCategorie3* FactoryArticleTarifQte()	Tarif par quantité, montant ou prix net pour une catégorie tarifaire	-	-
IBOArticleTarifQteClient3 *IBOArticleTarifClient3* FactoryArticleTarifQte()	Tarif par quantité, montant ou prix net pour un client	-	-
IBOArticleTarifQteFournisseur3 *IBOArticleTarifFournisseur3* FactoryArticleTarifQte()	Tarif par quantité, montant ou prix net pour un fournisseur	-	-
IBOBaremeCommission *BSCIALApplication100c* FactoryBaremeCommission()	Commissionnements	**FactoryBaremeCommissionQte()** *IBOBaremeCommissionQte*	Gamme de remise de commissionnement
IBOBaremeRabais *BSCIALApplication100c* FactoryBaremeRabais()	Rabais, remises et ristournes	**FactoryBaremeRabaisQte()** *IBOBaremeRabaisQte*	
IBOBaremeSolde *BSCIALApplication100c* FactoryBaremeSolde()	Soldes et promotions	**-**	-
IBODepot3 *BSCIALApplication100c* FactoryDepot()	Dépôt de stockage	**FactoryDepotContact()** *IBODepotContact3*	Contact dépôt
		FactoryDepotEmplacement() *IBODepotEmplacement*	Emplacement dépôt
IBODepotContact3 *IBODepot3* FactoryDepotContact()	Contact du dépôt de stockage	-	-

Interface objet *Interface objet maître* Propriété factory objet	**Description objet**	**Propriété factory sous-objet** *Interface sous-objet*	**Description sous-objet**
IBODepotEmplacement *IBODepot3* FactoryDepotEmplacement	Emplacement dépôt	**-**	-
IBODocument3 *BSCIALApplication100c* FactoryDocument()	Document	**FactoryDocumentLigne**() *IBODocumentLigne3*	Lignes de document.
		FactoryDocumentMedia() *IBODocumentMedia*	Document lié
IBODocumentAchat3 *BSCIALApplication100c* FactoryDocumentAchat()	Document des achats	**FactoryDocumentAcompte()** *IBODocumentAcompte3*	Acomptes.
		FactoryDocumentEcheance() *IBODocumentEcheance3*	Echéances.
		FactoryDocumentLigne() *IBODocumentLigne3*	Lignes de documents.
		FactoryDocumentPart() *IBODocumentPart3*	Lignes de documents faisant référence à un partenaire commercial
IBODocumentAchatLigne3	Ligne de document des achats	**FactoryDocumentLigne()** *IBODocumentLigne3*	Ligne de document.
IBODocumentAcompte3 *IBODocument3* *IBODocumentAchat3* *IBODocumentPart3* *IBODocumentVente3* FactoryDocumentAcompte()	Acompte	-	-

Interface objet *Interface objet maître* Propriété factory objet	**Description objet**	**Propriété factory sous-objet** *Interface sous-objet*	**Description sous-objet**
IBODocumentEcheance3 *IBODocument3* *IBODocumentAchat3* *IBODocumentPart3* *IBODocumentVente3* FactoryDocumentEcheance()	Echéance de règlement	-	-
IBODocumentInterne3 *BSCIALApplication100c* FactoryDocumentInterne()	Document interne.	-	-
IBODocumentInterneLigne3	Ligne de document interne	**FactoryDocumentLigne()** *IBODocumentLigne3*	Ligne de document
IBODocumentLigne3 *IBODocument3* FactoryDocumentLigne()	Ligne de document	**FactoryInfoComplement()** *IBOInfoComplementDocligne*	Informations complémentaires ligne
IBODocumentLigneEmplacement *IBODocumentLigne3* FactoryDocumentLigneEmplacement()	Emplacements associés à la ligne	**-**	-
IBODocumentLigneLienCM *IBODocumentLigne3* FactoryDocumentLigneLienCM()	Liens contremarque associés à la ligne	**-**	-
IBODocumentPart3	Document faisant référence à un partenaire commercial (vente, achat et interne)	**FactoryDocument()** *IBODocument3*	Document
		FactoryDocumentLigne() *IBODocumentLigne3*	Ligne de document
IBODocumentPartLigne3	Ligne de document faisant référence à un partenaire commercial (vente, achat et interne)	**FactoryDocumentLigne()** *IBODocumentLigne3*	Ligne de document

Interface objet *Interface objet maître* Propriété factory objet	**Description objet**	**Propriété factory sous-objet** *Interface sous-objet*	**Description sous-objet**
IBODocumentStock3 *BSCIALApplication100c* FactoryDocumentStock()	Document des stocks	**FactoryDocument()** *IBODocument3*	Document
		FactoryDocumentLigne() *IBODocumentLigne3*	Ligne de document
IBODocumentStockLigne3	Ligne de document des stocks	**FactoryDocumentLigne()** *IBODocumentLigne3*	Ligne de document
IBODocumentVente3 *BSCIALApplication100c* FactoryDocumentVente() Document des ventes		**FactoryDocument()** *IBODocument3*	Document
		FactoryDocumentAcompte() *IBODocumentAcompte3*	Acompte
		FactoryDocumentEcheance() *IBODocumentEcheance3*	Echeance de document
		FactoryDocumentLigne() *IBODocumentLigne3*	Ligne de document
		FactoryDocumentPart() *IBODocumentPart3*	Document faisant référence à un partenaire commercial
		FactoryDocumentVenteLigne() *IBODocumentVenteLigne3*	Ligne de document de vente
		FactoryInfoComplement() *IBOInfoComplementEntete*	Informations complémentaires Document
IBODocumentVenteLigne3	Ligne de document des ventes	**FactoryDocumentLigne()** *IBODocumentLigne3*	Ligne de document
IBOFamille3 *BSCIALApplication100c* FactoryFamille()		**FactoryFamilleParamCompta()** *IBOFamilleParamCompta3*	Paramètrage comptable de la famille

Interface objet *Interface objet maître* Propriété factory objet	**Description objet**	**Propriété factory sous-objet** *Interface sous-objet*	**Description sous-objet**
Famille d'article		**FactoryFamilleTarifCategorie()** *IBOFamilleTarifCategorie*	Tarif famille par catégorie
		FactoryFamilleTarifClient() *IBOFamilleTarifClient*	Tarif famille client
		FactoryFamilleTarifFournisseur() *IBOFamilleTarifFournisseur*	Tarif famille fournisseur
IBOFamilleParamCompta3 *IBOFamille3* FactoryFamilleParamCompta()	Paramètres comptables de la famille	-	-
IBOFamilleTarifCategorie *IBOFamille3* FactoryFamilleTarifCategorie()	Tarif famille par catégorie	-	-
IBOFamilleTarifClient *IBOFamille3* FactoryFamilleTarifClient()	Tarif famille client	-	-
IBOFamilleTarifFournisseur *IBOFamille3* FactoryFamilleTarifFournisseur()	Tarif famille fournisseur	-	-
IBOGlossaire2 *BSCIALApplication100c* FactoryGlossaire()	Glossaire	-	-
IBOModele2 *BSCIALApplication100c* FactoryModele()	Modèle d'enregistrement	-	-

IBOAgenda

Evénements agenda.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	AG_Comment() As String	Commentaires.
Lecture / Ecriture	AG_Confirme() As Boolean	Confirmation de l'événement.
Lecture / Ecriture	AG_Continue() As Boolean	Utilisation continue.
Lecture / Ecriture	AG_Ignorer() As Boolean	Evénement ignoré.
Lecture / Ecriture	AG_Period() As IDateTimePeriod	Période.
Lecture / Ecriture	AG_Veille() As Boolean	Evénement en veille.
Lecture / Ecriture	Agenda() As IBPAgenda	Agenda (uniquement pour les agendas de type Intéressé).
Lecture / Ecriture	AgendaEnum() As String	Enuméré d'agenda (uniquement pour les agendas de type Intéressé).
Lecture / Ecriture	AgendaType() As AgendaType	Type d'agenda (cf. énumérateur AgendaType).
Lecture seule	Article() As IBOArticle3	Article.
Lecture seule	Client() As IBOClient3	Client.
Lecture / Ecriture	Collaborateur() As IBOCollaborateur	Collaborateur.
Lecture / Ecriture	Depot() As IBODepot3	Dépôt.
Lecture / Ecriture	FactoryAgenda() As IBOAgendaFactory	Fabrique un objet événement agenda.
Lecture / Ecriture	Fournisseur() As IBOFournisseur3	Fournisseur.
Lecture / Ecriture	Ligne() As IBODocumentPartLigne3	Ligne de document (uniquement pour les agendas de type Document).
Lecture / Ecriture	Ressource() As IBIRessource	Ressource.
Lecture / Ecriture	TypeInteresse() As AgendaTypeInteresse	Type d'événement (si AgendaType est différent de AgendaInteresse alors TypeInteresse est forcément AgendaTypeInteresseRessource). Cf. énumérateur AgendaTypeInteresse.

IBOArticle3

Article.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	AR_CodeBarre() As String	Code barre.
Lecture / Ecriture	AR_EdiCode() As String	Code EDI.
Lecture / Ecriture	AR_CodeFiscal() As String	Code fiscal.
Lecture / Ecriture	AR_Coef() As Double	Coefficient.
Lecture / Ecriture	AR_Contremarque() As Boolean	Article géré en contremarque.
Lecture / Ecriture	AR_CoutStd() As Double	Coût standard.
Lecture / Ecriture	AR_Criticite() As FamilleCriticiteType	Niveau de criticité.
Lecture / Ecriture	AR_Cycle() As CycleType	Cycle de vie.
Lecture seule	AR_DateCreation() As Date	Date de création.
Lecture seule	AR_DateModif() As Date	Date de modification.
Lecture / Ecriture	AR_Delai() As Short	Délai de livraison (en jours).
Lecture / Ecriture	AR_DelaiFabrication() As Short	Délai de fabrication (en jours).
Lecture / Ecriture	AR_DelaiPeremption() As Short	Délai de péremption (en jours).
Lecture / Ecriture	AR_DelaiSecurite() As Short	Délai de sécurité (en jours).
Lecture / Ecriture	AR_Design() As String	Désignation.
Lecture / Ecriture	AR_Escompte() As Boolean	Article non soumis à escompte (True) ou soumis à l'escompte (False).
Lecture / Ecriture	AR_FactForfait() As Boolean	Facturation forfaitaire.
Lecture / Ecriture	AR_FactPoids() As Boolean	Facturation sur le poids.
Lecture / Ecriture	AR_Fictif() As Boolean	Gérer en tant que fictif.
Lecture / Ecriture	AR_Frais() As Ifrais	Frais (de stockage, de transport, etc…).
Lecture / Ecriture	AR_Garantie() As Short	Durée de la garantie (en mois).
Lecture / Ecriture	AR_HorsStat() As Boolean	Pris en compte dans les statistiques (True) ou non pris en compte dans les statistiques (False).
Lecture / Ecriture	AR_Langue1() As String	Langue 1.
Lecture / Ecriture	AR_Langue2() As String	Langue 2.
Lecture / Ecriture	AR_Nature() As FamNatureType	Type de nature de l'article (cf. énumérateur FamNatureType).
Lecture / Ecriture	AR_NbColis() As Short	Nombre de colis.
Lecture / Ecriture	AR_Nomencl() As NomenclatureType	Nomenclature article (cf. énumérateur NomenclatureType).

Accès	Syntaxe	Description
Lecture / Ecriture	AR_NotImp() As Boolean	Non impression document (True) ou impression document (False).
Lecture / Ecriture	AR_Pays() As String	Pays.
Lecture / Ecriture	AR_Photo() As String	Photo (chemin).
Lecture / Ecriture	AR_PoidsBrut() As Double	Poids brut.
Lecture / Ecriture	AR_PoidsNet() As Double	Poids net.
Lecture / Ecriture	AR_Prevision() As Boolean	Nomenclature en réappro/prévision
Lecture / Ecriture	AR_PrixAchat() As Double	Prix d'achat.
Lecture / Ecriture	AR_PrixTTC() As Boolean	Prix TTC (True) ou Prix HT (False).
Lecture / Ecriture	AR_PrixVen() As Double	Prix de vente.
Lecture / Ecriture	AR_Publie() As Boolean	Article publié sur le site marchand.
Lecture / Ecriture	AR_PUNet() As Double	Dernier prix d'achat.
Lecture / Ecriture	AR_QteComp() As Double	Composition de la nomenclature pour n composés fabriqués.
Lecture / Ecriture	AR_QteOperatoire() As Double	Quantité économique.
Lecture / Ecriture	AR_Raccourci() As String	Raccourci.
Lecture / Ecriture	AR_Ref() As String	Référence article.
Lecture / Ecriture	AR_Sommeil() As Boolean	Article en sommeil.
Lecture / Ecriture	AR_SousTraitance() As Boolean	Réserver en sous-traitance.
Lecture / Ecriture	AR_Stat(ByVal *sVal* As Short) As String	Enuméré statistique correspondant au Nieme champs statistique passé en paramètre.
Lecture / Ecriture	AR_SuiviStock() As SuiviStockType	Mode de suivi de stock (cf. énumérateur SuiviStockType).
Lecture / Ecriture	AR_Type() As ArticleType	Type d'article (cf. énumérateur ArticleType).
Lecture / Ecriture	AR_TypeLancement() As LancementType	Type de lancement (cf. énumérateur LancementType).
Lecture / Ecriture	AR_UnitePoids() As UnitePoidsType	Unité de poids (cf. énumérateur UnitéPoidsType).
Lecture / Ecriture	AR_VteDebit() As Boolean	Vente au débit.
Lecture / Ecriture	ArticleDepotPrincipal() As IBOArticleDepot3	Dépôt principal.
Lecture / Ecriture	ArticleSubstitution() As IBOArticle3	Article de substitution.
Lecture / Ecriture	Catalogue() As IBPProduit2	Catalogue.
Lecture / Ecriture	Conditionnement() AsIBPConditionnement	Conditionnement.
Lecture / Ecriture	ConditionnementAffichage() As IBOArticleCond3	Article de conditionnement affiché sur l'onglet stock

Accès	Syntaxe	Description
Lecture/ Ecriture	ConditionnementPrincipal() As IBOArticleCond3	Article principal pour le conditionnement
Lecture seule	FactoryArticle() As IBOArticleFactory3	Fabrique un objet article.
Lecture seule	FactoryArticleCond() As IBOArticleCondFactory	Fabrique un objet conditionnement article.
Lecture seule	FactoryArticleDepot() As IBOArticleDepotFactory	Fabrique un objet dépôt article.
Lecture seule	FactoryArticleGammeEnum1() As IBITypeObjectFactory	Fabrique un objet article énuméré de gamme 1.
Lecture seule	FactoryArticleGammeEnum2() As IBITypeObjectFactory	Fabrique un objet article énuméré de gamme 2.
Lecture seule	FactoryArticleGlossaire() As IBOArticleGlossaireFactory2	Fabrique un objet glossaire article.
Lecture seule	FactoryArticleMedia() As IBITypeObjectFactory	Fabrique un objet média article (photo ou document).
Lecture seule	FactoryArticleNomenclature() As IBITypeObjectFactory	Fabrique un objet nomenclature article.
Lecture seule	FactoryArticleParamCompta() As IBOArticleParamComptaFactory3	Fabrique un objet paramètres comptables article.
Lecture seule	FactoryArticleRessource() As IBOArticleRessourceFactory	Fabrique un objet ressource article.
Lecture seule	FactoryArticleTarifCategorie() As IBITypeObjectFactory	Fabrique un objet tarif par catégorie article.
Lecture seule	FactoryArticleTarifClient() As IBITypeObjectFactory	Fabrique un objet tarif client article.
Lecture seule	FactoryArticleTarifFournisseur() As IBITypeObjectFactory	Fabrique un objet tarif fournisseur article.
Lecture / Ecriture	Famille() As IBOFamille3	Famille.
Lecture / Ecriture	FournisseurPrincipal() As IBOArticleTarifFournisseur3	Fournisseur principal.
Lecture / Ecriture	Gamme1() As IBPGamme	Gamme 1.
Lecture / Ecriture	Gamme2() As IBPGamme	Gamme 2.
Lecture seule	InfoLibre() As IBIValues	Valeur information libre.
Lecture / Ecriture	ModeleAchat() As IBOModele2	Modèle d'enregistrement documents des achats.
Lecture / Ecriture	ModeleComptoir() As IBOModele2	Modèle d'enregistrement vente comptoir.
Lecture / Ecriture	ModeleInterne() As IBOModele2	Modèle d'enregistrement documents internes.
Lecture / Ecriture	ModeleStock() As IBOModele2	Modèle d'enregistrement documents des stocks.

Accès	Syntaxe	Description
Lecture / Ecriture	ModeleTousDomaine() As IBOModele2	Modèle d'enregistrement tous domaines.
Lecture / Ecriture	ModeleVente() As IBOModele2	Modèle d'enregistrement documents des stocks.
Lecture / Ecriture	RessourceDefault() As IBIRessource	Ressource par défaut.
Lecture / Ecriture	Unite() As IBPUnite	Unité.

Méthodes

Syntaxe	Description
CA_HTBrut(ByVal *pTiers* As IBICollection, ByVal *dDebut* As Date, ByVal *dFin* As Date, ByVal *TypeDe* As DocumentType, ByVal *TypeA* As DocumentType) As Double	Chiffre d'affaires HT pour la période indiquée, pour un ensemble de tiers et pour un type de document passés en paramètre (cf. énumérateur DocumentType).
CA_HTNet(ByVal *pTiers* As IBICollection, ByVal *dDebut* As Date, ByVal *dFin* As Date, ByVal *TypeDe* As DocumentType, ByVal *TypeA* As DocumentType) As Double	Chiffre d'affaires HT net pour la période indiquée, pour un ensemble de tiers et pour un type de document passés en paramètre (cf. énumérateur DocumentType).
CA_TTC(ByVal *pTiers* As IBICollection, ByVal *dDebut* As Date, ByVal *dFin* As Date, ByVal *TypeDe* As DocumentType, ByVal *TypeA* As DocumentType) As Double	Chiffre d'affaires TTC pour la période indiquée, pour un ensemble de tiers et pour un type de document passés en paramètre (cf. énumérateur DocumentType).
StockATerme() As Double	Stock à terme.
StockATermeDoubleGamme(ByVal *pEnum1* As IBOArticleGammeEnum3, ByVal *pEnum2* As IBOArticleGammeEnum3) As Double	Stock à terme pour un article à double gamme (énumérés de gamme 1 et 2 passés en paramètre).
StockATermeMonoGamme(ByVal *pEnum* As IBOArticleGammeEnum3) As Double	Stock à terme pour un article à simple gamme (énuméré de gamme passé en paramètre).
StockCommande() As Double	Stock commandé.
StockCommandeContremarque() As Double	Stock commandé contremarque.
StockCommandeContremarqueDoubleGamme(ByVal *pEnum1* As IBOArticleGammeEnum3, ByVal *pEnum2* As IBOArticleGammeEnum3) As Double	Stock commandé contremarque pour un article à double gamme (énumérés de gamme 1 et 2 passés en paramètre).
StockCommandeContremarqueMonoGamme(ByVal *pEnum* As IBOArticleGammeEnum3) As Double	Stock commandé contremarque pour un article à simple gamme (énuméré de gamme passé en paramètre).
StockCommandeDoubleGamme(ByVal *pEnum1* As IBOArticleGammeEnum3, ByVal *pEnum2* As IBOArticleGammeEnum3) As Double	Stock commandé pour un article à double gamme (énumérés de gamme 1 et 2 passés en paramètre).

Syntaxe	Description
StockCommandeMonoGamme(ByVal *pEnum* As IBOArticleGammeEnum3) As Double	Stock commandé pour un article à simple gamme (énuméré de gamme passé en paramètre).
StockDispo() As Double	Stock disponible.
StockDispoDoubleGamme(ByVal *pEnum1* As IBOArticleGammeEnum3, ByVal *pEnum2* As IBOArticleGammeEnum3) As Double	Stock disponible pour un article à double gamme (énumérés de gamme 1 et 2 passés en paramètre).
StockDispoMonoGamme(ByVal *pEnum* As IBOArticleGammeEnum3) As Double	Stock disponible pour un article à simple gamme (énuméré de gamme passé en paramètre).
StockPrepare() As Double	Stock préparé.
StockPrepareDoubleGamme(ByVal *pEnum1* As IBOArticleGammeEnum3, ByVal *pEnum2* As IBOArticleGammeEnum3) As Double	Stock préparé pour un article à double gamme (énumérés de gamme 1 et 2 passés en paramètre).
StockPrepareMonoGamme(ByVal *pEnum* As IBOArticleGammeEnum3) As Double	Stock préparé pour un article à simple gamme (énuméré de gamme passé en paramètre).
StockQte(ByVal *DateStock* As Date) As Double	Quantité en stock à la date indiquée (passée en paramètre).
StockQteDoubleGamme(ByVal *DateStock* As Date, ByVal *pEnum1* As IBOArticleGammeEnum3, ByVal *pEnum2* As IBOArticleGammeEnum3) As Double	Quantité en stock pour un article à double gamme à la date indiquée (date et énumérés de gamme 1 et 2 passés en paramètre).
StockQteMonoGamme(ByVal *DateStock* As Date, ByVal *pEnum* As IBOArticleGammeEnum3) As Double	Quantité en stock pour un article à simple gamme à la date indiquée (date et énuméré de gamme passé en paramètre).
StockReel() As Double	Stock réel.
StockReelDoubleGamme(ByVal *pEnum1* As IBOArticleGammeEnum3, ByVal *pEnum2* As IBOArticleGammeEnum3) As Double	Stock réel pour un article à double gamme (énumérés de gamme 1 et 2 passés en paramètre).
StockReelMonoGamme(ByVal *pEnum* As IBOArticleGammeEnum3) As Double	Stock réel pour un article à simple gamme (énuméré de gamme passé en paramètre).
StockReserve() As Double	Stock réservé.
StockReserveContremarque() As Double	Stock réservé contremarque.
StockReserveContremarqueDoubleGamme(ByVal *pEnum1* As IBOArticleGammeEnum3, ByVal *pEnum2* As IBOArticleGammeEnum3) As Double	Stock réservé contremarque pour un article à double gamme (énumérés de gamme 1 et 2 passés en paramètre).
StockReserveContremarqueMonoGamme(ByVal *pEnum* As IBOArticleGammeEnum3) As Double	Stock réservé contremarque pour un article à simple gamme (énuméré de gamme passé en paramètre).
StockReserveDoubleGamme(ByVal *pEnum1* As IBOArticleGammeEnum3, ByVal *pEnum2* As IBOArticleGammeEnum3) As Double	Stock réservé pour un article à double gamme (énumérés de gamme 1 et 2 passés en paramètre).

Syntaxe	Description
StockReserveMonoGamme(ByVal *pEnum* As IBOArticleGammeEnum3) As Double	Stock réservé pour un article à simple gamme (énuméré de gamme passé en paramètre).
StockValeur(ByVal *DateStock* As Date) As Double	Valeur du stock à la date indiquée (passée en paramètre).
StockValeurDoubleGamme(ByVal *DateStock* As Date, ByVal *pEnum1* As IBOArticleGammeEnum3, ByVal *pEnum2* As IBOArticleGammeEnum3) As Double	Valeur du stock pour un article à double gamme à la date indiquée (date et énumérés de gamme 1 et 2 passés en paramètre).
StockValeurMonoGamme(ByVal *DateStock* As Date, ByVal *pEnum* As IBOArticleGammeEnum3) As Double	Valeur du stock pour un article à simple gamme à la date indiquée (date et énuméré de gamme passé en paramètre).
TarifAchat() As ITarifAchat2	Prix d'achat.
TarifAchatDoubleGamme(ByVal *pEnum1* As IBOArticleGammeEnum3, ByVal *pEnum2* As IBOArticleGammeEnum3) As ITarifAchat	Prix d'achat pour un article à double gamme. (énumérés de gamme 1 et 2 passés en paramètres).
TarifAchatMonoGamme(ByVal *pEnum* As IBOArticleGammeEnum3) As ITarifAchat2	Prix d'achat pour un article à simple gamme. (énumérés de gamme 1 passé en paramètres).
TarifAchatTiers(ByVal *pFournisseur* As IBOFournisseur3, ByVal *dQtes* As Double) As ITarifAchat2	Prix d'achat unitaire (fournisseur et quantités achetées passés en paramètres).
TarifAchatTiersDoubleGamme(ByVal *pFournisseur* As IBOFournisseur3, ByVal *pEnum1* As IBOArticleGammeEnum3, ByVal *pEnum2* As IBOArticleGammeEnum3, ByVal *dQtes* As Double) As ITarifAchat2	Prix d'achat unitaire pour un article à double gamme (fournisseur, énumérés de gamme 1 et 2 et quantités achetées passés en paramètres).
TarifAchatTiersMonoGamme(ByVal *pFournisseur* As IBOFournisseur3, ByVal *pEnum* As IBOArticleGammeEnum2, ByVal *dQtes* As Double) As ITarifAchat2	Prix d'achat unitaire pour un article à simple gamme. (fournisseur et énumérés de gamme 1 et quantités achetées passés en paramètres).
TarifVente() As ITarifVente2	Prix de vente.
TarifVenteCategorie(ByVal *pCategorieTarif* As IBPCategorieTarif, ByVal *dQtes* As Double) As ItarifVente2	Prix de vente unitaire pour une catégorie tarifaire (catégorie tarifaire et quantités vendues passés en paramètres).
TarifVenteCategorieCond(ByVal *pCategorieTarif* As IBPCategorieTarif, ByVal *pConditionnement* As IBOArticleCond2, ByVal *dQtes* As Double) As ITarifVente2	Prix de vente unitaire pour une catégorie tarifaire et un conditionnement donnés (catégorie tarifaire, conditionnement et quantités vendues passés en paramètres).

Syntaxe	Description
TarifVenteCategorieDoubleGamme(ByVal *pCategorieTarif* As IBPCategorieTarif, ByVal *pEnum1* As IBOArticleGammeEnum3, ByVal *pEnum2* As IBOArticleGammeEnum3, ByVal *dQtes* As Double) As ITarifVente2	Prix de vente unitaire pour un catégorie tarifaire pour un article à double gamme (catégorie tarifaire et énumérés de gamme 1 et 2 passés en paramètres).
TarifVenteCategorieMonoGamme(ByVal *pCategorieTarif* As IBPCategorieTarif, ByVal *pEnum* As IBOArticleGammeEnum3, ByVal *dQtes* As Double) As ITarifVente2	Prix de vente unitaire pour un catégorie tarifaire pour un article à simple gamme (catégorie tarifaire et énumérés de gamme 1 passés en paramètres).
TarifVenteTiers(ByVal *pClient* As IBOClient2, ByVal *dQtes* As Double) As ITarifVente	Tarif client unitaire (client et quantités vendues passés en paramètres).
TarifVenteTiersCond(ByVal *pClient* As IBOClient3, ByVal *pConditionnement* As IBOArticleCond3, ByVal *dQtes* As Double) As ITarifVente2	Tarif client unitaire pour un conditionnement donné (catégorie tarifaire, tiers et quantités vendues passés en paramètres).
TarifVenteTiersDoubleGamme(ByVal *pClient* As IBOClient3, ByVal *pEnum1* As IBOArticleGammeEnum3, ByVal *pEnum2* As IBOArticleGammeEnum3, ByVal *dQtes* As Double) As ITarifVente2	Tarif client unitaire pour un article à double gamme (tiers et énumérés de gamme 1 et 2 passés en paramètres).
TarifVenteTiersMonoGamme(ByVal *pClient* As IBOClient3, ByVal *pEnum* As IBOArticleGammeEnum3, ByVal *dQtes* As Double) As ITarifVente2	Tarif client unitaire pour un article à simple gamme (tiers et énumérés de gamme 1 et 2 passés en paramètres).
TauxRemiseMoyen(ByVal *pTiers* As IBICollection, ByVal *dDebut* As Date, ByVal *dFin* As Date, ByVal *TypeDe* As DocumentType, ByVal *TypeA* As DocumentType) As Double	Taux de remise moyen pour un ensemble de tiers sur une période donnée et pour un type de document particulier passés en paramètres (cf. énumérateur DocumentType).
TotalCoutRevient(ByVal *pTiers* As IBICollection, ByVal *dDebut* As Date, ByVal *dFin* As Date, ByVal *TypeDe* As DocumentType, ByVal *TypeA* As DocumentType) As Double	Total coût de revient pour un ensemble de tiers sur une période donnée et pour un ensemble de documents passés en paramètres (cf. énumérateur DocumentType).
TotalMarge(ByVal *pTiers* As IBICollection, ByVal *dDebut* As Date, ByVal *dFin* As Date, ByVal *TypeDe* As DocumentType, ByVal *TypeA* As DocumentType) As Double	Total marge pour un ensemble de tiers sur une période donnée et pour un ensemble de documents passés en paramètres (cf. énumérateur DocumentType).
TotalPoidsBrut(ByVal *pTiers* As IBICollection, ByVal *dDebut* As Date, ByVal *dFin* As Date, ByVal *TypeDe* As DocumentType, ByVal *TypeA* As DocumentType) As Double	Total poids brut pour un ensemble de tiers sur une période donnée et pour un ensemble de documents passés en paramètres (cf. énumérateur DocumentType).
TotalPoidsNet(ByVal *pTiers* As IBICollection, ByVal *dDebut* As Date, ByVal *dFin* As Date, ByVal *TypeDe* As DocumentType, ByVal *TypeA* As DocumentType) As Double	Total poids net pour un ensemble de tiers sur une période donnée et pour un ensemble de documents passés en paramètres (cf. énumérateur DocumentType).
TotalQtes(ByVal *pTiers* As IBICollection, ByVal *dDebut* As Date, ByVal *dFin* As Date, ByVal *TypeDe* As	Total quantités pour un ensemble de tiers sur une période donnée et pour un

Syntaxe	Description
DocumentType, ByVal *TypeA* As DocumentType) As Double	ensemble de documents passés en paramètres (cf. énumérateur DocumentType).
TotalQtesColisees(ByVal *pTiers* As IBICollection, ByVal *dDebut* As Date, ByVal *dFin* As Date, ByVal *TypeDe* As DocumentType, ByVal *TypeA* As DocumentType) As Double	Total quantités colisées pour un ensemble de tiers sur une période donnée et pour un ensemble de documents passés en paramètres (cf. énumérateur DocumentType).

Traitements et initialisations par défaut

Méthode SetDefault()

Si l'objet est non persistant, l'appel de la méthode *SetDefault()* initialise les propriétés suivantes :

Propriété	Valeur	Description
AR_CodeFiscal	AR_CodeFiscal = Famille.FA_CodeFiscal	Affecte le Code fiscal de la famille à l'article.
AR_Coef	AR_Coef = Famille.FA_Coef	Affecte le Coefficient de la famille à l'article.
AR_Contremarque	AR_Contremarque = Famille.FA_Contremarque	Affecte l'option Article en contremarque de la famille à l'article.
AR_Delai	AR_Delai = Famille.FA_Delai	Affecte le Délai livraison de la famille à l'article.
AR_Design	Si AR_Design = "" Alors AR_Design = AR_Ref	Si la désignation article n'est pas renseignée elle reprend la référence article.
AR_Escompte	AR_Escompte = Famille.FA_Escompte	Affecte l'option Non soumis à l'escompte de la famille à l'article.
AR_FactForfait	AR_FactForfait = Famille.FA_FactForfait	Affecte l'option Facturation forfaitaire de la famille à l'article.
AR_FactPoids	AR_FactPoids = Famille.FA_FactPoids	Affecte l'option Facturation / poids net de la famille à l'article.
AR_Frais	AR_Frais = Famille.Frais	Affecte les différents coûts (de stockage, de transport, etc…) de la famille à l'article.
AR_Garantie	AR_Garantie = Famille.FA_Garantie	Affecte la Garantie de la famille à l'article.
AR_HorsStat	AR_HorsStat = Famille.FA_HorsStat	Affecte l'option Hors statistique de la famille à l'article.
AR_NotImp	AR_NotImp = Famille.FA_NotImp	Affecte l'option Non impression document de la famille à l'article.
AR_Pays	AR_Pays = Famille.FA_Pays	Affecte le Pays de la famille à l'article.
AR_Publie	AR_Publie = Famille.FA_Publie	Affecte l'option Publié sur le site marchand de la famille à l'article.

Propriété	Valeur	Description
AR_SuiviStock	AR_SuiviStock = Famille.FA_SuiviStock	Affecte le Suivi de stock de la famille à l'article.
AR_Type	AR_Type = ArticleType.ArticleTypeStandard	Affecte le type standard.
AR_UnitePoids	AR_UnitePoids = Famille.FA_UnitePoids	Affecte l'Unité de poids de la famille à l'article.
AR_VteDebit	AR_VteDebit = Famille.FA_VteDebit	Affecte l'option Vente au débit de la famille à l'article.
ModeleAchat	ModeleAchat = Modèle achat de la famille	Affecte le modèle d'enregistrement achat de la famille à l'article
ModeleComptoir	ModeleComptoir = Modèle vente comptoir de la famille	Affecte le modèle d'enregistrement comptoir de la famille à l'article
ModeleInterne	ModeleInterne = Modèle interne de la famille	Affecte le modèle d'enregistrement interne de la famille à l'article
ModeleStock	ModeleStock = Modèle stock de la famille	Affecte le modèle d'enregistrement stock de la famille à l'article
ModeleTousDomaine	ModeleTousDomaine = Modèle tous domaine de la famille	Affecte le modèle d'enregistrement tous domaine de la famille à l'article
ModeleVente	ModèleVente = Modèle vente de la famille	Affecte le modèle d'enregistrement vente de la famille à l'article
Unite	Unite = Famille.Unite	Affecte l'Unité de vente de la famille à l'article.

Si l'objet est persistant, l'appel de la méthode *SetDefault()* initialise les propriétés suivantes :

Propriété	Valeur	Description
AR_Design	Si AR_Design = "" Alors AR_Design = AR_Ref	Si la désignation article n'est pas renseignée elle reprend la référence article.

Méthode Write()

La méthode *Write()* enregistre l'objet dans la base de données après avoir effectué les traitements suivants :

- Si un modèle d'enregistrement est associé à l'article (propriétés ModeleAchat, ModeleComptoir, ModeleInterne, ModeleStock, ModeleTousDomaine, ModeleVente), un enregistrement correspondant à la liaison entre les tables F_MODELE et F_ARTICLE est créé dans la table F_ARTMODELE.
- Si des énumérés statistique ont été créé dans l'article, ils sont ajoutés dans la table F_ENUMSTATART.
- Copie du fichier multimédia (propriété AR_Photo) dans le répertoire Multimédia situé au même emplacement que le fichier .MAE. Le fichier porte le nom Article.AR_Ref + extension.

Méthode WriteDefault()

La méthode *WriteDefault()* enregistre l'objet dans la base de données après avoir effectué les traitements suivants :

- Si un modèle d'enregistrement est associé à l'article (propriétés ModeleAchat, ModeleComptoir, ModeleInterne, ModeleStock, ModeleTousDomaine, ModeleVente), un enregistrement correspondant à la liaison entre les tables F_MODELE et F_ARTICLE est créé dans la table F_ARTMODELE.
- Si des énumérés statistique ont été créés dans l'article, ils sont ajoutés dans la table F_ENUMSTATART.
- Héritage des tarifs de la famille (fournisseur, client et par catégorie tarifaire).
- Création des énumérés de conditionnement par défaut si l'article est géré par conditionnement.
- Création des énumérés de gamme par défaut si l'article est géré par gamme.
- Copie du fichier multimédia (propriété AR_Photo) dans le répertoire Multimédia situé au même emplacement que le fichier .MAE. Le fichier porte le nom Article.AR_Ref + extension.

IBOArticleCond3

Conditionnement article.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Article() As IBOArticle3	Article auquel est associé le conditionnement.
Lecture/ Ecriture	CO_CodeBarre() As String	Code barre de l'article conditionnement.
Lecture / Ecriture	CO_EdiCode() As String	Code EDI conditionnement article.
Lecture / Ecriture	CO_Ref() As String	Référence conditionnement.
Lecture / Ecriture	EC_Enumere() As String	Enuméré de conditionnement.
Lecture / Ecriture	EC_Quantite() As Double	Quantité conditionnement.
Lecture seule	FactoryArticleCond() As IBITypeObjectFactory	Fabrique un objet conditionnement article.

IBOArticleDepot3

Dépôt de stockage article.

Interface héritée

Syntaxe	Description
IBIArticleStock3	Cf. Interface IBIArticleStock2 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	AS_Emplacement() As String	Emplacement de stockage.
Lecture / Ecriture	AS_QteMaxi() As Double	Quantité maxi.
Lecture / Ecriture	AS_QteMini() As Double	Quantité mini.
Lecture seule	Depot() As IBODepot3	Dépôt de stockage de l'article.
Lecture seule	FactoryArticleDepot() As IBITypeObjectFactory	Fabrique un objet dépôt de stockage article.
Lecture seule	FactoryArticleDepotGamme() As IBOArticleDepotGammeFactory	Fabrique un objet dépôt de stockage article à gamme.
Lecture seule	FactoryArticleDepotLot() As IBOArticleDepotLotFactory	Fabrique un objet dépôt lot.

IBOArticleDepotLot

Lot article par dépôt.

Interface héritée

Syntaxe	Description
IBILot	Cf. Interface IBILot pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Complement() As String	Complément.
Lecture / Ecriture	DateFabrication() As Date	Date de fabrication.
Lecture / Ecriture	DatePeremption() As Date	Date de péremption.
Lecture seule	FactoryArticleDepotLot() As IBOArticleDepotLotFactory	Fabrique un objet lot article par dépôt.
Lecture seule	InfoLibre() As IBIValues	Informations libres.
Lecture / Ecriture	NoSerie() As String	Numéro de série/lot.

Accès	Syntaxe	Description
Lecture seule	QteReserved() As Double	Quantité réservée.

IBOArticleDepotGamme3

Dépôt de stockage article à gamme.

Interface héritée

Syntaxe	Description
IBIArticleStock3	Cf. Interface IBIArticleStock2 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	ArticleDepot() As IBOArticleDepot3	Dépôt de stockage de l'article à gamme.
Lecture / Ecriture	ArticleGammeEnum1() As IBOArticleGammeEnum3	Intitulé de l'énuméré de gamme 1.
Lecture / Ecriture	ArticleGammeEnum2() As IBOArticleGammeEnum3	Intitulé de l'énuméré de gamme 2.
Lecture seule	FactoryArticleDepotGamme() As IBITypeObjectFactory	Fabrique un objet dépôt article à gamme.
Lecture / Ecriture	GS_Emplacement() As String	Emplacement de stockage.
Lecture / Ecriture	GS_QteMaxi() As Double	Quantité maximale.
Lecture / Ecriture	GS_QteMini() As Double	Quantité minimale.

IBOArticleGammeEnum3

Enuméré de gamme article.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Article() As IBOArticle3	Article auquel est associé l'énuméré de gamme article.

Accès	Syntaxe	Description
Lecture / Ecriture	EG_Enumere() As String	Intitulé de l'énuméré de gamme article.
Lecture seule	FactoryArticleGammeEnum() As IBITypeObjectFactory	Fabrique un objet énuméré de gamme article.
Lecture seule	FactoryArticleGammeEnumRef() As IBITypeObjectFactory	Fabrique un objet référence énuméré de gamme article.
Lecture seule	Compteur () As int	Compteur de l'énuméré de gamme

Traitements et initialisations par défaut

Méthode WriteDefault()

La méthode *WriteDefault()* créé les enregistrements "énumérés de gamme" (table F_ENUMFAMME) et "stock gamme" (table F_GAMSTOCK) si l'article est suivi en stock.

IBOArticleGammeEnumRef3

Référence énuméré de gamme article.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	AE_CodeBarre() As String	Code barre énuméré de gamme.
Lecture / Ecriture	AE_PrixAch() As Double	Prix d'achat énuméré de gamme.
Lecture / Ecriture	AE_Ref() As String	Référence énuméré de gamme.
Lecture / Ecriture	AE_Sommeil() As Boolean	Référence gamme en sommeil.
Lecture / Ecriture	AR_CoutStd() As Double	Coût standard.
Lecture / Ecriture	AR_PUNet() As Double	Prix unitaire net énuméré de gamme.
Lecture seule	Article() As IBOArticle3	Article auquel est associé la référence énuméré de gamme.
Lecture seule	ArticleGammeEnum1() As IBOArticleGammeEnum3	Enuméré de gamme 1.
Lecture seule	ArticleGammeEnum2() As IBOArticleGammeEnum3	Enuméré de gamme 2.
Lecture seule	FactoryArticleGammeEnumRef() As IBITypeObjectFactory	Fabrique un objet référence énuméré de gamme.

IBOArticleMedia3

Information multimédia de l'article (photo/document).

Interface héritée

Syntaxe	Description
IBIMedia	Cf. Interface IBIMedia pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Article() As IBOArticle3	Article auquel est associée l'information multimédia.
Lecture seule	FactoryArticleMedia() As IBITypeObjectFactory	Fabrique un objet correspondant à une information multimédia.

IBOArticleNomenclature3

Composant de nomenclature article.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Article() As IBOArticle3	Article auquel est associé le composant de la nomenclature.
Lecture / Ecriture	ArticleComposant() As IBOArticle3	Article composant de la nomenclature.
Lecture/ Ecriture	ArticleComposeGammeEnum1() As IBOArticleGammeEnum3	Enuméré de gamme 1 du composé.
Lecture/ Ecriture	ArticleComposeGammeEnum2() As IBOArticleGammeEnum3	Enuméré de gamme 2 du composé.
Lecture / Ecriture	ArticleGammeEnum1() As IBOArticleGammeEnum3	Enuméré de gamme 1 du composant.
Lecture / Ecriture	ArticleGammeEnum2() As IBOArticleGammeEnum3	Enuméré de gamme 2 du composant.
Lecture / Ecriture	Depot() As IBODepot3	Dépôt de stockage.
Lecture seule	FactoryArticleNomenclature() As IBITypeObjectFactory	Fabrique un objet composant de nomenclature article.

Accès	Syntaxe	Description
Lecture / Ecriture	NO_Commentaire() As String	Commentaire.
Lecture / Ecriture	NO_Operation() As String	Opération.
Lecture / Ecriture	NO_Qte() As Double	Quantité.
Lecture / Ecriture	NO_Repartition() As Double	Répartition (0 -100).
Lecture / Ecriture	NO_Type() As ComposantType	Type de composant (cf. énumérateur ComposantType).
Lecture / Ecriture	SousTraitance() As Boolean	Nomenclature en sous traitance.

IBOArticleParamCompta3

Paramétrage comptable de l'article.

Interface héritée

Syntaxe	Description
IBIParamCompta3	Cf. Interface IBIParamCompta3 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Article() As IBOArticle3	Article auquel est associé le paramétrage comptable.
Lecture seule	FactoryArticleParamCompta() As IBOArticleParamComptaFactory3	Fabrique un objet Paramétragre comptable article.

IBOArticleTarifCategorie3

Tarif par catégorie tarifaire article.

Interface héritée

Syntaxe	Description
IBIArticleTarifVente3	Cf. Interface IBIArticleTarifVente3 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	CategorieTarif() As IBPCategorieTarif	Catégorie tarifaire à laquelle est associé le tarif.

Accès	Syntaxe	Description
Lecture seule	FactoryArticleTarifCategorie() As IBITypeObjectFactory	Fabrique un objet Tarif par catégorie tarifaire article.

IBOArticleTarifClient3

Tarif client article.

Interface héritée

Syntaxe	Description
IBIArticleTarifVente3	Cf. Interface IBIArticleTarifVente3 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Client() As IBOClient3	Client auquel est associé le tarif client de l'article.
Lecture / Ecriture	Reference() As String	Référence du tarif client article.

IBOArticleTarifCondCategorie3

Tarif par conditionnement d'une catégorie tarifaire article.

Interface héritée

Syntaxe	Description
IBIArticleTarifCond3	Cf. Interface IBIArticleTarifCond3 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	CategorieTarif() As IBPCategorieTarif	Catégorie tarifaire à laquelle est associé le tarif par conditionnement de l'article.
Lecture seule	FactoryArticleTarifCondCategorie() As IBITypeObjectFactory	Fabrique un objet Tarif par conditionnement associé à la catégorie tarifaire de l'article.

IBOArticleTarifCondClient3

Tarif par conditionnement d'un client article.

Interface héritée

Syntaxe	Description
IBIArticleTarifCond3	Cf. Interface IBIArticleTarifCond3 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Client() As IBOClient3	Client auquel est associé le tarif par conditionnement.
Lecture seule	FactoryArticleTarifCondClient() As IBITypeObjectFactory	Fabrique un objet Tarif par conditionnement d'un client article.

IBOArticleTarifFournisseur3

Tarif fournisseur.

Interface héritée

Syntaxe	Description
IBIArticleTarif3	Cf. Interface IBIArticleTarif3 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	AF_CodeBarre() As String	Code barres.
Lecture / Ecriture	AF_Colisage() As Double	Colisage.
Lecture / Ecriture	AF_ConvDiv() As Double	Diviseur de conversion.
Lecture / Ecriture	AF_Conversion() As Double	Conversion de l'unité d'achat en unité de vente.
Lecture / Ecriture	AF_DelaiAppro() As Short	Délai d'approvisionnement (en jours).
Lecture / Ecriture	AF_Garantie() As Short	Garantie (en mois).
Lecture / Ecriture	AF_QteMini() As Double	Quantité
Lecture / Ecriture	Fournisseur() As IBOFournisseur3	Fournisseur auquel est associé le tarif fournisseur.
Lecture / Ecriture	Reference() As String	Référence.
Lecture / Ecriture	Unite() As IBPUnite	Unité d'achat.

IBOArticleTarifGammeCategorie3

Tarif pour une catégorie tarifaire d'un article à gamme.

Interface héritée

Syntaxe	Description
IBIArticleTarifGamme3	Cf. Interface IBIArticleTarifGamme3 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	CategorieTarif() As IBPCategorieTarif	Catégorie tarifaire à laquelle est associée le tarif.
Lecture seule	FactoryArticleTarifGammeCategorie() As IBITypeObjectFactory	Fabrique un objet Tarif pour une catégorie tarifaire d'un article à gamme.

IBOArticleTarifGammeClient3

Tarif pour un client d'un article à gamme.

Interface héritée

Syntaxe	Description
IBIArticleTarifGamme3	Cf. Interface IBIArticleTarifGamme3 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Client() As IBOClient3	Client auquel est associée le tarif de l'article à gamme.
Lecture seule	FactoryArticleTarifGammeClient() As IBITypeObjectFactory	Fabrique un objet Tarif par client pour un article à gamme.
Lecture / Ecriture	TG_Ref() As String	Référence du tarif client pour l'article à gamme.

IBOArticleTarifGammeFournisseur3

Tarif pour un fournisseur d'un article à gamme.

Interface héritée

Syntaxe	Description
IBIArticleTarifGamme3	Cf. Interface IBIArticleTarifGamme3 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	FactoryArticleTarifGammeFournisseur() As IBITypeObjectFactory	Fabrique un objet Tarif fournisseur article à gamme.
Lecture seule	Fournisseur() As IBOFournisseur3	Fournisseur auquel est associé le tarif de l'article à gamme.
Lecture / Ecriture	TG_CodeBarre() As String	Code barre.
Lecture / Ecriture	TG_Ref() As String	Référence.

IBOArticleTarifQteCategorie3

Tarif par quantité, montant ou prix net pour une catégorie tarifaire.

Interface héritée

Syntaxe	Description
IBIArticleTarifQte3	Cf. Interface IBIArticleTarifQte3 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	CategorieTarif() As IBPCategorieTarif	Catégorie tarifaire à laquelle est associé le tarif par quantité, montant ou prix net.
Lecture seule	FactoryArticleTarifQteCategorie() As IBITypeObjectFactory	Fabrique un objet Tarif par quantité, montant ou prix net pour la catégorie tarifaire.

IBOArticleTarifQteClient3

Tarif par quantité, montant ou prix net pour un client de l'article.

Interface héritée

Syntaxe	Description
IBIArticleTarifQte3	Cf. Interface IBIArticleTarifQte3 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Client() As IBOClient3	Client auquel est associé le tarif par quantité, montant ou prix net.
Lecture seule	FactoryArticleTarifQteClient() As IBITypeObjectFactory	Fabrique un Tarif par quantité, montant ou prix net pour le client de l'article.

IBOArticleTarifQteFournisseur3

Tarif par quantité, montant ou prix net pour un fournisseur de l'article.

Interface héritée

Syntaxe	Description
IBIArticleTarifQte3	Cf. Interface IBIArticleTarifQte3 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	FactoryArticleTarifQteFournisseur() As IBITypeObjectFactory	Fabrique un objet Tarif par quantité, montant ou prix net pour un fournisseur de l'article.
Lecture seule	Fournisseur() As IBOFournisseur3	Fournisseur auquel est associé le tarif par quantité, montant ou prix net.

IBOBaremeCommission

Commissions.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Base() As BaremeCommissionBase	Base de calcul.
Lecture / Ecriture	Calcul() As BaremeCommissionCalcul	Type de calcul.
Lecture / Ecriture	DateDebut() As Date	Date de début.

Accès	Syntaxe	Description
Lecture / Ecriture	DateFin() As Date	Date de Fin.
Lecture / Ecriture	Domaine() As BaremeCommissionDomaine	Domaine.
Lecture seule	FactoryBaremeCommission() As IBOBaremeCommissionFactory	Fabrique un objet commissions.
Lecture seule	FactoryBaremeCommissionQte() As IBITypeObjectFactory	Fabrique un objet gamme de remises commissions.
Lecture / Ecriture	Interesse() As BaremeCommissionInteresse	Type de commissions.
Lecture / Ecriture	Intitule() As String	Intitulé.
Lecture / Ecriture	Objectif() As BaremeCommissionObjectif	Objectif.

IBOBaremeCommissionQte

Gamme de remise des commissions.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	BaremeCommission() As IBOBaremeCommission	Objet commissions.
Lecture/ Ecriture	BorneSup() As Double	Borne supérieure.
Lecture/ Ecriture	Remise() As IRemise2	Remise.

IBOBaremeRabais

Rabais, remises et ristournes.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture/ Ecriture	ArticleRemise() As IBOArticle3	Article de remise.
Lecture/ Ecriture	Calcul() As BaremeRabaisCalcul	Type de calcul (cf énumérateur BaremeRabaisCalcul).
Lecture/ Ecriture	DateDebut() As Date	Date de début.
Lecture/ Ecriture	DateFin() As Date	Date de Fin.
Lecture seule	FactoryBaremeRabais() As IBOBaremeRabaisFactory	Fabrique un objet rabais, remises et ristournes.
Lecture seule	FactoryBaremeRabaisQte() As IBITypeObjectFactory	Fabrique un objet gamme de remises rabais, remises et ristournes.
Lecture / Ecriture	Fournisseur() As IBOFournisseur3	Fournisseur (retourne null dans le cas d'un rabais client).
Lecture/ Ecriture	Intitule() As String	Intitulé.
Lecture/ Ecriture	Objectif() As BaremeRabaisObjectif	Objectif (cf. énumérateur BaremeRabaisObjectif).
Lecture seule	Type() As BaremeRabaisType	Type de barême (cf. énumérateur BaremeRabaisType).

IBOBaremeRabaisQte

Gamme de remise des rabais, remises et ristournes.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	BaremeRabais() As IBOBaremeRabais	Objet rabais, remises et ristournes.
Lecture/ Ecriture	BorneSup() As Double	Borne supérieure.
Lecture/ Ecriture	Remise() As IRemise2	Remise.

IBOBaremeSolde

Soldes et promotions.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture/ Ecriture	DateDebut() As Date	Date de début.
Lecture/ Ecriture	DateFin() As Date	Date de Fin.
Lecture seule	FactoryBaremeSolde() As IBOBaremeSoldeFactory	Fabrique un objet rabais, remises et ristournes.
Lecture / Ecriture	Fournisseur() As IBOFournisseur3	Fournisseur (retourne null dans le cas d'un Soldes et promotions client).
Lecture/ Ecriture	Intitule() As String	Intitulé.
Lecture/ Ecriture	Remise() As IRemise2	Remise.
Lecture seule	Type() As BaremeSoldeType	Type de barême (cf. énumérateur BaremeSoldeType).

IBODepot3

Dépôt de stockage.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Adresse() As IAdresse	Adresse.
Lecture / Ecriture	CategorieCompta() As IBICategorieCompta	Catégorie comptable.
Lecture / Ecriture	DE_Code() As String	Code dépôt.
Lecture / Ecriture	DE_Contact() As String	Responsable.
Lecture / Ecriture	DE_Exclure() As Boolean	Exclure du réapprovisionnement.
Lecture / Ecriture	DE_Intitule() As String	Intitulé.
Lecture / Ecriture	DE_Souche(ByVal *sElt* As DomaineType) As IBISouche	Souche des documents en saisie.
Lecture / Ecriture	DefaultEmplacement() As IBODepotEmplacement	Emplacement par defaut.

Accès	Syntaxe	Description
Lecture seule	FactoryDepot() As IBODepotFactory2	Fabrique un objet Dépôt de stockage.
Lecture seule	FactoryDepotContact() As IBITypeObjectFactory	Fabrique un objet Contact du dépôt.
Lecture seule	FactoryDepotEmplacement() As IBODepotEmplacementFactory	Fabrique un objet emplacement du dépôt.
Lecture / Ecriture	Telecom() As ITelecom	Télécommunications.
Lecture	Compteur () As int	Compteur du dépôt.

IBODepotContact3

Contact du dépôt.

Interface héritée

Syntaxe	Description
IBIContact2	Cf. Interface IBIContact2 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Depot() As IBODepot3	Dépôt auquel est associé le contact.
Lecture seule	FactoryDepotContact() As IBITypeObjectFactory	Fabrique un objet Contact du dépôt.

IBODepotEmplacement

Dépôt emplacement.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Depot() As IBODepot3	Dépôt.
Lecture / Ecriture	DP_Code() As String	Code emplacement.
Lecture / Ecriture	DP_Intitule() As String	Intitulé emplacement.

Accès	Syntaxe	Description
Lecture / Ecriture	DP_Type() As DepotEmplType	Type d'emplacement (cf. énumérateur DepotEmplType).
Lecture / Ecriture	DP_Zone() As DepotEmplZone	Zone emplacement (cf. énumérateur DepotEmplZone).
Lecture seule	FactoryDepotEmplacement() As IBODepotEmplacementFactory	Fabrique un objet dépôt emplacement.

IBODocument3

Document.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Collaborateur() As IBOCollaborateur	Collaborateur.
Lecture / Ecriture	DO_DepotStockage() As IBODepot3	Dépôt de stockage.
Lecture / Ecriture	DO_Date() As Date	Date.
Lecture seule	DO_Domaine() As DomaineType	Domaine (cf. énumérateur DomaineType).
Lecture seule	DO_Heure() As Date	Heure.
Lecture seule	DO_Imprim() As Boolean	Document imprimé (True) ou non (False).
Lecture / Ecriture	DO_Piece() As String	N° de pièce.
Lecture / Ecriture	DO_Ref() As String	Référence.
Lecture / Ecriture	DO_Transfere() As Boolean	Document transféré (True) ou non (False)
Lecture / Ecriture	DO_Type() As DocumentType	Type (cf. énumérateur DocumentType).
Lecture seule	FactoryDocument() As IBODocumentFactory	Fabrique un objet Document.
Lecture seule	FactoryDocumentLigne() As IBODocumentLigneFactory	Fabrique un objet Document ligne associé à l'objet Document.
Lecture seule	FactoryDocumentMedia() As IBITypeObjectFactory	Retourne l'objet document média associé au document.
Lecture seule	InfoLibre() As IBIValues	Valeur information libre.

Accès	Syntaxe	Description
Lecture / Ecriture	Souche() As IBISouche	Souche du document.
Lecture seule	TotalHT() As Double	Total HT du document.
Lecture / Ecriture	DO_TotalHTNet() As Double	Total HT net du document
Lecture / Ecriture	DO_TotalTTC() As Double	Total TTC du document
Lecture / Ecriture	DO_NetAPayer() As Double	Net à payer du document
Lecture / Ecriture	DO_MontantRegle() As Double	Montant reglé du document

Méthode

Syntaxe	Description
SetDefaultDO_Piece()	Affecte le prochain numéro de pièce en fonction de la souche.
RecalculTotaux (Bool bWrite)	Recalcul du total HT, du HT Net, du Net à payer et du Total TTC du document. Le booléen bWrite permet d'indiquer si le document doit être enregistré après le recalcul.
SetAutoRecalculTotaux (Bool bAuto)	bAuto = false : Permet d'empêcher le recalcul automatique des totaux du document à chaque écriture de ligne.
HasAutoRecalculTotaux ()	Renvoie la façon dont les lignes du document recalculent ou non les totaux automatiquement

IBODocumentAchat3

Document d'achat.

Interface héritée

Syntaxe	Description
IBODocumentPart3	Cf. Interface IBODocumentPart3 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	ConditionPaiement() As DocumentConditionPaiementType	Condition de paiement (cf.énumérateur DocumentConditionPaiementType).
Lecture/ Ecriture	DO_Motif() As String	Motif de rectification (spécifique aux versions espagnoles)
Lecture seule	FactoryDocumentAchat() As IBODocumentAchatFactory3	Fabrique un objet Document d'achat.
Lecture seule	FactoryDocumentAcompte() As IBITypeObjectFactory	Fabrique un objet Acompte.
Lecture seule	FactoryDocumentEcheance() As IBITypeObjectFactory	Fabrique un objet Echéance.

Accès	Syntaxe	Description
Lecture seule	Fournisseur() As IBOFournisseur3	Fournisseur.
Lecture / Ecriture	ModeleReglement() As IBOModeleReglement	Modèle de règlement.
Lecture seule	Valorisation() As IDOCValorisation	Informations sur la valorisation du document.

Méthode

Syntaxe	Description
SetDefaultFournisseur(pFournisseur As IBOFournisseur3)	Initialise les propriétés en fonction des données fournisseur.

Traitements et initialisations par défaut

Méthode SetDefaultFournisseur(ByVal pFournisseur As IBOFournisseur3)

L'appel de la méthode *SetDefaultFournisseur()* initialise les propriétés suivantes :

Propriété	Valeur	Description
DO_Langue () As LangueType	DO_Langue = Fournisseur.CT_Langue	Affecte la langue du fournisseur.
CategorieCompta () As IBICategorieCompta	CatégorieCompta = Fournisseur.CategorieCompta	Affecte la catégorie comptable du fournisseur.
Devise () As IBPDevise2	Devise = Fournisseur.Devise	Affecte la devise du fournisseur.
DO_Cours () As Double	DO_Cours = Fournisseur.DO_Cours	Affecte le cours de la devise du fournisseur.
DepotStockage () As IBODepot3	DepotStockage = Fournisseur.Depot	Affecte le dépôt principal du fournisseur.
TiersPayeur () As IBOTiersPart3	TiersPayeur = Fournisseur.TiersPayeur	Affecte le tiers payeur du fournisseur.
CompteG () As IBOCompteG3	CompteG = Fournisseur.CompteGPrinc	Affecte le compte général principal du fournisseur.
CompteA () As IBOCompteA3	CompteA = Fournisseur.CompteA	Affecte le code affaire du fournisseur.
Expedition () As IBPExpedition3	Expedition = Fournisseur.Expedition	Affecte les modes et frais d'expédition du fournisseur.
ConditionLivraison () As IBPConditionLivraison	ConditionLivraison = Fournisseur.ConditionLivraison	Affecte les conditions de livraison du fournisseur.
CompteAIFRS () As IBOCompteA3	CompteAIFRS = Fournisseur.CompteAIFRS	Affecte le compte IFRS du fournisseur.

Propriété	Valeur	Description
Collaborateur() As IBOCollaborateur	Collaborateur = Fournisseur.Representant	Affecte le représentant du fournisseur.
DO_TxEscompte() As Double	DO_TxEscompte = Fournisseur.TauxEscompte	Affecte le taux d'escompte du fournisseur.
CompteAIFRS () As IBOCompteA3	CompteAIFRS = Fournisseur.CompteAIFRS	Affecte le compte IFRS du fournisseur.

Méthode SetDefault()

Exécute la méthode *SetDefaultFournisseur()* dans le cas où le fournisseur est renseigné.

IBODocumentAchatLigne3

Ligne de document d'achat.

Interface héritée

Syntaxe	Description
IBODocumentPartLigne3	Cf. Interface IBODocumentPartLigne3 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	DL_Frais() As Double	Frais d'approche.
Lecture seule	DocumentAchat() As IBODocumentAchat2	Document d'achat auquel est associée la ligne de document d'achat.
Lecture seule	Fournisseur() As IBOFournisseur2	Fournisseur.
Lecture seule	StockValeur() As Double	Stock en valeur.
Lecture seule	Valorisation() As IDocLigneValorisation	Valorisation de la ligne.

Méthode

Syntaxe	Description
SetDefaultRemise ()	Initialise les propriétés en fonction des données de la remise.

Traitements et initialisations par défaut

Méthode SetDefaultArticleReferenceFournisseur(ByVal sRef As String, ByVal Qte As Double)

La méthode executée sera fonction de la valeur affectée au paramètre *sRef* :

- *sRef* = Référence fournisseur ou code barre référence fournisseur

  Méthode exécutée : SetDefaultArticle() (cf. IBODocumentLigne3)

- *sRef* = Référence de gamme ou code barre d'un énuméré de gamme

  Méthode exécutée : SetDefaultArticleMonoGamme() / SetDefaultArticleDoubleGamme()(cf. IBODocumentLigne3)

IBODocumentAcompte3

Acompte.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	DocumentPart() AsIBODocumentPart3	Document de type Achat/Vente.
Lecture / Ecriture	DR_Date() As Date	Date.
Lecture / Ecriture	DR_Libelle() As String	Libellé.
Lecture / Ecriture	DR_Montant() As Double	Montant.
Lecture / Ecriture	DR_MontantDev() As Double	Montant en devise.
Lecture seule	FactoryDocumentAcompte() As IBITypeObjectFactory	Fabrique un objet Acompte.
Lecture / Ecriture	Reglement() As IBPReglement3	Mode de règlement.
Lecture	HasPieceAcompte () As boolean	Teste l'existence du document acompte lié
Lecture	PieceAcompte () As IBODocumentPart3*	Lecture du document acompte lié
Lecture	HasDocumentReglement () As boolean	Teste l'existence du document Règlement lié
Lecture	DocumentReglement () As IBODocumentReglement *	Lecture du document Règlement lié à l'acompte

Traitements et initialisations par défaut

Méthode SetDefault()

Si le document est en devise et que DR_MontantDev est null, alors affectation de DR_MontantDev en fonction de DR_Montant et du cours de la devise du document.

Si le document est en devise et que DR_Montant est null, alors affectation de DR_Montant en fonction de DR_MontantDev et du cours de la devise du document.

IBODocumentEcheance3

Echéance.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	DocumentPart() AsIBODocumentPart3	Document de type Achat/Vente.
Lecture / Ecriture	DR_Date() As Date	Date.
Lecture seule	DR_Equil() As Boolean	Ligne d'équilibrage (True) ou non (False).
Lecture / Ecriture	DR_Montant() As Double	Montant.
Lecture seule	DR_MontantCalcule() As Double	Montant calculé en fonction du net à payer.
Lecture / Ecriture	DR_MontantDev() As Double	Montant en devise.
Lecture seule	DR_MontantCalculeDev() As Double	Montant calculé en devise en fonction du net à payer en devise.
Lecture / Ecriture	DR_Pourcent() As Double	Pourcentage du règlement.
Lecture seule	FactoryDocumentEcheance() As IBITypeObjectFactory	Fabrique un objet Echéance de document.
Lecture / Ecriture	Reglement() As IBPReglement3	Mode de règlement.

Méthode SetDefault()

Si le document est en devise et que DR_MontantDev est null, alors affectation de DR_MontantDev en fonction de DR_Montant et du cours de la devise du document.

Si le document est en devise et que DR_Montant est null, alors affectation de DR_Montant en fonction de DR_MontantDev et du cours de la devise du document.

IBODocumentInterne3

Document interne.

Interface héritée

Syntaxe	Description
IBODocumentPart3	Cf. Interface IBODocumentPart3 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	CategorieTarif() As IBPCategorieTarif	Catégorie tarifaire.
Lecture / Ecriture	CentraleAchat() As IBOClient3	Client centrale d'achat.
Lecture seule	Client() As IBOClient3	Client.
Lecture / Ecriture	DO_NbFacture() As Short	Nombre de factures.
Lecture seule	FactoryDocumentInterne() As IBODocumentInterneFactory3	Fabrique un objet Document interne.
Lecture / Ecriture	LieuLivraison() As IBOClientLivraison3	Lieu de livraison.
Lecture / Ecriture	Periodicite() As IBPPeriodicite	Périodicité.

Méthode

Syntaxe	Description
SetDefaultClient(pClient As IBOClient3)	Initialise les propriétés en fonction des données du client.

Traitements et initialisations par défaut

Méthode SetDefaultClient(ByVal pClient As IBOClient3)

L'appel de cette méthode affecte les mêmes propriétés que la méthode *SetDefaultClient()* de *IBODocumentVente3*.

Méthode SetDefault()

Exécute la méthode *SetDefaultClient()* dans le cas où le client est renseigné.

IBODocumentInterneLigne3

Ligne de document interne.

Interface héritée

Syntaxe	Description
IBODocumentPartLigne3	Cf. Interface IBODocumentPartLigne3 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	AC_RefClient() As String	Référence article client.
Lecture / Ecriture	ArticleCompose() As IBOArticle3	Référence de l'article composé. Si l'article est un composé : ArticleCompose = Article Si l'article est un composant : ArticleCompose = Article de l'article composé
Lecture seule	Client() As IBOClient3	Client.
Lecture / Ecriture	DL_QteRessource() As Long	Quantité ressource.
Lecture seule	DocumentInterne() As IBODocumentInterne3	Document interne auquel est associée la ligne de document interne.
Lecture seule	IsComposeDeComposant() As Boolean	Retourne vrai si l'article de la ligne est de nomenclature Commercial/Composant.
Lecture seule	IsComposeDeCompose() As Boolean	Retourne vrai si l'article de la ligne est de nomenclature Commercial/Composé.
Lecture seule	LignesComposants() As IBICollection	Retourne une collection de ligne de composants si l'article de la ligne est un composé.
Lecture seule	Marge() As Double	Marge en valeur de la ligne.
Lecture seule	PrixRevient() As Double	Prix de revient de la ligne.

IBODocumentLigne3

Fabrique un objet Ligne de document.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Article() As IBOArticle3	Article.
Lecture seule	ArticleGammeEnum1() As IBOArticleGammeEnum3	Enuméré de gamme 1.
Lecture seule	ArticleGammeEnum2() As IBOArticleGammeEnum3	Enuméré de gamme 2.
Lecture / Ecriture	Collaborateur() As IBOCollaborateur	Collaborateur.
Lecture seule	DefaultCMUP() As Double	CMUP de l'article de la ligne de document.
Lecture / Ecriture	Depot() As IBODepot3	Dépôt.
Lecture / Ecriture	DL_CMUP() As Double	CMUP.
Lecture / Ecriture	DL_Design() As String	Désignation.
Lecture seule	DL_FactPoids() As Boolean	Facturation sur le poids.
Lecture seule	DL_MontantHT() As Double	Montant HT.
Lecture seule	DL_MvtStock() As DocumentLigneMvtStockType	Mouvement de stock (Cf. énumérateur DocumentLigneMvtStockType).
Lecture / Ecriture	DL_Operation() As String	Numéro d'opération.
Lecture / Ecriture	DL_PoidsBrut() As Double	Poids brut.
Lecture / Ecriture	DL_PoidsNet() As Double	Poids net.
Lecture / Ecriture	DL_PieceOfProd() As Long	Numéro de pièce gestion de production.
Lecture / Ecriture	DL_PrixRU() As Double	Prix de revient unitaire.
Lecture / Ecriture	DL_PrixUnitaire() As Double	Prix unitaire.
Lecture / Ecriture	DL_Qte() As Double	Quantité.
Lecture / Ecriture	DO_Date() As Date	Date de la ligne de document.
Lecture seule	DO_Domaine() As DomaineType	Domaine (Cf. énumérateur DomaineType).
Lecture / Ecriture	DO_Ref() As String	Numéro de pièce externe.
Lecture seule	DO_Type() As DocumentType	Type de document (Cf. énumérateur DocumentType).
Lecture seule	Document() As IBODocument3	Document auquel est associée laligne de document.
Lecture / Ecriture	EU_Enumere() As String	Enuméré de conditionnement.
Lecture / Ecriture	EU_Qte() As Double	Quantité conditionnement.
Lecture seule	FactoryDocumentLigne() As IBODocumentLigneFactory	Fabrique un objet Ligne de document.
Lecture seule	FactoryDocumentLigneEmplacement() As IBITypeObjectFactory	Fabrique un objet emplacement de ligne.

Accès	Syntaxe	Description
Lecture seule	FactoryDocumentLigneLienCM() As IBITypeObjectFactory	Fabrique un objet lien contremarque.
Lecture seule	InfoLibre() As IBIValues	Valeur information libre.
Lecture / Ecriture	Lot() As IBOArticleLot	Lot de la ligne.
Lecture écriture	LS_ComplementOut() As String	Complément série/lot de la ligne de sortie.
Lecture / Ecriture	LS_Fabrication() As Date	Date de fabrication.
Lecture / Ecriture	LS_NoSerie() As String	Numéro de série de l'article (article géré par lot ou numéro de série).
Lecture / Ecriture	LS_Peremption() As Date	Date de péremption.
Lecture / Ecriture	Ressource() As IBIRessource	Ressource.
Lecture / Ecriture	TxtComplementaire() As String	Texte complémentaire.
Lecture / Ecriture	Valorisee() As Boolean	Ligne valorisée (True) ou non (False).
Lecture seule	DL_No () As int	Lecture de l'identifiant d'une ligne de document

Méthodes

Syntaxe	Description
MoveDown() As Boolean	Retourne vrai si la ligne a pu être descendue.
MoveUp() As Boolean	Retourne vrai si la ligne a pu être montée.
SetDefaultArticle(pArticle As IBOArticle3, ByVal Qte As Double)	Initialise les propriétés en fonction de l'article et de la quantité.
SetDefaultArticleConditionnement(pArtCond As IBOArticleCond3, ByVal Qte As Double)	Initialise les propriétés en fonction du conditionnement et de la quantité.
SetDefaultArticleDoubleGamme(pEnum1 As IBOArticleGammeEnum3, pEnum2 As IBOArticleGammeEnum3, ByVal Qte As Double)	Initialise les propriétés en fonction des énumérés de gamme et de la quantité.
SetDefaultArticleMonoGamme(pEnum As IBOArticleGammeEnum3, ByVal Qte As Double)	Initialise les propriétés en fonction de l'énuméré de gamme et de la quantité.
SetDefaultArticleReference(ByVal sRef As String, ByVal Qte As Double)	Initialise les propriétés en fonction de la référence et de la quantité.
SetDefaultLot(pLot As IBOArticleDepotLot, ByVal Qte As Double)	Initialise les proprieties en function du Série/lot et de la quantité.
WriteBefore(*ligne* As IBODocumentLigne3)	Ecrit la ligne avant la ligne passée en paramètre.
WriteDefaultBefore(*ligne* As IBODocumentLigne3)	Ecrit la ligne avant la ligne passée en paramètre avec exécution des automatismes SetDefault.

Traitements et initialisations par défaut

Méthode SetDefaultArticle(ByVal pArticle As IBOArticle3, ByVal Qte As Double)

L'appel de la méthode *SetDefaultArticle()* initialise les propriétés suivantes :

Domaine [1]	Propriété	Valeur	Description
V/A/S/I	ArticleGammeEnum1 () As IBOArticleGammeEnum3	ArticleGammeEnum1 = Nothing	Initialisation de la gamme1.
V/A/S/I	ArticleGammeEnum2 () As IBOArticleGammeEnum3	ArticleGammeEnum2 = Nothing	Initialisation de la gamme2.
V/A/S/I	DO_Ref () As string	DO_Ref = Entete.DO_Ref	Affecte la référence de l'entête.
V/A/I	CompteA () As IBOCompteA3	CompteA = Entete.CompteA	Affecte le code affaire de l'entête.
V/A/I	DO_DateLivr () As Date	Do_DateLivr = Entete.DO_DateLivr	Affecte la date de livraison de l'entête.
V/A/S/I	DO_Date () As Date	DO_Date = Entete.DO_Date	Affecte la date de l'entête.
V	Collaborateur () As IBOCollaborateur	Representant = Entete.Representant	Affecte le représentant de l'entête.
V/A/I	Devise () As IBPDevise	Devise = Entete.Devise	Affecte la devise de l'entête.
V/A/I	Depot () As IBODepot3	Depot = Entete.DepotStockage	Affecte le dépôt de stockage de l'entête si l'article est suivi en stock.
S	Depot () As IBODepot3	Depot = Entete.DepotStockage	Affecte le dépôt de stockage de l'entête.
V/A/I	AF_RefFourniss () As String	AF_RefFourniss = ArticleTarifFournisseur.Reference	Affecte la référence du fournisseur principal.
V/A/I/S	DL_Design () As String	DL_Design = Article.AR_Design	Affecte la désignation de l'article.

Domaine [1]	Propriété	Valeur	Description
		Si Entete.DO_Langue Renseigné alors DL_Design = Article.AR_Langue1/2	
V	AC_RefClient () As String	AC_RefClient = ArticleTarifClient.Reference	Affecte la référence du client.
V/A/I/S	DL_PoidsNet () As Double	DL_PoidsNet = Article.AR_PoidsNet	Affecte le poids net de l'article.
V/A/I/S	DL_PoidsBrut () As Double	DL_PoidsBrut = Article.AR_PoidsBrut	Affecte le poids brut de l'article.
V/A/I	DL_Escompte () As Boolean	DL_Escompte = Article.Ar_Escompte	Affecte l'option « Non soumis à l'escompte » de l'article.
V/A/I/S	DL_FactPoids () As Boolean	DL_FactPoids = Article.AR_FactPoids	Affecte l'option « Facturation / poids net » de l'article.
V/A/I/S	EU_Enumere () As String	EU_Enumere = Article.Unite	Affecte l'unité de vente de l'article.
V	DL_PrixUnitaire () As Double	DL_PrixUnitaire = ArticleTarifClient.Prix Si ArticleTarifClient.Prix = 0 Alors DL_PrixUnitaire = Article.PrixAchat * Article.Coef Si Article.PrixAchat * Article.Coef = 0 Alors DL_Prixunitaire = Article.AR_PrixVen	Affecte le prix de vente de l'article si le prix est HT.
V	DL_PrixTTC () As Double	DL_PrixTTC = ArticleTarifClient.Prix Si ArticleTarifClient.Prix = 0 Alors DL_PrixTTC = Article.PrixAchat * Article.Coef Si Article.PrixAchat * Article.Coef = 0 Alors DL_PrixTTC = Article.AR_PrixVen	Affecte le prix de vente de l'article si le prix est TTC.
A	DL_PrixUnitaire () As Double	DL_PrixUnitaire = ArticleTarifFournisseur.Prix Si ArticleTarifFournisseur.Prix = 0	Affecte le prix d'achat de l'article.

Domaine [1]	Propriété	Valeur	Description
		Alors DL_Prixunitaire = Article.AR_PrixAch	

Domaine [1] :

V : Document de Vente

A : Document d'Achat

S : Document de Stock

I : Document Interne

Méthode SetDefaultArticleConditionnement(ByVal pArtCond As IBOArticleCond3, ByVal Qte As Double)

L'appel de cette méthode exécute la méthode *SetDefaultArticle()* de *IBODocumentLigne3* en prenant en compte l'énuméré de conditionnement et la quantité passés en paramètres. Seule l'affectation de la propriété suivante varie :

Propriété	Valeur	Description
DL_QTE () As Double	DL_Qte = Quantité passée en paramètre * ArticleCond.EC_Quantite	Affecte la quantité en fonction de l'énuméré de conditionnement.

Méthode SetDefaultArticleMonoGamme(ByVal pEnum As IBOArticleGammeEnum3, ByVal Qte As Double)

L'appel de cette méthode exécute la méthode *SetDefaultArticle()* de *IBODocumentLigne3* en prenant en compte l'énuméré de gamme et la quantité passés en paramètres. Seule l'affectation de la propriété suivante varie :

Propriété	Valeur	Description
ArticleGammeEnum1 () As IBOArticleGammeEnum3	ArticleGammeEnum1 = Article.Gamme1	Affecte l'énuméré de gamme1 de l'article.

Méthode SetDefaultArticleDoubleGamme(ByVal pEnum1 As IBOArticleGammeEnum3, ByVal pEnum2 As IBOArticleGammeEnum3, ByVal Qte As Double)

L'appel de cette méthode exécute la méthode *SetDefaultArticle()* de *IBODocumentLigne3* en prenant en compte les deux énumérés de gamme et la quantité passés en paramètres. L'affectation des propriétés suivantes varie :

Propriété	Valeur	Description
ArticleGammeEnum1 () As IBOArticleGammeEnum3	ArticleGammeEnum1 = Article.Gamme1	Affecte l'énuméré de gamme1 de l'article.
ArticleGammeEnum2 () As IBOArticleGammeEnum3	ArticleGammeEnum2 = Article.Gamme2	Affecte l'énuméré de gamme2 de l'article.

Méthode SetDefaultLot(pLot As IBOArticleDepotLot, ByVal Qte As Double)

L'appel de cette méthode exécute la méthode *SetDefaultArticle()* de *IBODocumentLigne3* en prenant en compte le série/lot et la quantité passés en paramètres. L'affectation des propriétés suivantes varie :

Propriété	Valeur	Description
Lot () As IBOArticleDepotLot	Série/lot passé en paramètre	Affecte le série/lot.
Depot() As IBODepot3	Dépôt du série/lot passé en paramètre	Affecte le dépôt du série/lot.

Méthode SetDefaultArticleReference(ByVal sRef As String, ByVal Qte As Double)

La méthode executée sera fonction de la valeur affectée au paramètre *sRef* :

- *sRef* = Référence article ou code barre Article

  Méthode exécutée : SetDefaultArticle()

- *sRef* = Référence de gamme ou code barre d'un énuméré de gamme

  Méthode exécutée : SetDefaultArticleMonoGamme() / SetDefaultArticleDoubleGamme()

- *sRef* = Enuméré de conditionnement

  Méthode exécutée : SetDefaultArticleConditionnement()

Avec Sage 100cloud Objets Métiers, la valeur à affecter à la quantité (propriété DL_Qte) doit être du signe de celle réellement stockée dans la base de données. Il existe toutefois une exception sur les documents de stock de type fabrication (Préparation de fabrication, Ordre de fabrication et Bon de fabrication) pour lesquels la quantité saisie peut être positive ou négative alors que la valeur réellement stockée est toujours positive.

Vous trouverez ci-dessous le détail des quantités à affecter en fonction du type de document :

- **Documents de vente :**

Type de Document	Quantité saisie / affichée Sage 100c Gestion commerciale	Quantité saisie / affichée Sage 100cloud Objets Métiers	Type mouvement stock	Quantité stockée
Devis	Positive	Positive	Aucun	Positive
Bon de commande	Positive	Positive	Aucun	Positive
Préparation de livraison	Positive	Positive	Aucun	Positive
Bon de livraison	Positive	Positive	Sortie en quantité	Positive
	Négative	Négative	Entrée en quantité	Négative
Bon de retour	Positive	Positive	Entrée en quantité	Positive
Bon d'avoir financier	Positive	Positive	Aucun	Positive
	Négative	Négative	Aucun	Négative
Facture	Positive	Positive	Sortie en quantité	Positive
	Négative	Négative	Entrée en quantité	Négative
Facture de retour	Positive	Négative	Entrée en quantité	Négative
Facture d'avoir	Positive	Négative	Aucun	Négative
	Négative	Positive	Aucun	Positive

- **Documents d'achat :**

Type de Document	Quantité saisie / affichée Sage 100c Gestion commerciale	Quantité saisie / affichée Sage 100cloud Objets Métiers	Type mouvement stock	Quantité stockée
Préparation de commande	Positive	Positive	Aucun	Positive
Bon de commande	Positive	Positive	Aucun	Positive
Bon de livraison	Positive	Positive	Entrée en quantité	Positive
	Négative	Négative	Sortie en quantité	Négative
Bon de retour	Positive	Positive	Sortie en quantité	Positive
Bon d'avoir financier	Positive	Positive	Sortie en montant	Positive
	Négative	Négative	Sortie en montant	Négative

Type de Document	Quantité saisie / affichée Sage 100c Gestion commerciale	Quantité saisie / affichée Sage 100cloud Objets Métiers	Type mouvement stock	Quantité stockée
Facture	Positive	Positive	Entrée en quantité	Positive
	Négative	Négative	Sortie en quantité	Négative
Facture de retour	Positive	Négative	Sortie en quantité	Négative
Facture d'avoir	Positive	Négative	Sortie en montant	Négative
	Négative	Positive	Sortie en montant	Positive

- **Documents de stock :**

Document	Quantité saisie / affichée Sage 100c Gestion commerciale	Quantité saisie / affichée Sage 100cloud Objets Métiers	Type mouvement stock	Quantité stockée
Mouvement d'entrée	Positive	Positive	Entrée en quantité	Positive
Mouvement de sortie	Positive	Positive	Sortie en quantité	Positive
Dépréciation	Positive	Positive	Sortie en montant	Positive
	Négative	Négative	Sortie en montant	Positive
Virement de dépôt à dépôt	Positive	Positive	- Sortie en quantité pour la ligne de sortie - Entrée en quantité pour la ligne d'entrée	Positive
Préparation de fabrication	Positive	Positive	Aucun	Positive
	Négative	Négative	Aucun	Positive
Ordre de fabrication	Positive	Positive	- Sortie en quantité pour les lignes des composants - Aucun pour la ligne du composé	Positive
	Négative	Négative	- Sortie en quantité pour la ligne du composé - Aucun pour les lignes des composants	Positive
Bon de fabrication	Positive	Positive	- Sortie en quantité pour les lignes des composants - Aucun pour la ligne du composé	Positive
	Négative	Négative	- Sortie en quantité pour la ligne du composé - Aucun pour les lignes des composants	Positive

- **Documents internes :**

Document	Quantité saisie / affichée Sage 100c Gestion commerciale	Quantité saisie / affichée Sage 100cloud Objets Métiers	Type mouvement stock	Quantité stockée
Aucun	Positive	Positive	Aucun	Positive
Entrée en stock	Positive	Positive	Entrée en quantité	Positive
Sortie de stock	Positive	Positive	Sortie en quantité	Positive
Gain financier	Positive	Positive	Aucun	Positive
Perte financière	Positive	Positive	Aucun	Positive

Méthode SetDefaultLot(pLot As IBOArticleDepotLot, ByVal Qte As Double)

L'appel de cette méthode exécute la méthode *SetDefaultArticle()* de *IBODocumentLigne3* en prenant en compte le lot article et la quantité passés en paramètres.

> *Remarque sur les Emplacements :*
>
> Lorsque le fichier commercial gère le multi-emplacements (Fichier / Paramètres société / Logistique / Gestion des stocks), la méthode WriteDefault() génère automatiquement les lignes d'emplacements en fonction du paramétrage défini sous la Gestion commerciale. Par contre, l'appel de la méthode Write() ne créé pas les lignes d'emplacements. Ainsi, si les lignes de documents sont créées avec la méthode Write(), il ne sera nécessaire que l'application utilisant Sage 100cloud Objets Métiers, créée manuellement les lignes d'emplacements (Cf. FactoryDocumentLigneEmplacement).

IBODocumentLigneEmplacement

Fabrique un objet emplacement de ligne de document.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	DL_Qte() As Double	Quantité sur l'emplacement.
Lecture seule	DL_QteAControler() As Double	Quantité à contrôler sur l'emplacement.
Lecture seule	DocumentLigne() As IBODocumentLigne3	Ligne de document associée à la ligne d'emplacement.
Lecture / Ecriture	Emplacement() As IBODepotEmplacement	Emplacement.
Lecture seule	FactoryDocumentLigneEmplacement() As IBITypeObjectFactory	Fabrique un objet emplacement de ligne de document.

IBODocumentLigneLienCM

Liens contremarque pour une ligne d'article géré en contremarque.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	DocumentLigneIn() As IBODocumentLigne3	Ligne de document d'entrée liée en contremarque.
Lecture seule	DocumentLigneOut() As IBODocumentLigne3	Ligne de document de sortie liée en contremarque.
Lecture seule	Qte() As Double	Quantité en contremarque.

IBODocumentPart3

Documents de type Achat ou Vente

Interface héritée

Syntaxe	Description
IBODocument3	Cf. Interface IBODocument3 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	CategorieCompta() As IBICategorieCompta	Catégorie comptable.

Accès	Syntaxe	Description
Lecture / Ecriture	CompteA() As IBOCompteA3	Compte analytique.
Lecture / Ecriture	CompteAIFRS() As IBOCompteA3	Compte analytique IFRS.
Lecture / Ecriture	CompteG() As IBOCompteG3	Compte général.
Lecture / Ecriture	ConditionLivraison() As IBPConditionLivraison	Condition de livraison.
Lecture / Ecriture	Devise() As IBPDevise2	Devise.
Lecture / Ecriture	DO_Contact() As String	Contact du document.
Lecture / Ecriture	DO_Coord(ByVal *sElt* As Short) As String	Coordonnées du document (indice en paramètre de 1 à 4).
Lecture / Ecriture	DO_Cours() As Double	Cours.
Lecture / Ecriture	DO_DateLivr() As Date	Date de livraison.
Lecture / Ecriture	DO_DateLivrRealisee() As Date	Date de livraison réalisée.
Lecture / Ecriture	DO_Ecart() As Double	Ecart de la valorisation.
Lecture / Ecriture	DO_Langue() As LangueType	Langue (Cf. énumérateur LangueType).
Lecture / Ecriture	DO_Provenance() As DocumentProvenanceType	Provenance du document (cf. énumérateur DocumentProvenanceType).
Lecture / Ecriture	DO_Regime() As Short	Régime.
Lecture / Ecriture	DO_Reliquat() As Boolean	Reliquat.
Lecture / Ecriture	DO_Statut() As DocumentStatutType	Statut (Cf. énumérateur DocumentStatutType).
Lecture / Ecriture	DO_Transaction() As Short	Transaction.
Lecture / Ecriture	DO_TxEscompte() As Double	Taux d'escompte.
Lecture / Ecriture	Expedition() As IBPExpedition3	Expédition.
Lecture seule	FactoryDocumentPart() As IBITypeObjectFactory	Fabrique un objet Document de type Achat/Vente.
Lecture / Ecriture	FraisExpedition() As Double	Frais d'expédition.
Lecture / Ecriture	Tiers() As IBOTiersPart3	Tiers.
Lecture / Ecriture	TiersPayeur() As IBOTiersPart3	Tiers payeur.

IBODocumentPartLigne3

Ligne de document de type Achat ou Vente.

Interface héritée

Syntaxe	Description
IBODocumentLigne3	Cf. Interface IBODocumentLigne3 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	AF_RefFourniss() As String	Référence fournisseur.
Lecture / Ecriture	CompteA() As IBOCompteA3	Compte analytique.
Lecture seule	DL_DateBC() As Date	Date de commande.
Lecture seule	DL_DateBL() As Date	Date de livraison.
Lecture seule	DL_Escompte() As Boolean	Article non soumis à l'escompte (*True*) ou soumis à l'escompte (*False*).
Lecture seule	DL_MontantTTC() As Double	Montant TTC.
Lecture seule	DL_NonLivre() As Boolean	Article non livré (*True*) ou livré (*False*).
Lecture seule	DL_PieceBC() As String	N° de pièce du Bon de commande.
Lecture seule	DL_PieceBL() As String	N° de pièce du Bon de livraison.
Lecture seule	DL_PUBC() As Double	Prix unitaire HT du Bon de commande.
Lecture / Ecriture	DL_PUDevise() As Double	Prix unitaire en devise.
Lecture / Ecriture	DL_PUTTC() As Double	Prix unitaire TTC.
Lecture / Ecriture	DL_QteBC() As Double	Quantités commandées.
Lecture / Ecriture	DL_QteBL() As Double	Quantités livrées.
Lecture seule	DL_TypePL() As DocumentLigneProvenanceType	Type de la pièce livrée (cf. énumérateur DocumentLigneProvenanceType).
Lecture / Ecriture	DO_DateLivr() As Date	Date de livraison.
Lecture seule	DocumentPart() As IBODocumentPart3	Document auquel est associé l'objet Ligne de document.
Lecture / Ecriture	IsRemiseExceptionnelle() As Boolean [1]	Ligne de remise exceptionnelle (*True*) ou non (*False*).
Lecture / Ecriture	IsRemisePied() As Boolean [1]	Ligne de remise en pied de document (*True*) ou non (*False*).
Lecture / Ecriture	IsTotalisatrice() As Boolean [1]	Ligne totalisatrice (*True*) ou non (*False*).
Lecture / Ecriture	Remise() As IRemise	Remise.
Lecture / Ecriture	Taxe(ByVal sElt As Integer) As IBOTaxe3	Taxes associées à la ligne de document.
Lecture / Ecriture	Tiers() As IBOTiersPart3	Tiers.

[1] : Ces propriétés doivent être renseignées avant d'exécuter *SetDefaultArticle()*. De plus, *IsTotalisatrice* et *IsRemiseExceptionnelle* adressent le même champ dans la base de données (*F_Docligne.DL_TRemExep*). Ainsi, il ne faut renseigner que l'une ou l'autre de ces propriétés.

Méthode

Syntaxe	Description
SetDefaultArticleReferenceFournisseur(ByVal sRef As String, ByVal Qte As Double)	Initialise les propriétés en fonction de la référence et de la quantité.

Traitements et initialisations par défaut

Méthode SetDefaultArticleReferenceFournisseur(ByVal sRef As String, ByVal Qte As Double)

La méthode executée sera fonction de la valeur affectée au paramètre *sRef* :

- *sRef* = Référence fournisseur ou code barre référence fournisseur

  Méthode exécutée : SetDefaultArticle() (cf. IBODocumentLigne3)

- *sRef* = Référence de gamme ou code barre d'un énuméré de gamme

  Méthode exécutée : SetDefaultArticleMonoGamme() / SetDefaultArticleDoubleGamme() (cf. IBODocumentLigne3)

IBODocumentStock3

Document de stock.

Interface héritée

Syntaxe	Description
IBODocument3	Cf. Interface IBODocument3 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	DepotDestination() As IBODepot3	Dépôt de destination (accessible uniquement pour les Virements de dépôt à dépôt).
Lecture / Ecriture	DepotOrigine() As IBODepot3	Dépôt d'origine (accessible uniquement pour les Virements de dépôt à dépôt).
Lecture seule	FactoryDocumentStock() As IBODocumentStockFactory3	Fabrique un objet Document de stock.

IBODocumentStockLigne3

Ligne de document de stock.

Interface héritée

Syntaxe	Description
IBODocumentLigne3	Cf. Interface IBODocumentLigne3 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	ArticleCompose() As IBOArticle3	Article composé.
Lecture / Ecriture	DO_DateFabrication() As Date	Date de fabrication.
Lecture / Ecriture	DL_DateAvancement() As Date	Date d'avancement.
Lecture / Ecriture	DL_NoColis() As String	Numéro de colis.
Lecture seule	DL_TNomencl() As Boolean	Ligne de type nomenclature (True) ou non (False).
Lecture seule	DocumentStock() As IBODocumentStock3	Document de stock auquel est associée la ligne de document de stock.

Pour les documents de type *Virement de dépôt à dépôt*, les lignes de sortie et d'entrée en stock devront être créées manuellement. En effet, la deuxième ligne (entrée en stock) n'est pas créée automatiquement comme le fait la gestion commerciale.
Ainsi, il conviendra de respecter la procédure suivante :

- Création de la ligne de sortie de stock en affectant à la propriété DepotOrigine le dépôt de sortie..

- Création de la ligne d'entrée en stock en affectant à la propriété DepotDestination le dépôt d'entrée.

Lors de la suppression d'une ligne d'entrée ou sortie de stock, il sera également nécessaire de supprimer manuellement la ligne associée à la ligne supprimée.

De plus, dans le cas d'un article géré en série ou lot, le numéro de série ou lot (propriété : *LS_NoSerie*) devra être renseigné pour la ligne de sortie et pour la ligne d'entrée en stock.

Pour les articles non suivis au CMUP, il sera nécessaire d'affecter à la propriété *DL_PrixUnitaire* de la ligne d'entrée en stock, la même valeur que celle renseignée sur la ligne de sortie de stock.

Pour les articles suivis au CMUP, la propriété *DL_PrixUnitaire* sera automatiquement affectée à la ligne d'entrée en stock.

IBODocumentVente3

Document de vente.

Interface héritée

Syntaxe	Description
IBODocumentPart3	Cf. Interface IBODocumentPart3 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	CategorieTarif() As IBPCategorieTarif	Catégorie tarifaire.
Lecture / Ecriture	CentraleAchat() As IBOClient3	Client centrale d'achat.
Lecture seule	Client() As IBOClient3	Client.
Lecture / Ecriture	ConditionPaiement() As DocumentConditionPaiementType	Condition de paiement (cf. énumérateur DocumentConditionPaiementType).
Lecture / Ecriture	DO_BLFact() As Boolean	Une facture par bon de livraison.
Lecture seule	DO_Cloture() As Boolean	Document clôturé.
Lecture / Ecriture	DO_Colisage() As Short	Colisage.
Lecture/ Ecriture	DO_Motif() As String	Motif de rectification (spécifique aux vesions espagnoles).
Lecture / Ecriture	DO_NbFacture() As Short	Nombre de factures.
Lecture / Ecriture	DO_NoWeb() As String	Numéro de commande du site marchand.
Lecture seule	DO_Valide() As Boolean	Document validé.
Lecture / Ecriture	DO_MotifDevis() As IBPMotifDevis	Motif devis perdus
Lecture seule	FactoryDocumentAcompte() As IBITypeObjectFactory	Fabrique un objet Acompte.
Lecture seule	FactoryDocumentEcheance() As IBITypeObjectFactory	Fabrique un objet Echéance.
Lecture seule	FactoryDocumentVente() As IBODocumentVenteFactory3	Fabrique un objet Document de vente.
Lecture seule	FactoryInfoComplement () As IBOFactoryInfoComplement	Fabrique un objet Informations complémentaires.
Lecture / Ecriture	LieuLivraison() As IBOClientLivraison3	Lieu de livraison.
Lecture / Ecriture	ModeleReglement() As IBOModeleReglement	Modèle de règlement.
Lecture / Ecriture	Periodicite() As IBPPeriodicite	Périodicité.
Lecture / Ecriture	Unite() As IBPUnite	Unité.

Accès	Syntaxe	Description
Lecture seule	Valorisation() As IDOCValorisation	Informations sur la valorisation du document.
Lecture / Ecriture	MotifDevis() As IBPMotifDevis	Motif devis perdu

Méthode

Syntaxe	Description
SetDefaultClient(pClient As IBOClient3)	Initialise les propriétés en fonction du client.

Traitements et initialisations par défaut

Méthode SetDefaultClient(ByVal pClient As IBOClient3)

L'appel de cette méthode effectue la même traitement que la méthode *SetDefaultFournisseur ()* de *IBODocumentAchat3*. En plus, les proriétés suivantes sont renseignées :

Propriété	Valeur	Description
Devise() As IBPDevise2	Devise = Client.Devise	Affecte la devise du client.
DO_Cours() As Double	DO_Cours = Devise.D_Cours	Affecte le cours de la devise si une devise est renseignée.
DepotStockage() As IBPDepot3	DepotStockage = Client.Depot	Affecte le dépôt du client.
TiersPayeur() As IBOTiersPart3	TiersPayeur = Client	Affecte le tiers payeur.
CompteA() As IBOCompteA3	CompteA = Client.CompteA	Affecte le code affaire du client.
Collaborateur() As IBOCollaborateur	Collaborateur = Client.Representant	Affecte le représentant du client.
CentraleAchat() As IBOClient3	CentraleAchat = Client.CentraleAchat	Affecte la centrale d'achat du client.
CompteG() As IBOCompteG3	CompteG = Client.CompteGPrinc	Affecte le compte général principal du client.
DO_TxEscompte () As Double	DO_TxEscompte = Client.CT_Taux02	Affecte le taux d'escompte du client.
Periodicite () AsIBPPeriodicité	Periodicite = Client.Periodicite	Affecte la périodicité du client.

Propriété	Valeur	Description
CategorieTarif () As IBPCategorieTarif	CategorieTarif = Client.CategorieTarif	Affecte la catégorie tarifaire du client. Si une centrale d'achat est renseignée, ce sera la catégorie tarifaire de la centrale qui sera affectée.
LieuLivraison () As IBOClientLivraison3	LieuLivraison = Client.LivraisonPrincipal	Affecte le dépôt de livraison principal du client.
DO_NbFacture () As Short	DO_NbFacture = Client.CT_Facture	Affecte le nombre de facture du client.
DO_BlFact () As Boolean	DO_BlFact = Client.BL_Fact	Affecte l'option BL/Facture du client.

Méthode SetDefault()

Exécute la méthode *SetDefaultClient()* dans le cas où le client est renseigné.

IBODocumentVenteLigne3

Ligne de document de vente.

Interface héritée

Syntaxe	Description
IBODocumentPartLigne3	Cf. Interface IBODocumentPartLigne3 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	AC_RefClient() As String	Référence article client.
Lecture / Ecriture	ArticleCompose() As IBOArticle3	Article composé auquel correspond la ligne s'il s'agit d'une ligne d'articles composants.
Lecture seule	Client() As IBOClient3	Client.
Lecture seule	DL_DateDE() As String	Date de devis.
Lecture seule	DL_DatePL() As Date	Date de préparation de livraison.
Lecture seule	DL_PieceDE() As String	Numéro de pièce de devis.
Lecture seule	DL_PiecePL() As String	N° de pièce de la Préparation de Livraison.
Lecture seule	DL_QteDE() As Double	Quantité du devis.

Accès	Syntaxe	Description
Lecture / Ecriture	DL_QtePL() As Double	Quantité livrée.
Lecture/ Ecriture	DL_QteRessource() As Long	Quantité ressource.
Lecture / Ecriture	DL_NoColis() As String	Numéro de colis.
Lecture seule	DocumentVente() As IBODocumentVente3	Document de vente auquel est associée la ligne de document.
Lecture seule	FactoryInfoComplement () As IBOFactoryInfoComplement	Fabrique un objet Informations complémentaires.
Lecture seule	IsComposeDeComposant() As Boolean	L'article de la ligne correspond à un composé de composant (True) ou non (False).
Lecture seule	IsComposeDeCompose() As Boolean	L'article de la ligne correspond à un composé de composé (True) ou non (False).
Lecture seule	LignesComposants() As IBICollection	Retourne une collection de lignes de composants si l'article de la ligne de document est un article composé.
Lecture / Ecriture	TTC() As Boolean	Ligne valorisée en TTC (True) ou en HT (False).
Lecture seule	Valorisation() As IDocLigneValorisation	Valorisation de la ligne.

Méthodes

Syntaxe	Description
SetDefaultArticleReferenceClient(ByVal sRef As String, ByVal Qte As Double)	Initialise les propriétés en fonction de la référence client et de la quantité.
SetDefaultRemise()	Initialise la propriété remise.

Traitements et initialisations par défaut

Méthode SetDefaultArticleReferenceClient(ByVal sRef As String, ByVal Qte As Double)

La méthode executée sera fonction de la valeur affectée au paramètre *sRef* :

- *sRef* = Référence client

  Méthode exécutée : SetDefaultArticle()(cf. IBODocumentLigne3)

- *sRef* = Référence de gamme ou code barre d'un énuméré de gamme

  Méthode exécutée : SetDefaultArticleMonoGamme() / SetDefaultArticleDoubleGamme() (cf. IBODocumentLigne3)

Méthode SetDefaultRemise()

L'appel de cette méthode initialise la propriété *remise* en fonction des paramètres de remise définis pour le client et pour l'article.

IBODocumentMedia

Documents média associés aux documents.

Interface héritée

Syntaxe	Description
IBIMedia	Cf. Interface de base des fichiers média.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Document() As IBODocument3	Document associé aux documents média.
Lecture / Ecriture	IsATransmettre() As Boolean	Document média à transmettre.

IBODocumentReglement

L'objet IBODocumentReglement permet d'ajouter, de modifier ou de supprimer des règlements de clients ou de fournisseurs et sur des documents de vente et achat persistants uniquement (les documents dit « mémoire » ne sont pas gérés). Les règlements de caisse ou de ticket ne peuvent pas être ajoutés, modifiés ou supprimés mais juste être lus (ils sont retournés par les différentes méthodes Query du factory).

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	TiersPayeur As IBOTiersPart3	Client ou fournisseur du règlement. Ce champ peut être affecté en création uniquement.
Lecture / Ecriture	RG_Date() As Date	Date du règlement.
Lecture / Ecriture	RG_Reference() As String	Référence du règlement.
Lecture / Ecriture	RG_Libelle() As String	Libellé du règlement.
Lecture / Ecriture	RG_Montant() As Double	Montant du règlement.
Lecture / Ecriture	RG_MontantDev() As Double	Montant du règlement en devise.
Lecture / Ecriture	Reglement() As IPBReglement	Mode de règlement (tous les modes sont acceptés à l'exception de ceux du type bon d'achat).
Lecture seule	RG_Impute() As Boolean	Règlement totalement imputé.

Accès	Syntaxe	Description
Lecture seule	RG_Compta() As Boolean	Règlement comptabilisé.
Lecture / Ecriture	RG_Cours() As Double	Cours de la devise.
Lecture / Ecriture	Devise() As IBPDevise2	Devise du règlement.
Lecture / Ecriture	Journal() As IBOJournal3	Journal du règlement (seuls les journaux de type trésorerie ou général sont acceptés).
Lecture / Ecriture	CompteGContrepartie() As IBOCompteG3	Compte général contrepartie (seuls les comptes généraux de type détail sont acceptés).
Lecture / Ecriture	RG_Impaye() As Date	Date d'impayé.
Lecture / Ecriture	CompteG() As IBOCompteG3	Compte général du règlement (seuls les comptes généraux de type détail sont acceptés).
Lecture / Ecriture	RG_Piece() As String	Numéro de pièce du règlement.
Lecture seule	HasCaisse() As Boolean	Indique si le règlement a une caisse rattachée.
Lecture seule	RG_Banque() As Boolean	Mise en banque.
Lecture seule	RG_Cloture() As Boolean	Règlement clôturé.
Lecture seule	RG_Ticket() As Boolean	Indique si le règlement a un ticket rattaché.
Lecture seule	TiersPayeurOrigine() As IBOTiersPart3	Client ou fournisseur du règlement (origine).
Lecture / Ecriture	RG_DateEchCont() As Date	Date échéance contrepartie.
Lecture / Ecriture	CompteGEcart() As IBOCompteG3	Compte général de l'écart (seuls les comptes généraux de type détail sont acceptés et le montant d'écart doit être différent de 0).
Lecture / Ecriture	JournalEcart() As IBOJournal3	Journal d'écart (seuls les journaux de type général sont acceptés et le montant d'écart doit être différent de 0) .
Lecture / Ecriture	RG_MontantEcart() As Double	Montant de l'écart (le montant d'écart peut être spécifié que si le règlement est associé à au moins une échéance de document).
Lecture seule	RG_Valide() As Boolean	Règlement validé.
Lecture seule	RG_MontantCommission() As Double	Montant de la commission.
Lecture seule	RG_MontantNet() As Double	Montant net.

Traitements et initialisations par défaut

Méthode SetDefault()

Si le document est en devise et que DR_MontantDev est null, alors affectation de DR_MontantDev en fonction de DR_Montant et du cours de la devise du document.

Si le document est en devise et que DR_Montant est null, alors affectation de DR_Montant en fonction de DR_MontantDev et du cours de la devise du document.

Ajout / Modification / Suppression d'un règlement

Ajout d'un règlement

Pour pouvoir ajouter un règlement, il est nécessaire que :

- la date soit renseignée
- la date ne soit pas incluse dans une période clôturée
- le tiers payeur soit renseigné et ne soit pas du type prospect
- le montant doit être différent de 0
- le compte général soit spécifié
- le journal soit spécifié

A l'affectation du tiers payeur (mode ajout uniquement), les valeurs suivantes sont modifiées :

- le type du règlement (client ou fournisseur)
- le compte général de contrepartie si défini dans les paramètres société
- le tiers d'origine
- la devise du règlement et le cours de change
- le compte général du règlement si défini au niveau de la fiche Tiers
- le mode de règlement
- le journal du règlement basé sur les paramètres société et le mode de règlement
- le numéro de pièce du règlement

Remarque : Les prospects ne sont pas acceptés.

Modification d'un règlement

La modification d'un règlement n'est pas autorisée si :

- le règlement est clôturé
- le règlement est rattaché à une caisse
- le règlement est rattaché à un ticket ou a au moins une échéance de type ticket
- le règlement est associé à un bon d'achat
- une propriété du règlement non autorisée en modification a été modifié (du fait de son statut validé ou comptabilisé)

Nom de la propriété	Modifiable avant validation	Modifiable après validation	Modifiable après comptabilisation
TiersPayeur	Non	Non	Non
RG_Date	Oui	Non	Non
RG_Reference	Oui	Oui	Non
RG_Libelle	Oui	Oui	Non
RG_Montant	Oui	Non	Non
RG_MontantDev	Oui	Non	Non
Reglement	Oui	Oui	Non
RG_Cours	Oui	Non	Non
Devise	Oui	Non	Non
Journal	Oui	Oui	Non
CompteGContrepartie	Oui	Oui	Non
RG_Impaye	Oui	Oui	Non
CompteG	Oui	Non	Non
RG_Piece	Non	Non	Non
RG_DateEchCont	Oui	Oui	Non
CompteGEcart	Oui	Oui	Oui
JournalEcart	Oui	Oui	Oui
RG_MontantEcart	Oui	Oui	Oui

Suppression d’un règlement

La suppression d’un règlement n’est pas autorisée si :

- le règlement est validé
- le règlement est clôturé
- le règlement est comptabilisé
- le règlement est rattaché à une caisse
- le règlement est rattaché à un ticket ou a au moins une échéance de type ticket
- le règlement est associé à un bon d’achat

IBOFamille3

Famille.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Catalogue() As IBPProduit2	Catalogue.
Lecture / Ecriture	FA_CodeFamille() As String	Code famille.
Lecture / Ecriture	FA_CodeFiscal() As String	Code fiscal.
Lecture / Ecriture	FA_Coef() As Double	Coefficient.
Lecture / Ecriture	FA_Contremarque() As Boolean	Article en contremarque.
Lecture / Ecriture	FA_Criticite() As FamilleCriticiteType	Niveau de criticité.
Lecture / Ecriture	FA_Delai() As Short	Délai livraison.
Lecture / Ecriture	FA_Escompte() As Boolean	Non soumis à l'escompte (True) ou soumis à l'escompte (False).
Lecture / Ecriture	FA_FactForfait() As Boolean	Facturation forfaitaire (True) ou facturation normale (False).
Lecture / Ecriture	FA_FactPoids() As Boolean	Facturation sur le poids (True) ou facturation normale (False).
Lecture / Ecriture	FA_Fictif() As Boolean	Gérer en tant que fictif.
Lecture / Ecriture	FA_Garantie() As Short	Garantie (en mois).
Lecture / Ecriture	FA_HorsStat() As Boolean	Hors statistiques (True) ou pris en compte dans les statistiques (False).
Lecture / Ecriture	FA_Intitule() As String	Intitulé.
Lecture / Ecriture	FA_Nature() As FamNatureType	Type de nature de la famille (Cf. énumérateur FamNatureType).
Lecture / Ecriture	FA_NbColis() As Short	Nombre de colis.
Lecture / Ecriture	FA_NotImp() As Boolean	Non impression document (True) ou impression document (False).
Lecture / Ecriture	FA_Pays() As String	Pays.
Lecture / Ecriture	FA_Publie() As Boolean	Publié sur le site marchand (True) ou non publié sur le site marchand (False).
Lecture / Ecriture	FA_Stat(ByVal *sElt* As Short) As String	Enuméré correspond au N ieme champ statistique.
Lecture / Ecriture	FA_SousTraitance() As Boolean	Réserver en sous-traitance.

Accès	Syntaxe	Description
Lecture / Ecriture	FA_SuiviStock() As SuiviStockType	Mode de suivi de stock (Cf. énumérateur SuiviStockType).
Lecture / Ecriture	FA_Type() As FamilleType	Type famille (Cf. énumérateur FamilleType).
Lecture / Ecriture	FA_UnitePoids() As UnitePoidsType	Unité de poids (Cf. Énumérateur UnitePoidsType).
Lecture / Ecriture	FA_VteDebit() As Boolean	Vente au débit (True) ou vente normale (False).
Lecture seule	FactoryFamille() As IBITypeObjectFactory	Fabrique un objet Famille.
Lecture seule	FactoryFamilleParamCompta() As IBITypeObjectFactory	Fabrique un objet Paramètre comptable famille.
Lecture seule	FactoryFamilleTarifCategorie() As IBITypeObjectFactory	Fabrique un objet tarif par catégorie.
Lecture seule	FactoryFamilleTarifClient() As IBITypeObjectFactory	Fabrique un objet tarif par client.
Lecture seule	FactoryFamilleTarifFournisseur() As IBITypeObjectFactory	Fabrique un objet tarif par fournisseur.
Lecture / Ecriture	FamilleCentral() As IBOFamille3	Famille centralisatrice.
Lecture / Ecriture	Frais() As Ifrais	Frais.
Lecture / Ecriture	ModeleAchat() As IBOModele2	Modèle d'enregistrement de type Achat.
Lecture / Ecriture	ModeleComptoir() As IBOModele2	Modèle d'enregistrement de type Vente comptoir.
Lecture / Ecriture	ModeleInterne() As IBOModele2	Modèle d'enregistrement de type Interne.
Lecture / Ecriture	ModeleStock() As IBOModele2	Modèle d'enregistrement de type Stock.
Lecture / Ecriture	ModeleTousDomaine() As IBOModele2	Modèle d'enregistrement pour tous les domaines.
Lecture / Ecriture	ModeleVente() As IBOModele2	Modèle d'enregistrement de type Vente.
Lecture / Ecriture	Produit() As IBPProduit	Catégorie de produit.
Lecture / Ecriture	Unite() As IBPUnite	Unité de vente.

Traitements et initialisations par défaut

Méthode SetDefault()

Si l'objet est non persistant, l'appel de la méthode *SetDefault()* initialise les propriétés suivantes :

Propriété	Valeur	Description
FA_Type	Si FA_Type <> FamilleTypeDetail Alors	Remet à zéro les champs inutilisés si la famille est d'un type différent de Détail.

Propriété	Valeur	Description
	Remise à zero des champs inutilisés	
FA_Intitule	Si FA_Intitule = "" Alors FA_Intitule = FA_CodeFamille	Si l'intitulé de la famille est vide, affectation du champ Code famille au champ Intitulé.

Si l'objet est persistant, l'appel de la méthode *SetDefault()* initialise les propriétés suivantes :

Propriété	Valeur	Description
FA_Intitule	Si FA_Intitule = "" Alors FA_Intitule = FA_CodeFamille	Si l'intitulé de la famille est vide, affectation du champ Code famille au champ Intitulé.

Méthode Write()

La méthode *Write()* enregistre l'objet dans la base de données après avoir effectué les traitements suivants :

- Si des énumérés statistiques ont été créés dans la famille, ils sont ajoutés dans la table F_ENUMSTATART ;
- Reprise dans la table F_FAMTARIF des Tarifs TTC de toutes les catégories tarifaires paramétrées dans Paramètres société.

Méthode WriteDefault()

La méthode *WriteDefault()* effectue les mêmes traitements que la méthode *Write()*.

IBOFamilleTarifClient

Tarifs famille par client.

Interface héritée

Syntaxe	Description
IBIFamilleTarifVente	Cf. Interface IBIFamilleTarifVente pour les propriétés et méthodes héritées.

Propriété

Accès	Syntaxe	Description
Lecture / Ecriture	Client() As IBOClient3	Client.

IBOFamilleTarifCategorie

Tarifs famille par catégorie.

Interface héritée

Syntaxe	Description
IBIFamilleTarifVente	Cf. Interface IBIFamilleTarifVente pour les propriétés et méthodes héritées.

Propriété

Accès	Syntaxe	Description
Lecture / Ecriture	CategorieTarif() As IBPCategorieTarif	Catégorie tarifaire.

IBOFamilleTarifQteCategorie

Gamme de remises de tarif famille par catégorie.

Interface héritée

Syntaxe	Description
IBIFamilleTarifVente	Cf. Interface IBIFamilleTarifVente pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	CategorieTarif() As IBPCategorieTarif	Catégorie tarifaire.
Lecture seule	FactoryFamilleTarifQteCategorie() As IBITypeObjectFactory	Fabrique un objet gamme de remise de tarif famille par catégorie.

IBOFamilleTarifFournisseur

Tarifs famille par fournisseur.

Interface héritée

Syntaxe	Description
IBIFamilleTarif	Cf. Interface IBIFamilleTarif pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	FF_Colisage() As Double	Colisage.
Lecture / Ecriture	FF_ConvDiv() As Double	Quantité unité d'achat.
Lecture / Ecriture	FF_Conversion() As Double	Quantité unité de vente.
Lecture / Ecriture	FF_DelaiAppro() As Integer	Délai d'approvisionnement.
Lecture / Ecriture	FF_Garantie() As Integer	Garantie.
Lecture / Ecriture	FF_QteMini() As Double	Quantité économique de commande.
Lecture / Ecriture	Fournisseur() As IBOFournisseur3	Fournisseur.
Lecture / Ecriture	Unite() As IBPUnite	Unité d'achat.

IBOFamilleTarifQteFournisseur

Gamme de remises de tarif famille par fournisseur.

Interface héritée

Syntaxe	Description
IBIFamilleTarifQte	Cf. Interface IBIFamilleTarifVente pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	FactoryFamilleTarifQteFournisseur() As IBITypeObjectFactory	Fabrique un objet gamme de remise de tarif famille par fournisseur.
Lecture seule	Fournisseur() As IBOFournisseur3	Fournisseur.

IBOFamilleParamCompta3

Paramétrage comptable famille.

Interface héritée

Syntaxe	Description
IBIParamCompta3	Cf. Interface IBIParamCompta3 pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	FactoryFamilleParamCompta() As IBITypeObjectFactory	Fabrique un objet Paramétrage comptable famille.
Lecture seule	Famille() As IBOFamille3	Famille à laquelle est associé le paramétrage comptable.

IBOGlossaire2

Glossaire.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	FactoryGlossaire() As IBITypeObjectFactory	Fabrique un objet Glossaire.
Lecture / Ecriture	GL_Domaine() As GlossaireDomaineType	Domaine (Cf. énumérateur GlossaireDomaineType)
Lecture / Ecriture	GL_Intitule() As String	Intitulé.
Lecture / Ecriture	GL_Langue1() As String	Langue 1.
Lecture / Ecriture	GL_Langue2() As String	Langue 2.
Lecture / Ecriture	GL_PeriodDeb() As Date	Début de période.
Lecture / Ecriture	GL_PeriodFin() As Date	Fin de période.
Lecture / Ecriture	GL_Raccourci() As String	Raccourci.
Lecture / Ecriture	GL_Text() As String	Texte du glossaire.

IBOModele2

Modèle d'enregistrement.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	FactoryModele() As IBITypeObjectFactory	Fabrique un objet Modèle d'enregistrement.
Lecture / Ecriture	MO_Calcul() As String	Mode de calcul.
Lecture / Ecriture	MO_Intitule() As String	Intitulé.

IBOCollaborateur

Collaborateur.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Acheteur() As Boolean	Acheteur (True) ou non (False).
Lecture / Ecriture	Adresse() As IAdresse	Adresse.
Lecture / Ecriture	Caissier() As Boolean	Caissier (True) ou non (False).
Lecture / Ecriture	ChargeRecouvrement() As Boolean	Chargé de recouvrement (True) ou Non (False).
Lecture seule	DateCreation() As Date	Date de création.
Lecture seule	FactoryCollaborateur() As IBOCollaborateur	Fabrique un objet Collaborateur.
Lecture / Ecriture	Fonction() As String	Fonction.
Lecture / Ecriture	Matricule() As String	Matricule du collaborateur.
Lecture / Ecriture	Nom() As String	Nom.
Lecture / Ecriture	Prenom() As String	Prénom.
Lecture / Ecriture	Receptionnaire() As Boolean	Collaborateur de type réceptionnaire (True) ou non (False).
Lecture / Ecriture	Service() As String	Service.
Lecture / Ecriture	Telecom() As ITelecom	Télécommunications.
Lecture / Ecriture	Utilisateur() As String	Utilisateur.
Lecture / Ecriture	Vendeur() As Boolean	Vendeur (True) ou non (False).
Lecture / Ecriture	Financier () As Boolean	Vendeur (True) ou non (False).
Lecture / Ecriture	Facebook() As String	Compte Facebook.

Accès	Syntaxe	Description
Lecture / Ecriture	LinkedIn() As String	Compte LinkedIn.
Lecture / Ecriture	Skype() As String	Compte Skype.
Lecture / écriture	ChefVentes () As boolean	Valeur de l'option Chef des ventes.
Lecture	FactoryVendeurs () As IBOVendeursAssociesFactory	Gestion de la liste des vendeurs associés.

IBOVendeursAssociesFactory

Gestion de la liste des vendeurs associés.

Interface héritée

Syntaxe	Description
IBITypeObjectFactory	Cf. Interface IBITypeObjectFactory pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Ecriture	AddVendeur(IBOCollaborateur)	Associe un vendeur à un chef des ventes.
Ecriture	RemoveVendeur (IBOCollaborateur)	Supprime un vendeur à un chef des ventes
Lecture	ExistVendeur (IBOCollaborateur)	Vérifie l'appartenance d'un vendeur à un chef des ventes
Lecture	List As Collection IBOCollaborateur	Retourne la liste des vendeurs associés

IBORessource

Ressource.

Interface héritée

Syntaxe	Description
IBIRessource	Cf. Interface IBIRessource pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Adresse() As IAdresse	Adresse.
Lecture seule	FactoryRessource() As IBORessourceFactory	Fabrique un objet ressource.
Lecture / Ecriture	RP_CodeExterne() As String	Code externe.
Lecture / Ecriture	Telecom() As ITelecom	Informations télécommunication.

Accès	Syntaxe	Description
Lecture seule	InfoLibre() As IBIValues	Valeur information libre.

IBORessourceCentre

Centre de charge.

Interface héritée

Syntaxe	Description
IBIRessource	Cf. Interface IBIRessource pour les propriétés et méthodes héritées.

Propriété

Accès	Syntaxe	Description
Lecture seule	FactoryRessourceCentre() As IBORessourceCentreFactory	Fabrique un objet centre de charges.

IBOInfoComplementClient

Information complémentaire Client.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture	Client() As IBOClient3	Fabrique objet maître Client.
Lecture	FactoryInfoComplement() As IBORessourceFactory	Fabrique un objet Info.
Lecture	CI_Code() As String	Code externe.
Lecture	CI_Intitule() As String	Intitulé.
Lecture	CI_Domaine () As InfoDomaine	Domaine de l'information client.
Lecture	CI_Type () As InfoType	Type de la valeur de l'information client.
Lecture / Ecriture	CI_Valeur() As String	valeur de l'information complémentaire.

IBOInfoComplementEntete

Information complémentaire Document Vente.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture	DocumentVente () As IBODocumentVente3	Fabrique objet maître Document.
Lecture	FactoryInfoComplement() As IBORessourceFactory	Fabrique un objet Info.
Lecture	DI_Code() As String	Code externe.
Lecture	DI_Intitule() As String	Intitulé.
Lecture	DI_Type () As InfoType	Type de la valeur de l'information.
Lecture / Ecriture	DI_Valeur() As String	valeur de l'information complémentaire.

IBOInfoComplementDocligne

Information complémentaire Ligne de document de vente.

Interface héritée

Syntaxe	Description
IBIPersistObject	Cf. Interface IBIPersistObject pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture	FactoryInfoComplement() As IBORessourceFactory	Fabrique un objet Info.
Lecture	DC_Code() As String	Code externe.
Lecture	DC_Intitule() As String	Intitulé.
Lecture	DC_Type () As InfoType	Type de la valeur de l'information.
Lecture / Ecriture	DC_Valeur() As String	valeur de l'information complémentaire.

Interfaces Processus (IFailxxx, IPMxxx)

IFailInfo

Erreur renvoyée par les processus.

Propriétés

Accès	Syntaxe	Description
Lecture seule	ErrorCode() As Variant	Code erreur.
Lecture seule	Indice() As Long	Indice de l'erreur dans la collection.
Lecture seule	Texte() As String	Texte de l'erreur.

IFailInfoCol

Collection d'erreurs renvoyées par les processus.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Count() As Long	Nombre d'erreurs dans la collection.
Lecture seule	Item(ByVal IIndex As Long) As IFailInfo	Retourne l'élément erreur pour l'indice passé en paramètre.

IPMProcess

Interface de base des processus.

Propriétés

Accès	Syntaxe	Description
Lecture seule	CanProcess() As Boolean	Teste si le processus peut être validé.
Lecture seule	Errors() As IFailInfoCol	Retourne la collection des erreurs.

Méthode

Syntaxe	Description
Process()	Validation du processus : Insertion des données en base.

IPMEncoder

Processus de création de pièce comptable. Ce processus est accessible depuis l'interface IBSCPTAApplication100c.

Description

Le processus de saisie de pièce comptable permet d'insérer une collection d'écritures comptables en une seule fois. Ceci permet, à la différence d'insertion d'écritures directement à partir d'objets *IBOEcriture3*, de garantir que les écritures insérées par le processus ne pourront pas générer de déséquilibre de la période, et qu'aucunes écritures insérées par une autre application ne pourront s'intercaler entre deux écritures du processus.

Ce processus publie également certaines méthodes permettant de générer automatiquement les écritures analytiques, les écritures d'échéances des comptes tiers ainsi que les écritures HT, TTC, TVA et solde (*AddTiersPart*, *Equilibrer*) en fonction de certains paramètres.

Syntaxe	Description
IPMProcess	Cf. Interface IPMProcess pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture/ Ecriture	bAnalytiqueAuto() As Boolean	Génération automatique des écritures analytiques.
Lecture/ Ecriture	bMultiEcheanceAuto() As Boolean	Génération automatique des échéances.
Lecture seule	Credit() As Double	Somme des montants au crédit pour le processus.
Lecture/ Ecriture	Date() As Date	Date de la pièce.
Lecture seule	Debit() As Double	Somme des montants au débit pour le processus.
Lecture/ Ecriture	Devise() As IBPDevise2	Devise.
Lecture/ Ecriture	EC_Intitule() As String	Intitulé.
Lecture/ Ecriture	Ec_Parite() As Double	Parité.
Lecture/ Ecriture	EC_Piece() As String	Numéro de pièce.
Lecture/ Ecriture	EC_Reference() As String	Référence pièce.
Lecture/ Ecriture	EC_RefPiece() As String	Numéro de facture.

Accès	Syntaxe	Description
Lecture seule	FactoryEcritureIn() As IBOEcritureFactory3	Fabrique un objet écriture pour le processus.
Lecture/ Ecriture	Journal() As IBOJournal3	Journal.
Lecture seule	ListEcritureOut() As IBICollection	Retourne la collection des écritures insérées dans la base.
Lecture seule	Solde() As Double	Solde des écritures du processus.

Méthodes

Syntaxe	Description
AddTiersPart(*pTiers* As IBOTiersPart3, ByVal *dMontantTTC* As Double, *pCompteGChargeProduit* As IBOCompteG3, *pCompteGTVA* As IBOCompteG3, *pCompteGTiers* As IBOCompteG3)	Génère automatiquement les écritures HT, TVA et TTC en fonction des valeurs passées en paramètres. Seuls les paramètres IBOTiersPart3 et dMontantTTC sont obligatoires.
Equilibrer(*pCompteG* As IBOCompteG3) As IBOEcriture3	Retourne une écriture d'équilibre pour le processus en fonction du compte général passé en paramètre.

Méthode AddTiersPart()

Pour la méthode *AddTiersPart()*, le signe du montant passé en paramètre à la méthode a la signification suivante :

- Dans un journal d'achat :
  - ◊ Un montant **positif** correspond à un **Avoir**
  - ◊ Un montant **négatif** correspond à une **Facture**
- Dans un journal de vente
  - ◊ Un montant **positif** correspond à une **Facture**
  - ◊ Un montant **négatif** correspond à un **Avoir**

Ainsi, quelque soit le type de journal (achat ou vente), si le montant passé en paramètre à la méthode *AddTiersPart()* est positif, alors ce montant sera mis au **Débit**. Si le montant passé en paramètre est négatif, alors celui-ci sera mis au **Crédit**.

IPMLettrer

Processus de lettrage d'écritures comptables. Ce processus est accessible depuis l'interface IBSCPTAApplication100c.

Description

Le processus de lettrage permet de lettre, pré-lettrer ou pointer une collection d'écritures comptables. Ce processus équivaut à la fonction ***Interrogation et lettrage*** de la comptabilité.

Interface héritée

Syntaxe	Description
IPMProcess	Cf. Interface IPMProcess pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Credit() As Double	Montant crédit de la collection d'écritures ajoutées au processus.
Lecture seule	CreditDevise() As Double	Montant crédit devise de la collection d'écritures ajoutées au processus.
Lecture seule	Debit() As Double	Montant débit de la collection d'écritures ajoutées au processus.
Lecture seule	DebitDevise() As Double	Montant débit en devise de la collection d'écritures ajoutées au processus.
Lecture / Ecriture	Lettre() As String	Lettre à affecter à la collection d'écritures ajoutées au processus.
Lecture seule	ListEcritures() As IBICollection	Collection d'écritures ajoutées au processus.
Lecture seule	Solde() As Double	Solde de la collection d'écritures ajoutées au processus.
Lecture seule	SoldeDevise() As Double	Solde devise de la collection d'écritures ajoutées au processus.
Lecture/ Ecriture	Type() As LettrageType	Type de lettrage.

Méthodes

Syntaxe	Description
AddEcriture(pObj As IBOEcriture3)	Ajoute une écriture comptable au processus.
RmvEcriture(pObj As IBOEcriture3)	Supprime une écriture comptable du processus.
SetDefaultLettre()	Initialise la lettre à utiliser en fonction du type de lettrage.

Contrôles de cohérence

La validation du processus ne peut être réalisée que lorsque les écritures ajoutées au processus respectent les contraintes suivantes :

- Les écritures doivent appartenir au même exercice.
- Les écritures doivent avoir le même compte général.
- Les écritures doivent avoir le même compte tiers. Si une écriture n'a pas de tiers, alors aucune écriture ajoutée au processus ne doit avoir de tiers. Dans le cas d'un pointage, les écritures doivent toutes avoir le même compte général ou le même compte tiers.
- Les écritures doivent être du type Normal (EC_Type = EcritureTypeNormal).
- La norme des écritures doit être différente de IFRS (EC_Norme <> EcritureNormeIFRS).
- Les écritures ne doivent pas être lettrées (non bloquant dans le cas du pointage).

Contrôles de cohérence par type de lettrage

Type de lettrage	Format lettre	Solde obligatoire	Devise des écritures	Contrôle lettrage
Pré-lettrage montant	3 car. Alphabétique minuscule	Non	-	EC_Lettre = ''
Lettrage montant	3 car. Alphabétique majuscule	Oui sur montant	-	EC_Lettre = ''
Pré-lettrage devise	3 car. Alphabétique minuscule	Non	Doit être identique pour toutes les écritures	EC_LettreQ = ''
Lettrage devise	3 car. Alphabétique majuscule	Oui sur montant devise	Doit être identique pour toutes les écritures	EC_LettreQ = ''
Pointage	3 car. Alphabétique minuscule ou majuscule	Non	-	-

IPMDocument

Processus de création de documents commerciaux. Ce processus est accessible depuis l'interface IBSCIALApplication100c.

Description

A l'instar du processus ***IPMEncoder*** qui permet d'écrire en base une pièce comptable en une seule fois, le processus de création de document permet quant à lui d'écrire en base un document complet (entête et lignes) en une seule fois. Si pour une quelconque raison l'entête ou une ligne de document ne peut pas être validée (état du stock insuffisant par exemple), alors le processus échouera et le document ne sera pas inséré en base.

L'utilisation de ce processus est simple de mise en œuvre puisqu'il s'appuie sur les mêmes interfaces que celles utilisées pour la création de documents et lignes de document : ***IBODocument3*** et ***IBODocumentLigne3***.

La différence se situe au niveau de la persistance des interfaces. En création standard de document, à l'appel des méthodes ***Write()*** ou ***WriteDefault()***, les données sont physiquement écrites en base, elles passent donc de l'état non persistant à l'état persistant. En utilisation du processus de création de document, l'appel de ces méthodes n'écrit pas les données en base. Tant que le processus n'est pas validé (***.Process()***), les interfaces manipulées (***IBODocument3*** et ***IBODocumentLigne3***) sont des interfaces mémoire. Ce n'est qu'après validation du processus, que le document et les lignes de documents seront réellement écrits en base, les interfaces passeront alors de l'état non persistant à l'état persistant.

Ce comportement permet notamment, d'éviter que pendant l'ajout de ligne sur le document, celui-ci soit vérouillé par un autre poste client (Application Sage 100c ou Développement Objets Métiers). En effet, le document étant non persistant tant que le processus n'est pas validé, il ne sera pas visible ni accessible par d'autres postes clients.

Le processus de création de document permet également de simplifier la gestion des articles suivis en nomenclature. Ainsi, à l'ajout d'une ligne pour un article géré en nomenclature, les lignes d'articles composant la nomenclature sont automatiquement ajoutées à la collection des lignes du document à créer.

Interface héritée

Syntaxe	Description
IPMProcess	Cf. Interface IPMProcess pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Document() As IBODocument3	Retourne un objet document non persistant.
Lecture seule	DocumentResult() As IBODocument3	Retourne un objet document persistant.

Méthodes

Syntaxe	Description
AddArticle(*pArt* As IBOArticle3, ByVal *DL_Qte* As Double) As IBODocumentLigne3	Ajoute une ligne de document au processus pour l'article et la quantité passés en paramètres.
AddArticleConditionnement(*pArtCond* As IBOArticleCond3, ByVal *dQte* As Double) As IBODocumentLigne3	Ajoute une ligne de document au processus pour le conditionnement article et la quantité passés en paramètres.
AddArticleDoubleGamme(*pEnum1* As IBOArticleGammeEnum3, *pEnum2* As	Ajoute une ligne de document au processus pour les énumérés de

Syntaxe	Description
IBOArticleGammeEnum3, ByVal *Qte* As Double) As IBODocumentLigne3	gamme et la quantité passes en paramètres.
AddArticleMonoGamme(*pEnum* As IBOArticleGammeEnum3, ByVal *Qte* As Double) As IBODocumentLigne3	Ajoute une ligne de document au processus pour l'énuméré de gamme et la quantité passés en paramètres.
AddArticleReference(ByVal *sRef* As String, ByVal *Qte* As Double) As IBODocumentLigne3	Ajoute une ligne de document au processus pour la reference article et la quantité passées en paramètres.

Lors de l'ajout d'une ligne pour un article géré en nomenclature, les lignes composants la nomenclature de l'article sont automatiquement ajoutées à la collection des lignes du processus.

IPMAppliquerBareme

Processus permettant d'appliquer les barèmes sur un document. Ce processus est accessible depuis l'interface IBSCIALApplication100c.

Description

Le processus d'application des barèmes permet d'ajouter automatiquement des lignes de remises à un document, en fonction des barèmes répondant aux critères d'application des barèmes. Ce processus équivaut à la commande « Barèmes » disponible en entête de document sous la Gestion commerciale.

La mise en œuvre de ce processus nécessite uniquement, d'affecter un document persistant au processus. Après validation du processus, les lignes de remises générées pourront être retrouvées en parcourant la collection des lignes du document affecté en entrée du processus.

Interface héritée

Syntaxe	Description
IPMProcess	Cf. Interface IPMProcess pour les propriétés et méthodes héritées.

Propriété

Accès	Syntaxe	Description
Lecture / Ecriture	Document() As IBODocumentPart3	Document (achat, vente ou interne) sur lequel la fonction d'application des barèmes doit être exécutée.

IPMControleQualite

Processus de contrôle qualité. Ce processus est accessible depuis l'interface IBSCIALApplication100c.

Description

Le processus de contrôle qualité permet de réaliser un traitement identique à la fonction « Traitement \ Contrôle qualité » de la Gestion commerciale Sage 100c. Ainsi, il permet de valider, retourner ou mettre au rebus des quantités pour des articles placés sur des emplacements de contrôle.

Pour ce faire, le processus permet dans un premier temps, de récupérer une collection de lignes d'emplacement (*IBODocumentLigneEmplacement*) en fonction de différents critères :

- Période,
- Domaine,
- Contrôleur,
- Dépôt.

Ensuite, pour chacune des lignes d'emplacement de la collection, il sera nécessaire d'appeler les méthodes *Valider(LigneEmpl As IBODocumentLigneEmplacement, Qte As Double)*, *Retourner(LigneEmpl As IBODocumentLigneEmplacement, Qte As Double,sMotifRetour As String)* ou *MettreRebut(LigneEmpl As IBODocumentLigneEmplacement, Qte As Double,sMotifRebut As String*) pour respectivement, valider, retourner ou mettre au rébut la ligne d'emplacement pour la quantité passée en paramètre.

A la validation du processus, suivant les méthodes appelées, différents types de document peuvent être générés :

- Valider : génère des mouvements de transfert
- Retourner : génère des bons ou factures de retour (dépend de la valeur affectée à la propriété *TypeDocumentRetour*)
- MettreRebut : génère des mouvements de sortie.

Interface héritée

Syntaxe	Description
IPMProcess	Cf. Interface IPMProcess pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	Control(LigneEmpl As IBODocument) As IMPControleQualiteInfo	Retourne les informations de contrôle qualité de la ligne d'emplacement passée en paramètre.
Lecture / Ecriture	Controleur() As IBOCollaborateur	Contrôleur sur lequel les lignes d'emplacements seront sélectionnées par le processus.

Accès	Syntaxe	Description
Lecture / Ecriture	DateValidation() As Date	Date des documents générés par le processus.
Lecture / Ecriture	Depot() As IBODepot3	Dépôt sur lequel les lignes d'emplacements seront sélectionnées par le processus.
Lecture / Ecriture	DO_RefOut() As String	Référence reportée sur tous les documents générés par le processus.
Lecture / Ecriture	Domaine() As DomaineType	Domaine sur lequel les lignes d'emplacements seront sélectionnées par le processus (cf. énumérateur DomaineType).
Lecture seule	ListMouvementSortieOut() As IBICollection	Collection des mouvements de sortie de stock générés par le processus.
Lecture seule	ListMouvementTransfertOut() As IBICollection	Collection des mouvements de transfert générés par le processus.
Lecture seule	ListRetourOut() As IBICollection	Collection des bons de retour/factures de retour fournisseur pour le domaine Achat ou Bons de livraison/factures client pour le domaine Vente, générés par le processus.
Lecture / Ecriture	PeriodDO_Date() As IDateTimePeriod	Période de date des documents sur laquelle les lignes d'emplacements seront sélectionnées par le processus.
Lecture / Ecriture	PeriodDO_DateLivraison() As IDateTimePeriod	Période de date de livraison des documents sur laquelle les lignes d'emplacements seront sélectionnées par le processus.
Lecture / Ecriture	TypeDocumentRetour() As DocumentType	Type de document à générer pour le retour de marchandise. Bon de retour fournisseur ou facture de retour fournisseur pour le domaine Achat. Bon de livraison client ou facture client pour le domaine Vente.

Méthodes

Syntaxe	Description
MettreRebut(*LigneEmpl* As IBODocumentLigneEmplacement, ByVal *Qte* As Double, ByVal *sMotifRebut* As String) As IPMControlerQualiteInfo	Génère un mouvement de sortie de stock pour la ligne d'emplacement, la quantité et le motif passés en paramètres.
QueryLigneEmplacement() As IBICollection	Retourne la collection de lignes d'emplacements satisfaisants les critères du processus.
RemoveControl(*LigneEmpl* As IBODocumentLigneEmplacement)	Supprime le contrôle qualité saisi sur la ligne d'emplacement passée en paramètre.

Syntaxe	Description
Retourner(*LigneEmpl* As IBODocumentLigneEmplacement, ByVal *Qte* As Double, ByVal *sMotifRetour* As String) As IPMControlerQualiteInfo	Génère un bon de retour/facture de retour fournisseur pour le domaine Achat ou un bon de livraison/facture client pour le domaine Vente, pour la ligne d'emplacement, la quantité et le motif passés en paramètres.
Valider(*LigneEmpl* As IBODocumentLigneEmplacement, ByVal *Qte* As Double) As IPMControlerQualiteInfo	Génère un mouvement de transfert pour la ligne d'emplacement et la quantité passées en paramètres.

IPMControleQualiteInfo

Informations de contrôle qualité sur une ligne d'emplacement. Cette interface est implémentée dans le cadre de l'utilisation du processus de contrôle qualité : **IPMControleQualite**.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	DepotEmplDest() As IBODepotEmplacement	Emplacement de destination.
Lecture / Ecriture	DL_QteRebut() As Double	Quantité rebut.
Lecture / Ecriture	DL_QteRetour() As Double	Quatité retour.
Lecture / Ecriture	DL_QteValidee() As Double	Quantité validée.
Lecture seule	DocLigneEmpl() As IBODocumentLigneEmplacement	Ligne d'emplacement.
Lecture / Ecriture	EU_QteRebut() As Double	Quantité colisée rebut.
Lecture / Ecriture	EU_QteRetour() As Double	Quantité colisée retour.
Lecture / Ecriture	EU_QteValidee() As Double	Quantité colisée validée.
Lecture / Ecriture	MotifRebut() As String	Motif rebut.
Lecture / Ecriture	MotifRetour() As String	Motif retour.

IPMColiser

Processus de colisage. Ce processus est accessible depuis l'interface IBSCIALApplication100c.

Description

Le processus de colisage permet de décomposer une ligne de préparation de livraison en plusieurs lignes de préparation de livraison. Cette décomposition s'effectue en fonction des numéros de colis et quantités qui seront affectés à la ligne.

Ce processus équivaut au traitement d'affectation des numéros de colis dans la fonction de « Validation des préparations de livraison clients » de la Gestion commerciale.

Ce processus prend en entrée une ligne de document de type préparation de livraison. Après validation du processus (***.Process()***), la ligne d'entrée sera décomposée en n lignes, où n représente le nombre de colis qui auront été attribués.

Dans le cas où les numéros de colis ne sont pas attribués pour toute la quantité de la ligne, une ligne sans numéro de colis et pour la quantité restante à coliser (***QteRestantAColiser***) sera générée.

Interface héritée

Syntaxe	Description
IPMProcess	Cf. Interface IPMProcess pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	LigneOrigine() As IBODocumentVenteLigne3	Ligne de préparation de livraison à décomposer en plusieurs colis.
Lecture seule	ListLignesResult() As IBICollection	Collection des lignes résultantes de la décomposition de la ligne d'origine en plusieurs colis.
Lecture seule	QteAColiser() As Double	Quantité à coliser.
Lecture seule	QteRestanteAColiser() As Double	Quantité restante à coliser.
Lecture seule	UserColis() AS IBICollectionDispatch	Permet d'accéder, d'ajouter ou supprimer des colis (IUserColis) à la ligne.

IPMDocTransferer

Processus de transfert d'un article d'un dépôt/emplacement vers un autre. Ce processus est accessible depuis l'interface IBSCIALApplication100c.

Description

Le processus de transfert permet de transférer un article, d'un dépôt vers un autre, et/ou d'un emplacement vers un autre. Ce processus ne peut être appliqué que sur les documents de stock de type **Mouvement de transfert**. Après validation du processus (***.Process()***), les lignes de sortie et d'entrée sont automatiquement générées, en fonction, des dépôts d'origine et de destination du document, de l'article et des emplacements qui lui seront affectés.

Interface héritée

Syntaxe	Description
IPMProcess	Cf. Interface IPMProcess pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	DepotEmplDest() As IBODepotEmplacement	Emplacement de destination à appliquer à la ligne d'entrée de stock.
Lecture / Ecriture	DepotEmplOrigine() As IBODepotEmplacement	Emplacement d'origine à appliquer à la ligne de sortie de stock.
Lecture / Ecriture	Document() AsIBODocumentStock3	Document de stock de type « Mouvement de transfert » sur lequel doit s'appliquer le processus.
Lecture seule	ListLignesResult() As IBICollection	Collection de lignes résultantes du transfert de l'article.
Lecture seule	NoSerie() As String	Numéro de série/lot de l'article.
Lecture seule	UserEmplacementOrigineToUse() As IBICollectionDispatch	Collection des emplacements d'origine (IUserEmplacement).
Lecture seule	UserLotsQteRestantAFournir() As Double	Quantité série/lot nécessaire pour le transfert de l'article.
Lecture seule	UserLotsToUse() As IBICollectionDispatch	Permet d'accéder, d'ajouter ou supprimer des série/lot (IUserLot) pour l'article à transférer.
Lecture / Ecriture	ComplementSerie (sName As String)	Permet de retrouver en **priorité** les lots suivant un complément donné. Agit comme un filtre.
Lecture / Ecriture	Transactional (bLock As Boolean)	Option qui permet de passer le processus en mode transaction sécurisée

Méthodes

Syntaxe	Description
SetDefaultArticle(*pArticle* As IBOArticle3, ByVal *QteTransfert* As Double)	Affecte l'article et la quantité au processus.
SetDefaultArticleDoubleGamme(*pEnum1* As IBOArticleGammeEnum3, *pEnum2* As IBOArticleGammeEnum3, ByVal *QteTransfert* As Double)	Affecte l'article double gamme et la quantité au processus.
SetDefaultArticleMonoGamme(*pEnum* As IBOArticleGammeEnum3, ByVal *QteTransfert* As Double)	Affecte l'article mono gamme et la quantité au processus.
SetDefaultArticleNumeroSerie(*pArticle* As IBOArticle3, ByVal *QteTransfert* As Double, ByVal *sNumSerie* As String, *pDepotEmpl* As IBODepotEmplacement)	Affecte l'article, la quantité, le numéro série/lot et l'emplacement d'origine au processus.

Après l'affectation d'un article au processus (SetDefaultArticle()), les emplacements d'origine et de destination (DepotEmplOrigine et DepotEmplDest) sont automatiquement initialisés (fonctionnement identique à la gestion commerciale). De plus, pour les articles gérés en série/lot, la collection **UserLotsToUse** est également automatiquement intitialisée avec des série/lot équivalents à l'affectation « Automatique » de la gestion commerciale. Enfin, pour les articles gérés en série/lot, la propriété **DepotEmplOrigine** ne doit pas être renseignée puisque ce sont les série/lot affectés au processus qui déterminent les emplacements de sortie.

IPMDocTransformer

Processus de transformation de documents de vente et achat. Ce processus est accessible depuis les propriétés **Transformation.Vente** et **Transformation.Achat** de l'interface **IBSCIALApplication100c**.

Description

Un processus de transformation permet de transformer des documents de vente ou d'achat vers un type de document supérieur dans le même domaine. Ce processus équivaut aux fonctions de transformation de documents de la Gestion commerciale.

Le processus de transformation publie cinq types de transformation, trois sur les documents de vente et deux sur les documents d'achat.

Transformation des documents de vente

- Processus **Commander** : permet de transformer des documents de vente de type devis (documents complets ou lignes de documents) dans un ou plusieurs bons de commande existants ou à créer.
- Processus **Livrer** : permet de transformer des documents de vente de type devis, bon de commande et préparation de livraison (documents complets ou lignes de documents) dans un ou plusieurs bons de livraison existants ou à créér.
- Processus **Facturer** : permet de transformer des documents de vente de type devis, bon de commande, préparation de livraison et bon de livraison (documents complets ou lignes de documents) dans une ou plusieurs factures existantes ou à créér.

Transformation des documents d'achat

- Processus **Commander** : permet de transformer des documents d'achat de type préparation de commande (documents complets ou lignes de documents) dans un ou plusieurs bons de commande existants ou à créér.
- Processus **Réceptionner** : permet de transformer des documents d'achat de type préparation de commande et bon de commande (documents complets ou lignes de documents) dans un ou plusieurs bons de livraison existants ou à créér.

Paramètres d'entrée/sortie

Un processus de transformation prend en entrée :

- 1 à n lignes de documents
- 1 à n documents
- 0 à n documents de destinations (documents dans lesquels seront mises les lignes transformées)

Si aucun document de destination n'est spécifié, le processus le créera automatiquement.

Un processus de transformation fournit en sortie :

- 1 à n documents résultats
- 1 à n lignes de documents résultats

La transformation de lignes d'articles présentant une indisponibilité en stock ne génère pas d'erreur d'exécution du processus.Ainsi, dans le cas d'une indisponibilité partielle, le processus transformera la quantité disponible et laissera un reliquat dans le document d'origine. Dans le cas où la totalité de la quantité est indisponible, la ligne ne sera pas transformée et elle sera conservée dans le document d'origine.Il existe toutefois une particularité pour les articles suivis en série/lot, pour lesquels, lorsque le numéro série/lot associé à la ligne est indisponible, le processus de transformation génère une exception et aucune ligne du processus ne sera alors transformée.

Interface héritée

Syntaxe	Description
IPMProcess	Cf. Interface IPMProcess pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	DO_Date() As Date	Date du document généré par la transformation.
Lecture seule	ListDocumentsATransformer() As IBICollection	Collection des documents à transformer.
Lecture seule	ListDocumentsDestination() As IBICollection	Collection des documents de destination.
Lecture seule	ListDocumentsResult() As IBICollection	Collection des documents générés par la transformation.

Accès	Syntaxe	Description
Lecture seule	ListLignesResult() As IBICollection	Collection des lignes de documents générées par la transformation.
Lecture / Ecriture	MajCoursType) As TransfoMajCoursType	Type de mise à jour du cours de la devise durant la transformation.
Lecture / Ecriture	MajTauxTaxe() As Boolean	Mise à jour des taux de taxe durant la transformation.
Lecture / Ecriture	RecalculFraisExpedition() As Boolean	Conditionne le recalcul des frais d'expéditions durant la transformation.
Lecture seule	UserLotsQteRestantAFournir(pLigne As IBODocumentLigne3) As Double	Pour une ligne de document sur un article suivi en série/lot, retourne la quantité série/lot nécessaire pour la transformation de la ligne.
Lecture seule	UserLotsToUse(pLigne As IBODocumentLigne3) As IBICollectionDispatch	Pour une ligne de document sur un article suivi en série/lot, permet d'accéder, d'ajouter ou supprimer des éléments série/lot (IUserLot).

Méthodes

Syntaxe	Description
AddDocument(pDoc As IBODocument3)	Ajoute un document à la liste des documents à transformer.
AddDocumentDestination(pDoc As IBODocument3)	Ajoute un document à la liste des documents de destination. Le document passé en paramètre doit être un document persistant.
AddDocumentLigne(pDoc As IBODocumentLigne3)	Ajoute une ligne de document à la liste des lignes à transformer.
RmvDocument(pDoc As IBODocument3)	Supprime l'entête de document de la liste des documents à transformer. Attention : la suppression de l'entête ne supprime pas les lignes du documents présentes dans la collection ListLignesATransformer().
RmvDocumentDestination(pDoc As IBODocument3)	Supprime un document de la liste des documents à transformer.
RmvDocumentLigne(pDoc As IBODocumentLigne3)	Supprime une ligne de la liste des lignes de document à transformer.

Entités manipulées

Durant la procédure de transformation de documents, les entités suivantes sont manipulées :

- Entêtes de documents
- Lignes de documents

- Stocks
- Lots/séries des articles
- Acomptes et échéances des documents

Dans le cas d'une transformation de n documents dans un seul document, le regroupement ne pourra être effectué, comme sous la gestion commerciale, que si les 6 champs suivants sont identiques :

- Tiers,
- 1 BL/Facture,
- Catégorie comptable,
- Devise,
- Cours de la devise,
- Souche.

De plus, les critères de fusion ne sont par défaut pas activés. Ainsi, le regroupement de document ne sera pas réalisé si au moins un des champs suivants diffère entre les documents à transformer :

- Caissier [1]
- Caisse [1]
- Dépôt
- Dépôt client [1]
- Expédition [1]
- Condition de livraison [1]
- Périodicité [1]
- Tarif (inclut Centrale d'achat) [1]
- Nb de facture
- Transaction
- Régime
- Provenance
- Langue
- Affaire

- Payeur/Encaisseur
- N° pièce site [1]
- Colisage [1]
- Taux d'escompte

[1] : Documents des ventes uniquement.

Concernant le calcul des dates d'échéances, il s'appuie sur le paramétrage défini sous la gestion commerciale (*Paramètres société* \ *Documents* \ *Général* \ *Figer la date d'échéance à partir du*).

Gestion des emplacements

Durant les processus de transformation, les emplacements sont gérés automatiquement par les processus. Ainsi, si l'emplacement est renseigné sur la ligne de document d'origine, ce sera cette information qui sera utilisée pour transformer la ligne. Dans le cas où l'emplacement n'est pas renseigné sur la ligne d'origine, il sera alors automatiquement affecté en s'appuyant sur le paramétrage défini dans la base commerciale.

Gestion des articles Série/Lot

Comme pour le cas de la gestion des emplacements, les processus de transformation utiliseront le numéro de Série/Lot défini sur la ligne de document d'origine. Toutefois, pour les lignes de document pour lesquelles le numéro série/lot n'est pas renseigné, il sera possible d'affecter manuellement les numéros de Série/Lot à utiliser durant le processus en implémentant la propriété ***UserLotsToUse***.

Dans le cas des transformations sur les documents de vente, si le numéro de Série/Lot n'est pas renseigné, il sera alors automatiquement affecté (équivalent à la commande ***Automatique*** de la fenêtre Série/Lot de la gestion commerciale).

Pour les transformations des documents d'achat, il faudra **impérativement** renseigner le numéro de Série/Lot à utiliser pour que la transformation puisse aboutir.

Regroupement de lignes

Pour le processus des achats **Commander**, un regroupement de lignes est automatiquement réalisé lorsque plusieurs lignes de document répondent aux conditions suivantes :

- Même référence article,
- Même dépôt,
- Même code affaire,
- Même collaborateur,

- Aucun lien contremarque.

> Le regroupement ne s'effectue que pour des lignes présentes dans le processus. Si un document de destination contenant une ligne répondant aux conditions de regroupement est affecté au processus, cette ligne ne sera pas utilisée dans le regroupement. De plus, lorsque l'option de gestion Multi-emplacement est activée sur la base, seuls les articles non suivis en stock ou suivi au CMUP seront regroupés.

IPMPreleverLot

Processus de prélèvement des série/lot. Ce processus est accessible depuis l'interface IBSCIALApplication100c.

Description

Le processus de prélèvement des série/lot permet de décomposer une ligne de préparation de livraison sur un article suivi en série/lot, en plusieurs lignes de préparation de livraison. Cette décomposition s'effectue en fonction des numéros de série/lot et quantités affectés à la ligne.

Ce processus équivaut au traitement d'affectation des numéros de série/lot dans la fonction de « Validation des préparations de livraison clients » de la Gestion commerciale.

Ce processus prend en entrée une ligne de préparation de livraison. Après l'affectation d'une ligne de préparation de livraison au processus (affectation de ***LigneOrigine***), la collection des série/lot (***UserLots****)* sera automatiquement alimentée avec des série/lot équivalents au calcul *Automatique* de la Gestion commerciale.

Pour affecter des série/lot différents de ceux proposés par défaut, il sera donc nécessaire de préalablement supprimer les éléments non souhaités de la collection des série/lot (cf. ***Remove()*** et ***RemoveAll()*** de ***IBICollectionDispatch***), pour ensuite ajouter les série/lot souhaités.

De plus, si l'affectation des série/lot n'est pas réalisée pour l'intégralité de la quantité à répartir (***QteARepartir***), à la validation du processus, une ligne sans série/lot et pour une quantité correspondant à la quantité restante à répartir (***QteRestantARepartir***) sera générée.

> Les propriétés QteARepartir et QteRestanteARepartir ne tiennent pas compte des quantités liées en contremarque puisque celles-ci sont déjà associées à des série/lot.

Interface héritée

Syntaxe	Description
IPMProcess	Cf. Interface IPMProcess pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	LigneOrigine() As IBODocumentVenteLigne3	Ligne de préparation de livraison à décomposer en plusieurs série/lot.
Lecture seule	ListLignesResult() As IBICollection	Collection des lignes résultantes de la décomposition de la ligne d'origine en plusieurs série/lot.
Lecture seule	QteARepartir() As Double	Quantité à répartir en série/lot.
Lecture seule	QteRestanteARepartir() As Double	Quantité restante à répartir en série/lot.
Lecture seule	UserLots() AS IBICollectionDispatch	Permet d'accéder, d'ajouter ou supprimer des éléments série/lot (IUserLot) à la ligne.
Lecture / Ecriture	Transactional (bLock As Boolean)	Option qui permet de passer le processus en mode transaction sécurisée, Vraie par défaut.

IPMDocInsererSousTotal

Processus de création de ligne de sous total. Ce processus est accessible depuis l'interface IBSCIALApplication100c.

Description

Ce processus permet d'insérer une ligne de sous total dans un document commercial (persistant ou non persistant). Ce processus équivaut à la fonction ***Insérer un sous-total*** disponible après sélection de lignes dans un document commercial.

Interface héritée

Syntaxe	Description
IPMProcess	Cf. Interface IPMProcess pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Commentaire() As String	Commentaire à affecter à la ligne de sous total.
Lecture seule	LigneTotal() As IBODocumentPartLigne3	Ligne de résultat du processus.

Méthodes

Syntaxe	Description
AddDocumentLigne(*pLigne* As IBODocumentPartLigne3)	Ajout d'une ligne de document à la collection des lignes à totaliser.
RmvDocumentLigne(*pLigne* As IBODocumentPartLigne3)	Supprime une ligne de document de la collection des lignes à totaliser.

IPMDocRecalculPrixCompose

Processus de recalcul de prix de revient d'une nomenclature fabrication. Ce processus est accessible depuis l'interface IBSCIALApplication100c.

Description

Ce processus permet de recalculer le prix de revient des nomenclatures de type fabication contenues dans un bon de fabrication. Ce processus équivaut à la fonction ***Recalculer le PR du composé*** disponible dans un bon de fabrication de Sage 100c Gestion commerciale.

Interface héritée

Syntaxe	Description
IPMProcess	Cf. Interface IPMProcess pour les propriétés et méthodes héritées.

Méthodes

Syntaxe	Description
AddDocumentLigne(*pLigne* As IBODocumentLigne3)	Ajout d'une ligne de document à la collection des lignes à utiliser pour le recalcul.
RmvDocumentLigne(*pLigne* As IBODocumentLigne3)	Supprime une ligne de document de la collection des lignes à utiliser pour le recalcul.

Dans le cas où le processus est initialisé sur un document, mais qu'aucune ligne de document n'est ajoutée à la collection (***.AddDocumentLigne***), à la validation du processus (***.Process***), **c'est l'intégralité des lignes du document qui sera utilisée pour procéder au recalcul**.

IPMSortirLots

Processus de gestion d'une sortie pour un article géré par lot. Ce processus est accessible depuis l'interface IBSCIALApplication100c.

Description

Ce processus permet, pour une ligne de document **non persistante** dans un document **persistant**, de créer autant de lignes que nécessaire pour sortir un lot en fonction d'une quantité. Ce processus équivaut à la fonction ***Automatique*** de la fenêtre de ***Gestion des lots*** affichée lors de la validation d'une ligne de document dans la Gestion commerciale. Il permet de s'abstaire de la recherche des entrées en stock d'un article géré par lot. En effet, c'est le processus, suivant les paramètres passés en entrée, qui se chargera d'affecter automatiquement le ou les lots aux lignes de sortie de stock.

Interface héritée

Syntaxe	Description
IPMProcess	Cf. Interface IPMProcess pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	DepotEmpl() As IBODepotEmplacement	Emplacement d'origine à appliquer à la ligne de sortie de stock.
Lecture seule	LigneOrigine() As IBODocumentLigne3	Ligne d'origine à décomposer en plusieurs lots.
Lecture seule	ListLignesResult() As IBICollection	Collection des lignes générées par la validation du processus.
Lecture seule	NoSerie() As String	Numéro de série/lot.
Lecture seule	QteARepartir() As Double	Quantité à répartir.
Lecture seule	UserEmplacementOrigineToUse() As IBICollection	Permet d'accéder, ajouter ou supprimer les emplacements des lots d'entrée pour la ligne de sortie de stock.
Lecture seule	UserLots() As IBICollection	Collection des éléments lots pour la ligne à décomposer.
Lecture / Ecriture	ComplementSerie (sName As String)	Permet de retrouver en **priorité** les lots suivant un complément donné. Agit comme un filtre.
Lecture / Ecriture	Transactional (bLock As Boolean)	Option qui permet de passer le processus en mode transaction sécurisée

Méthode

Syntaxe	Description
SetLigneDefaultLot(*pLigne* As IBODocumentLigne3, ByVal *sNumSerie* As String, *pDepotEmpl* As IBODepotEmplacement)	Ajout d'une ligne au processus pour le numéro de lot et l'emplacement passés en paramètres. Les paramètres *sNumSerie* et *pDepotEmpl* sont optionnels.

Ce processus ne peut être utilisé que sur les lignes de documents de type **sortie en quantité** qui ne sont **pas persistantes**. L'entête de document quant à lui doit être **persistant**. Le processus IPMSortirLots() ne peut donc pas être utilisé dans le processus de création de document : IPMDocument().

IPMConversionClient

Processus de transformation d'un prospect en client. Ce processus est accessible depuis l'interface IBSCIALApplication100c.

Description

Ce processus permet de transformer un prospect en client. Il équivaut à la fonction « **Transformer** » disponible sur la fiche d'un prospect dans Sage 100c Gestion commerciale.

Interface héritée

Syntaxe	Description
IPMProcess	Cf. Interface IPMProcess pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture Ecriture	Classement() As String	Classement.
Lecture seule	Client() As IBOClient3	Prospect transformé en client.
Lecture Ecriture	Intitule() As String	Intitulé.
Lecture Ecriture	NumPayeur() As IBOClient3	Compte client payeur.
Lecture Ecriture	NumPrinc() As IBOCompteG3	Compte général principal.
Lecture Ecriture	Prospect() As IBOClient3	Prospect à transformer en client.

IPMReglerEcheances

Processus de transformation d'un prospect en client. Ce processus est accessible depuis l'interface IBSCIALApplication100c.
Si la fonction « Loi Anti-Fraude est activée », il faut obligatoirement que la ***facture soit validée*** *par l'application avant de pouvoir régler l'échéance, car la validation nécessite une impression pdf de la facture..*

Description

Ce processus permet le règlement d'échéances.
En paramètre il doit avoir un règlement existant.
Méthode AddDocumentEcheance qui associe une échéance d'un document.
Méthode AddDocumentEcheanceMontant qui associe une échéance avec un montant spécifique

Interface héritée

Syntaxe	Description
IPMProcess	Cf. Interface IPMProcess pour les propriétés et méthodes héritées.

Propriétés

Accès	Syntaxe	Description
Lecture seule	ListLignesResult() As IBICollection	Collection résultante des échéances qui ont été réglées
Lecture Ecriture	Reglement() As IBODocumentReglement	Règlement à imputer
Lecture Ecriture	AddDocumentEcheance () As IBODocumentEcheance3	Ajout des échéances que l'on veut régler.
Lecture Ecriture	AddDocumentEcheanceMontant () As IBODocumentEcheance3, double	Echéance que l'on veut régler pour un montant spécifique

IPMInventaire

Description

Ce processus permet de gérer l'inventaire pour des articles, un dépôt, un emplacement (facultatif) et une date donnée.

Interface héritée

Syntaxe	Description
IPMProcess	Cf. Interface IPMProcess pour les propriétés et méthodes héritées.

Classe d'application IBSCIALApplication100c

Ce processus est accessible depuis l'interface IBSCIALApplication100c :

CreateProcess_Inventaire() As IPMInventaire

Propriétés

Accès	Syntaxe	Description
Lecture seule	ListDocuments() As IBODocumentStock3	Retourne le résultat : les documents de stock générés
Lecture Ecriture	DateInventaire() As Date	Précise la date de l'inventaire (date du jour si absente)
Lecture Ecriture	Depot() As IBODepot3	Précise le dépôt de l'inventaire (Dépôt principal si absent)

Méthodes

Syntaxe	Description
AddArticle(IBOArticle3, *Qte* As Double, Prix As Double, pDepotEmpl As IBODepotEmplacement (null possible))	Ajoute un article, quantité et Prix unitaire à l'inventaire (sur un emplacement si défini)
AddArticleMonoGamme(IBOArticle3, *Qte* As Double, Prix As Double, IBOArticleGammeEnum3, IBODepotEmplacement* pDepotEmpl, pDepotEmpl As IBODepotEmplacement (null possible))	Ajoute un article, quantité, Prix unitaire pour un énuméré de gamme à l'inventaire (sur un emplacement si défini)
AddArticleDoubleGamme(IBOArticle3, *Qte* As Double, PrixU As Double, IBOArticleGammeEnum3, IBOArticleGammeEnum3, pDepotEmpl As IBODepotEmplacement (null possible))	Ajoute un article, quantité, Prix unitaire pour 2 énumérés de gamme à l'inventaire (sur un emplacement si défini)
AddArticleSerie(IBOArticle3, Qte As Double, Prix As Double, IBOArticleDepotLot, pDepotEmpl As IBODepotEmplacement (null possible))	Ajoute un article, quantité, Prix unitaire pour un lot/Serie à l'inventaire (sur un emplacement si défini)

Le paramètre pDepotEmpl As IBODepotEmplacement peut être null. En ce cas si le multi-emplacement est géré, l'emplacement par défaut de l'article ou de la gamme sera utilisé.

Attention à utiliser les bonnes fonctions :
- Un article à une gamme doit être ajouté par la fonction AddArticleMonoGamme.
- Un article à double Gamme : AddArticleDoubleGamme
- Un article Série/lot : AddArticleSerie
- Les articles au CMUP ou LIFO/FIFO : AddArticle

Résultat : 2 documents de mouvement de stock sont générés : Un mouvement de sortie et un mouvement d'entrée pour réajuster le stock des articles en quantité et prix demandés.

IPMBonAPayer

Description

Ce processus permet d'affecter un bon à payer à une écriture.
Il doit être decliné en CreateProcess_BAPValider, CreateProcess_BAPAttendre, CreateProcess_BAPPayer, CreateProcess_BAPConformer ou CreateProcess_BAPRejeter.

Interface héritée

Syntaxe	Description
IPMProcess	Cf. Interface IPMProcess pour les propriétés et méthodes héritées.

Ajout des Propriétés

Accès		Syntaxe	Description
Ecriture		Ecriture (As IBOEcriture3)	Permet d'affecter l'écriture sur laquelle le Bon à payer sera affecté
Lecture / Ecriture		Complement (As String)	Complément du Bon à payer.

```
private void TestBonAPayer(ref BSCPTAApplication100c BaseCpta)
    try
    {
        IBOCollaborateur cCollabo =
BaseCpta.FactoryCollaborateur.ReadNomPrenom("PLATY", "Nelly");
        cCollabo.Financier = true;
        cCollabo.Write();

        IBPBonAPayer bonAPayer = (IBPBonAPayer) BaseCpta.FactoryBonAPayer.List[1];
        bonAPayer.Responsable = cCollabo;
        bonAPayer.Autorisation = eBAPAutorisationType.BAP_Financier;
        bonAPayer.Facture = eBAPAValider.BAP_SelonMontant;
        bonAPayer.SeuilValidation = 999;
        bonAPayer.Write();

        short EC_No = 65;

        if (BaseCpta.FactoryEcriture.ExistNumero(EC_No))
        {
            IBOEcriture3 cEcrit = BaseCpta.FactoryEcriture.ReadNumero(EC_No);
            IPMBonAPayer pBonAPayer = BaseCpta.CreateProcess_BAPPayer(cCollabo);
            pBonAPayer.Commentaire = "Ok pour Payer";
            pBonAPayer.Ecriture = cEcrit;
            pBonAPayer.Process();

            IBOEcriture3 cEcrit2 = BaseCpta.FactoryEcriture.ReadNumero(EC_No);
            IBonAPayer BonAPayer = cEcrit2.HistoriqueBonAPayer;
            MessageBox.Show(BonAPayer.Collaborateur.Nom +" "+ BonAPayer.Commentaire,
"Bon à Payer");
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Resultat Errors");
    }
```

Les énumérateurs

AgendaType

Type d'agenda.

Syntaxe	Description
AgendaDocument	Document.
AgendaFabrication	Fabrication.
AgendaInteresse	Intéressé.

AgendaTypeInteresse

Type d’événement intéressé.

Syntaxe	Description
AgendaTypeInteresseClient	Evénement client.
AgendaTypeInteresseCollaborateur	Evénement collaborateur.
AgendaTypeInteresseDepot	Evénement Dépôt.
AgendaTypeInteresseFournisseur	Evénement fournisseur.
AgendaTypeInteresseRessource	Evénement ressource.

AnalytiqueRuptureType

Type de rupture pour le plan analytique IFRS.

Syntaxe	Description
AnalytiqueRuptureTypeAucun	Aucun.
AnalytiqueRuptureTypeSecteurActivite	Secteur d’activité.
AnalytiqueRuptureTypeZoneGeographique	Zone géographique.

ArrondiType

Type d'arrondi.

Syntaxe	Description
ArrondiTypeFinInf	Fin inférieur.
ArrondiTypeFinProche	Fin proche.
ArrondiTypeFinSup	Fin supérieur.

Syntaxe	Description
ArrondiTypeInf	Inférieur.
ArrondiTypeProche	Proche.
ArrondiTypeSup	Supérieur.

ArticleType

Type d'article.

Syntaxe	Description
ArticleTypeGamme	Type gamme.
ArticleTypeRessourceUnitaire	Type ressource unitaire.
ArticleTypeRessourceMultiple	Type ressource multiple.
ArticleTypeStandard	Type standard.

BanqueFormatVirement

Format virement.

Syntaxe	Description
BanqueFormatVirement320S	Format 320S.
BanqueFormatVirementAFB	Format AFB.
BanqueFormatVirementSEPA	Format SEPA.

BanqueModeRemise

Mode de remise en banque.

Syntaxe	Description
BanqueModeRemiseFichier	Fichier magnétique.
BanqueModeRemiseInternet	Internet.
BanqueModeRemiseMessagerie	Messagerie.
BanqueModeRemisePapier	Papier.
BanqueModeRemiseTeleTrans	Télétransmission.

BanqueRemise

Donneur d'ordres type de remise.

Syntaxe	Description
BanqueRemiseMonoDevMonoDate	Mono devise / Mono date d'exécution.
BanqueRemiseMonoDevMultiDate	Mono devise / Multi date d'exécution.
BanqueRemiseMultiDevMonoDate	Multi devise / Mono date d'exécution.
BanqueRemiseMultiDevMonoDate	Multi devise / Multi date d'exécution.

BaremeCommissionInteresse

Barême de commissionnement.

Syntaxe	Description
BaremeCommissionInteresseRepresentant	Commissionnement représentant.
BarameCommissionInteresseGlobal	Commissionnement global.

BaremeCommissionObjectif

Objectif de commissionnement.

Syntaxe	Description
BaremeCommissionObjectifQuantite	Objectif sur quantité.
BaremeCommissionObjectifValeur	Objectif sur valeur.
BaremeCommissionObjectifTauxRemise	Objectif sur taux de remise.

BaremeCommissionDomaine

Domaine de commissionnement.

Syntaxe	Description
BaremeCommissionDomaineCommande	Domaine commande.
BaremeCommissionDomainePreparation	Domaine préparation.
BaremeCommissionDomaineLivraison	Domaine livraison.
BaremeCommissionDomaineFacturation	Domaine facturation.
BaremeCommissionDomaineEncaissement	Domaine encaissement.

BaremeCommissionBase

Base de commissionnement.

Syntaxe	Description
BaremeCommissionBaseCaHT	Commissionnement sur le C.A. HT.
BaremeCommissionBasemargeHT	Commissionnement sur la marge.

BaremeCommissionCalcul

Méthode de calcul du commissionnement.

Syntaxe	Description
BaremeCommissionCalculTranche	Calcul par tranche.
BaremeCommissionCalculGlobal	Calcul global.

BaremeRabaisObjectif

Objectif rabais, remises et ristournes.

Syntaxe	Description
BaremeRabaisObjectifQuantite	Objectif sur quantité.
BaremeRabaisObjectifValeur	Objectif sur valeur.

BaremeRabaisCalcul

Méthode de calcul du rabais, remises et ristournes.

Syntaxe	Description
BaremeRabaisCalculTranche	Calcul par tranche.
BaremeRabaisCalculGlobal	Calcul global.

BaremeRabaisType

Type de rabais, remises et ristournes.

Syntaxe	Description
BaremeRabaisTypeClient	Type client.
BaremeRabaisTypeFournisseur	Type fournisseur.

BaremeSoldeType

Type de soldes et promotions.

Syntaxe	Description
BaremeSoldeTypeClient	Type client.
BaremeSoldeTypeFournisseur	Type fournisseur.

ClientEncoursCtrlType

Type de contrôle encours client.

Syntaxe	Description
ClientEncoursCtrlTypeAuto	Contrôle automatique.
ClientEncoursCtrlTypeBloque	Compte bloqué.
ClientEncoursCtrlTypeCodeRisque	Selon code risque.

ComposantType

Type de composant de nomenclature.

Syntaxe	Description
ComposantTypeFixe	Fixe.
ComposantTypeVariable	Variable.

CompteADomaineType

Type de domaine de compte analytique.

Syntaxe	Description
CompteADomaineTypeAchat	Achat.
CompteADomaineTypeVente	Vente.
CompteADomaineTypeVenteAchat	Vente et achat.

CompteAType

Type de compte analytique.

Syntaxe	Description
CompteATypeDetail	Détail.

Syntaxe	Description
CompteATypeTotal	Total.

CompteAModeFacturationType

Type de mode de facturation des affaires.

Syntaxe	Description
CompteAModeFacturationForfait	Facturation forfaitaire.
CompteAModeFacturationAvancement	Facturation à l'avancement.

CompteAStatutType

Type de statut des sections analytiques.

Syntaxe	Description
CompteAStatutProposition	Proposition.
CompteAStatutAccepte	Accepté.
CompteAStatutPerdu	Perdu.
CompteAStatutEnCours	En cours.
CompteAStatutAttente	En attente.
CompteAStatutTermine	Terminé.

CompteGReportType

Type de report-à-nouveau de compte général.

Syntaxe	Description
CompteGReportTypeAucun	Aucun.
CompteGReportTypeDetail	Détail.
CompteGReportTypeSolde	Solde.

CompteGType

Type de compte général.

Syntaxe	Description
CompteGTypeDetail	Détail.
CompteGTypeTotal	Total.

CompteRType

Type de compte reporting.

Syntaxe	Description
CompteRTypeDetail	Détail.
CompteRTypeTotal	Total.

ContactCivilite

Civilité contact.

Syntaxe	Description
ContactCiviliteMonsieur	M.
ContactCiviliteMadame	Mme.
ContactCiviliteMademoiselle	Mlle.

CycleType

Cycle de vie des articles.

Syntaxe	Description
CycleTypeLancement	Lancement.
CycleTypeMaturite	Maturité.
CycleTypeDeclin	Déclin.

DatabaseType

Type de base de données.

Syntaxe	Description
DatabaseTypeNone	Typede base indéterminé.
DatabaseTypeSageSql	Type de base SageSQL.
DatabaseTypeSqlServer	Type de base SQL Server.

DepotEmplZone

Zone emplacement.

Syntaxe	Description
DepotEmplZoneAucune	Aucune.
DepotEmplZoneA	Zone A.

DepotEmplZoneB	Zone B.
DepotEmplZoneC	Zone C.

DepotEmplType

Type d'emplacement.

Syntaxe	Description
DepotEmplTypeStandard	Emplacement standard.
DepotEmplTypeTransit	Emplacement de transit.
DepotEmplTypeControle	Emplacement de contrôle.

DeviseMode

Mode de cotation de devise.

Syntaxe	Description
DeviseModeCertain	Certain.
DeviseModeIncertain	Incertain.

DeviseRemise

Code de remise.

Syntaxe	Description
DeviseRemiseAucun	Aucun.
DeviseRemiseBlanc	Blanc.
DeviseRemiseEuro	Euro.
DeviseRemiseFranc	Franc.

DocumentConditionPaiementType

Type des conditions de paiement sur les documents.

Syntaxe	Description
DocumentConditionPaiementTiers	Condition de paiement du tiers.
DocumentConditionPaiementSaisie	Condition de paiement saisie.

DocumentConditionPaiementModele	Condition de paiement du modèle.

DocumentFraisType

Type des frais d'expédition du document.

Syntaxe	Description
DocFraisTypeForfait	Montant forfaitaire.
DocFraisTypePoidsBrut	Quantité.
DocFraisTypePoidsNet	Poids net.
DocFraisTypeQuantite	Poids brut.

DocumentInterneMvtType

Type de mouvement interne.

Syntaxe	Description
DocumentInterneMvtTypeAucun	Aucun.
DocumentInterneMvtTypeEntreeQte	Entrée en quantité.
DocumentInterneMvtTypeSortieQte	Sortie en quantité.
DocumentInterneMvtTypeGainFinancier	Gain financier.
DocumentInterneMvtTypePerteFinancier	Perte financière.
DocumentInterneMvtTypeRealise	Réalisé.

DocumentLigneMvtStockType

Type des frais d'expédition du document.

Syntaxe	Description
DocLigneMvtStockTypeAucun	Aucun mouvement de stock.
DocLigneMvtStockTypeEntreeQte	Mouvement d'entrée en quantité.
DocLigneMvtStockTypeEntreeValeur	Mouvement d'entrée en valeur.
DocLigneMvtStockTypeSortieQte	Mouvement de sortie en quantité.
DocLigneMvtStockTypeSortieVal	Mouvement de sortie en valeur.

DocumentLigneProvenanceType

Type de provenance de la ligne de document.

Syntaxe	Description
DocLigneProvenanceNormale	Provenance Normale.
DocLigneProvenanceRetour	Provenance Retour.
DocLigneProvenanceAcompte	Provenance Facture d'Acompte
DocLigneProvenanceAvoir	Provenance Avoir.

DocumentProvenanceType

Type de la provenance du document.

Syntaxe	Description
DocProvenanceAvoir	Provenance Avoir.
DocProvenanceNormale	Provenance Normale.
DocProvenanceAcompte	Provenance Facture d'Acompte
DocProvenanceRetour	Provenance Retour.
DocProvenanceTicket	Provenance Ticket.

DocumentStatutType

Type de statut de document.

Syntaxe	Description
DocumentStatutTypeAPrepare	A Préparer.
DocumentStatutTypeConfirme	Confirmé.
DocumentStatutTypeSaisie	Saisie.
DocumentStatutTypeDevisPerdu	Devis perdu
DocumentStatutTypeDevisArchive	Devis archivé

DocumentType

Type de document.

Syntaxe	Description
DocumentTypeAchatArchive	Document d'achat archivé.
DocumentTypeAchatAvoir	Document d'achat : Bon d'avoir financier.
DocumentTypeAchatCommande	Document d'achat : Préparation de commande fournisseur.

Syntaxe	Description
DocumentTypeAchat CommandeConf	Document d'achat : Bon de commande fournisseur.
DocumentTypeAchatDemande	Document d’achat : Demande d’achat.
DocumentTypeAchatFacture	Document d'achat : Facture.
DocumentTypeAchatFactureCpta	Document d'achat : Facture comptabilisée.
DocumentTypeAchatLivraison	Document d'achat : Bon de livraison.
DocumentTypeAchatReprise	Document d'achat : Bon de retour.
DocumentTypeInterne1	Document interne 1.
DocumentTypeInterne2	Document interne 2.
DocumentTypeInterne3	Document interne 3.
DocumentTypeInterne4	Document interne 4.
DocumentTypeInterne5	Document interne 5.
DocumentTypeInterne6	Document interne 6.
DocumentTypeInterne7	Saisie du réalisé.
DocumentTypeInterneArchive	Document interne archivé.
DocumentTypeStockArchive	Document de stock archivé.
DocumentTypeStockDeprec	Document de stock : Dépréciation de stock.
DocumentTypeStockFabrication	Document de stock : Bon de fabrication.
DocumentTypeStockMouvIn	Document de stock : Mouvement d'entrée.
DocumentTypeStockMouvOut	Document de stock : Mouvement de sortie.
DocumentTypeStockOrdreFabrication	Document de stock : Ordre de fabrication.
DocumentTypeStockPreparation	Document de stock : Préparation de fabrication.
DocumentTypeStockVirement	Document de stock : Virement de dépôt à dépôt.
DocumentTypeTicket	Ticket de caisse.
DocumentTypeVenteArchive	Document de vente archivé.
DocumentTypeVenteAvoir	Document de vente : Bon d'avoir financier.
DocumentTypeVenteCommande	Document de vente : Bon de commande.
DocumentTypeVenteDevis	Document de vente : Devis
DocumentTypeVenteFacture	Document de vente : Facture.
DocumentTypeVenteFactureCpta	Document de vente : Facture comptabilisée.
DocumentTypeVenteLivraison	Document de vente : Bon de livraison.
DocumentTypeVentePrepaLivraison	Document de vente : Péparation de livraison.
DocumentTypeVenteReprise	Document de vente : Bon de retour.

DomaineType

Type de domaine.

Syntaxe	Description
DomaineTypeAchat	Achat.
DomaineTypeInterne	Document interne.
DomaineTypeStock	Stock.
DomaineTypeTicket	Ticket de caisse.
DomaineTypeVente	Vente.

DossierParamEmplacementPriorite

Type de priorité de déstockage.

Syntaxe	Description
EmplacementPrioriteNonPrincipal	Emplacement non principal.
EmplacementPrioritePrincipal	Emplacement principal.
EmplacementPrioriteZone	Zone emplacement.

EcritureANType

Type d'écriture d'à-nouveau.

Syntaxe	Description
EcritureANTypeDetail	Détail.
EcritureANTypeManuel	Manuel.
EcritureANTypeNormal	Normal.
EcritureANTypeResultat	Résultat.
EcritureANTypeSolde	Solde.

EcritureExpertType

Type d'écriture expert-comptable

Syntaxe	Description
EcritureExpertTypeEmis	Emis.
EcritureExpertTypeNEmis	Non émis.
EcritureExpertTypeReception	Réception.

EcritureImpressionType

Type impression écriture.

Syntaxe	Description
EcritureImpressionTypeBrouillard	Brouillard.
EcritureImpressionTypeJournal	Journal.
EcritureImpressionTypeNImp	Non imprimé.

EcritureNormeType

Type impression écriture.

Syntaxe	Description
EcritureNormeIFRS	IFRS.
EcritureNormeLesDeux	IFRS et Nationale.
EcritureNormeNationale	Nationale.

EcritureSensType

Type de sens de l'écriture.

Syntaxe	Description
EcritureSensTypeCredit	Crédit.
EcritureSensTypeDebit	Débit.

EcritureType

Type d'écriture.

Syntaxe	Description
EcritureTypeCentral	Ecriture centralisée.
EcritureTypeNormal	Ecriture normale.

EcritureODType

Type d'écriture d’OD.

Syntaxe	Description
EcritureODTypeOD	Ecriture type OD.
EcritureODTypeReport	Ecriture type report.

EdiCodeType

Type de code EDI.

Syntaxe	Description
EdiCodeTypeGLN	Non défini.
EdiCodeTypeDUNS	Non défini.
EdiCodeTypeEDIFACT	Non défini.

FamilleCriticiteType

Type de criticité des familles.

Syntaxe	Description
FamilleCriticiteTypeMineur	Mineur.
FamilleCriticiteTypeMajeur	Majeur.
FamilleCriticiteTypeCritique	Critique.

FamilleNatureType

Type de nature des familles.

Syntaxe	Description
FamilleNatureTypeComposant	Composant.
FamilleNatureTypePieceDetachee	Pièce détachée.
FamilleNatureTypeProduitFini	Produit fini.
FamilleNatureTypeProduitSemiFini	Produit semi-fini.

FamilleType

Type de famille.

Syntaxe	Description
FamilleTypeCentral	Centralisatrice.
FamilleTypeDetail	Détail.
FamilleTypeTotal	Total.

FieldType

Type de champ.

Syntaxe	Description
FieldTypeByte	Champ réservé.
FieldTypeChaine	Champ réservé.
FieldTypeChar	Champ réservé.
FieldTypeConstante	Champ réservé.
FieldTypeCptStr	Champ réservé.
FieldTypeCStr	Texte.
FieldTypeDate	Date.
FieldTypeDouble	Valeur.
FieldTypeFloat	Champ réservé.
FieldTypeLDate	Date longue.
FieldTypeLong	Champ réservé.
FieldTypeMontant	Montant.
FieldTypeNumStr	Champ réservé.
FieldTypePath	Champ réservé.
FieldTypePStr	Champ réservé.
FieldTypePwdStr	Champ réservé.
FieldTypeShort	Champ réservé.
FieldTypeStruct	Champ réservé.
FieldTypeTable	Table.
FieldTypeTime	Champ réservé.
FieldTypeULong	Champ réservé.
FieldTypeUShort	Champ réservé.

GammeType

Type de gamme.

Syntaxe	Description
GammeTypeDivers	Divers.
GammeTypeMontant	Montant.
GammeTypePrixNet	Prix net.
GammeTypeQtes	Quantités.

GlossaireDomaineType

Type du domaine du glossaire.

Syntaxe	Description
GlossaireDomaineTypeArticle	Article.
GlossaireDomaineTypeDocument	Document.

InfoDomaine Type

Domaine de l'information complémentaire Client.

Syntaxe	Description
eCDT_CLIENT	Information complémentaire Client.
eCDT_ENTETE	Information complémentaire Document vente.
eCDT_LIGNE	Information complémentaire Document ligne.

InfoType Type

Type de la valeur de l'information complémentaire.

Syntaxe	Description
eCDT_TEXTE	Valeur de type Texte.
eCDT_DATE	Valeur de type Date.
eCDT_MONTANT	Valeur de type Montant.
eCDT_VALEUR	Valeur de type Double (4 décimales).

JournalNumPieceType

Type de numérotation de pièce du journal.

Syntaxe	Description
JournalNumPieceTypeContFile	Continue pour le fichier.
JournalNumPieceTypeContJrnl	Continue pour le journal.
JournalNumPieceTypeManuelle	Manuelle.
JournalNumPieceTypeMensuelle	Mensuelle.

JournalRapproType

Type de rapprochement du journal.

Syntaxe	Description
JournalRapproTypeAucun	Aucun
JournalRapproTypeContrepartie	Contrepartie.

JournalRapproTypeTresorerie	Trésorerie.

JournalType

Type de journal.

Syntaxe	Description
JournalTypeAchat	Achat.
JournalTypeGeneral	Général.
JournalTypeSituation	Situation.
JournalTypeTresorerie	Trésorerie.
JournalTypeVente	Vente.

LancementType

Type de lancement des articles.

Syntaxe	Description
LancementTypeSpécifique	Lancement spécifique.
LancementTypeStandard	Lancement standard.

LangueType

Type de langue.

Syntaxe	Description
LangueTypeAucune	Aucune.
LangueTypeLangue1	Langue 1.
LangueTypeLangue2	Langue 2.

LettrageType

Type de lettrage.

Syntaxe	Description
LettrageTypeLettrageDevise	Lettrage devise.
LettrageTypeLettrageMontant	Lettrage montant.
LettrageTypePointageMontant	Pointage montant.
LettrageTypePrelettrageDevise	Pré-lettrage devise.
LettrageTypePrelettrageMontant	Pré-lettrage montant.

ModeleGrilleType

Type de modèle de grille.

Syntaxe	Description
ModeleGrilleTypeAnalytique	Grille de type analytique.
ModeleGrilleTypeGeneral	Grille de type général.

NomenclatureType

Type de nomenclature.

Syntaxe	Description
NomenclatureTypeAucun	Aucun.
NomenclatureTypeComposant	Composant.
NomenclatureTypeCompose	Composé.
NomenclatureTypeFabrication	Fabrication.
NomenclatureTypeLies	Article lié.

PeriodClotureType

Type de clôture des périodes.

Syntaxe	Description
PeriodClotureNone	Non clôturée.
PeriodCloturePartial	Clôture partielle.
PeriodClotureTotal	Clôture totale.

ReglementConditionType

Type de condition de règlement.

Syntaxe	Description
ReglementConditionTypeJour	Jour(s) net(s).
ReglementCondtionTypeMois	Fin de mois.
ReglementConditionTypeMoisCivil	Fin de mois civil.

ReglementRepartitionType

Type de répartition de règlement.

Syntaxe	Description
ReglementRepartitionTypeEquilib	Equilibre.
ReglementRepartitionTypeMontant	Montant.
ReglementRepartitionTypePourcent	Pourcentage.

ReglementType

Type de règlement.

Syntaxe	Description
ReglementTypeAucun	Aucun.
ReglementTypeBonAchat	Bon d’achat.
ReglementTypeCarte	Carte.
ReglementTypeCheque	Chèque.
ReglementTypeEspece	Espèces.

RepartitionType

Type de répartition.

Syntaxe	Description
RepartitionTypePourcent	Répartition en pourcentage.
RepartitionTypeEquilib	Répartition d’équilibre.
RepartitionTypeMontant	Répartition en montant.

RemiseType

Type de remise.

Syntaxe	Description
RemiseTypeMontant	Montant.
RemiseTypePourcent	Pourcentage.
RemiseTypeUnite	Unité.

RibType

Type de RIB.

Syntaxe	Description
RibTypeAutre	Autre.
RibTypeBban	BBAN.
RibTypeIban	IBAN.
RibTypeLocal	Local.

RisqueType

Type risque (action).

Syntaxe	Description
RisqueTypeBloquer	A bloquer.
RisqueTypeLivrer	A livrer.
RisqueTypeSurveiller	A surveiller.

StructBanqueControleRib

Contrôle de la clé RIB.

Syntaxe	Description
StructBanqueControleRibAucun	Aucun.
StructBanqueControleRibBelgique	Belgique.
StructBanqueControleRibEspagne	Espagne.
StructBanqueControleRibFrance	France.
StructBanqueControleRibPortugal	Portugal.

StructBanqueFieldType

Type des champs de structure banque.

Syntaxe	Description
StructBanqueFieldTypeAlphanum	Alphanumérique.
StructBanqueFieldTypeNumeric	Numérique.

SuiviStockType

Type de suivi de stock.

Syntaxe	Description
SuiviStockTypeAucun	Aucun.
SuiviStockTypeCmup	CMUP.
SuiviStockTypeFifo	FIFO.
SuiviStockTypeLifo	LIFO.
SuiviStockTypeLot	Lot.
SuivistockTypeSerie	Série.

TaxeProvenanceType

Type provenance taxe.

Syntaxe	Description
TaxeProvenanceTypeDivers1	Divers 1.
TaxeProvenanceTypeDivers2	Divers 2.
TaxeProvenanceTypeDivers3	Divers 3.
TaxeProvenanceTypeDivers4	Divers 4.
TaxeProvenanceTypeDivers5	Divers 5.
TaxeProvenanceTypeExport	Export.
TaxeProvenanceTypeIntracom	Intracommunautaire.
TaxeProvenanceTypeNational	National.

TaxeSensType

Type sens de taxe.

Syntaxe	Description
TaxeSensTypeCollecte	Collecté.
TaxeSensTypeDeductible	Déductible.

TaxeTauxType

Type taux de taxe.

Syntaxe	Description
TaxeTauxTypeMontant	Montant.

TaxeTauxTypePourcent	Pourcentage.
TaxeTauxTypeQuantite	Quantité.

TaxeType

Type de taxe.

Syntaxe	Description
TaxeTypeAgraire	Agraire.
TaxeTypeCEE	CEE.
TaxeTypeIGIC	IGIC.
TaxeTypeIRPF	IRPF.
TaxeTypeSurtaxe	Surtaxe.
TaxeTypeTPHT	Taxe parafiscale sur le hors taxe.
TaxeTypeTPPoids	Taxe parafiscale sur le poids.
TaxeTypeTPTTC	Taxe parafiscale sur le TTC.
TaxeTypeTVADebit	TVA sur les débits.
TaxeTypeTVAEncaiss	TVA sur les encaissements.

TiersType

Type de tiers.

Syntaxe	Description
TiersTypeAutre	Autre.
TiersTypeClient	Client.
TiersTypeFournisseur	Fournisseur.
TiersTypeSalarie	Salarié.

TransfoMajCoursType

Type de mise à jour des cours des devises durant la transformation de documents.

Syntaxe	Description
TransfoMajCoursTypeAucune	Aucune.
TransfoMajCoursTypeDevEntete	Devise entête.
TransfoMajCoursTypeDevDossier	Devise dossier.
TransfoMajCoursTypeTenueEntete	Monnaie de tenue entête.
TransfoMajCoursTypeTenueDossier	Monnaie de tenue dossier.

TypeTiersNumerotation

Type de numérotation tiers.

Syntaxe	Description
TypeTiersNumerotationManuel	Numérotation manuelle.
TypeTiersNumerotationAuto	Numérotation automatique.
TypeTiersNumerotationRacine	Numérotation manuelle avec racine.

TypeTiersCompteType

Type de racine.

Syntaxe	Description
TypeTiersCompteTypeRadical	Type radical.
TypeTiersCompteTypeCompte	Type compte.

UnitePoidsType

Type unité de poids.

Syntaxe	Description
UnitePoidsTypeGramme	Gramme.
UnitePoidsTypeKilogramme	Kilogramme.
UnitePoidsTypeMilligramme	Milligramme.
UnitePoidsTypeQuintal	Quintal.
UnitePoidsTypeTonne	Tonne.

eTypeBAP

Statut Bon à payer.

Syntaxe	Description
Aucun	Aucun
AValider	A Valider
NonConforme	Non Conforme
Conforme	Conforme
EnAttente	En Attente
BonAPayer	Bon A Payer

eBAPAutorisationType

Niveau de validation.

Syntaxe	Description
BAP_TypeAucun	Aucun
BAP_Financier	Responsable financier uniquement
BAP_LesDeux	Acheteur et Responsable financier

eBAPAValider

Factures à valider.

Syntaxe	Description
BAP_ToutesFactures	Toutes les Factures
BAP_SelonMontant	Selon Montant

TypeLigneRegistre

Type de ligne de registre taxe.

Syntaxe	Description
TypeLigneRegistreAucune	Aucune
TypeLigneRegistreAcompte	Acompte

Nouveautés des versions

Nouveautés version 6.0

Modification de l'interface

L'interface a été modifiée (Publication des dates de modification et de création pour chaque Objet)
Ce qui implique une recompilation des développements existants .

Objets et factory Réglements

L'objet **IBODocumentReglement** permet d'ajouter, de modifier ou de supprimer des règlements clients ou fournisseurs de type RG_REGLEMENT sur des documents persistants.
IBODocumentReglementFactory : Méthodes de lecture et écriture des réglements
IBODocumentReglementEcheance : Objet Imputations entre réglements et échéances
IBODocumentReglementEcheanceReglementFactory : Méthodes lecture des Imputations de réglements
IBODocumentReglementEcheanceEcheanceFactory : Méthodes lecture des échéances réglées

Processus Régler des échéances

Le processus **IPMReglerEcheances** permet d'imputer les échéances à un règlement.

Voir Objets Réglements

Modification du processus de sortie d'article géré par lot

A partir de cette version, le processus **IPMSortirLots** permet de sortir tout type d'article.
L'article sera sorti en respectant son type de suivi de stock.
Le processus garantit la sortie unique des lots.

Nouveautés version 7.0

Nouvelles interfaces Objets et factory Infos complémentaires

Ces méthodes permettent de lire et modifier la valeur des informations complémentaires associées aux clients, aux entêtes de documents de vente et aux lignes de documents.

IBOInfoComplementClientFactory Méthode d'accés, accessible par le client
IBOInfoComplementClient Objet Informations complémentaires associé à un client
IBOInfoComplementEnteteFactory Méthode d'accés, accessible par le document de vente
IBOInfoComplementEntete Objet Informations complémentaires associé à un document de vente
IBOInfoComplementDocligneFactory Méthode d'accés, accessible par une ligne de document de vente
IBOInfoComplementDocligne Objet Informations complémentaires associé à une ligne de document de vente

Objets IBOArticleDepot3 et IBOArticleDepotGamme3

Les méthodes StockValeur et StockQte ont été optimisées pour les objets IBOARTICLEDEPOT et IBOArticleDepotGamme.

Objets IBOCollaborateur

Les méthodes **ChefVentes** et **FactoryVendeurs** ont été ajoutées pour gérer les chefs des ventes

Nouvelle interface factory IBOVendeursAssociesFactory

Ces méthodes permettent de lire et modifier les éléments de la liste des vendeurs associé à un chef des ventes.

Nouveautés version 7.20

Objet ILicence
Ajout de la méthode **Version ,** chaîne de caractères qui donne la version de la Dll ex : “7.10”

Objet IBODocligne3
- Ajout de la méthode **SetEmplacementEntree ,** Permet de saisir un emplacement d’entrée différent de l’emplacement par défaut.
- Ajout de la méthode **WriteCumulLot ,** Ecriture de la ligne avec la particularité de modifier la ligne pour cumuler le lot en entrée s’il existe déjà.

Création structure info libre pour les Objets :
IBODocumentFactory
IBODocumentLigneAllFactory
IBOTiersFactory3
IBOCompteGFactory3
IBOCompteAFactory3
IBOEcritureFactory3
IBOArticleFactory3
IBOArticleDepotLotAllFactory
IBIRessourceFactory
- Ajout de la méthode **CreateInfoLibre** (String Nom, eTypeInfoLibre Type, short lenght) Permet de créer une entrée dans la structure Infos Libre du fichier.

Objet IBOCollaborateur
Collaborateur. Ajout d’une propriété : Financier Booleen si vendeur ou non.

Objet IBPBonAPayer
Paramètres de comptabilisation – Gestion du Bon à payer (Paramètres société / Comptabilisation)

Nouveaux Process : IPMBonAPayer
CreateProcess_BAPValider
CreateProcess_BAPAttendre
CreateProcess_BAPPayer
CreateProcess_BAPConformer
CreateProcess_BAPRejeter
Ces processus permettent d’affecter un bon à payer à une écriture.

Process :

IPMDocTransferer

IPMSortirLots

Ajout de la propriété **ComplementSerie** qui Permet de retrouver en priorité les lots suivant un complément donné. Agit comme un filtre.

Nouveautés version 8.00

Modification de l'énuméré DocumentProvenanceType :

Suppression de la valeur DocProvenanceRectif
Ajout de la valeur DocProvenanceAcompte

Modification de l'énuméré DocumentLigneProvenanceType :

Suppression des valeurs
DocLigneProvenanceRectifVal
DocLigneProvenanceRectifQte
Ajout de la valeur DocLigneProvenanceAcompte

IBODepotFactory2

Ajout des propriétés
ExistCompteur (int DE_No) As boolean
ReadCompteur (int DE_No) As IBODepot3*

IBODepot3

Ajout de la propriété
Compteur () As int Compteur du dépôt

IBOArticleGammeEnumFactory2

Ajout des propriétés
ExistCompteur (int AG_No) As boolean
ReadCompteur (int AG_No) As IBOArticleGammeEnum3*

IBOArticleGammeEnum3

Ajout de la propriété
Compteur () As int Compteur de l'énuméré de gamme

Nouveautés version 9.00

Ajout de l'énuméré TypeLigneRegistre :

TypeLigneRegistreAucune
TypeLigneRegistreAcompte

IBODocumentLigneAllFactory

Ajout des propriétés
ExistLigne (int DL_No) As boolean
ReadLigne (int DL_No) As IBODocumentLigne3*

IBODocumentLigne3

Ajout des propriétés
DL_No () As int

IBOArticleGlossaireFactory2

Ajout des propriétés
RemoveGlossaire (ByVal *pGlossaire* As IBOGlossaire2)

IBODocumentAcompte3

Ajout des propriétés
HasDocumentReglement() As boolean
DocumentReglement () As IBODocumentReglement *

IBORegistreTaxeLigne

Ajout des propriétés
GetRT_TypeLigne() As TypeLigneRegistre
SetRT_TypeLigne (TypeLigneRegistre)

ITransformationAchat

Ajout des propriétés
CreateProcess_Facturer() As IPMDocTransformer

ITransformationVente

Ajout des propriétés
CreateProcess_Preparer() As IPMDocTransformer

IPMInventaire

nouveau processus permettant de gérer l'inventaire pour des articles, un dépôt, un emplacement (facultatif) et une date donnée