WebServices100

Documentation

Version 1.17

12/08/2026

Table des matières

I. Installation 12
A. Objets Métiers 13
B. Mise à jour 14
II. Configuration 15
A. Administration du service 15
1. Définition des champs 15
2. Particularité HTTPS 16
a) Les certificats 17
3. Test SOAP et REST 23
a) Erreurs lors du paramétrage HTTPS 24
b) Navigateur compatible 24
B. Environnements 25
1. Résolution erreurs OM 31
2. Migration de la configuration vers une nouvelle machine 32
C. Utilisation du token d'authentification 33
D. Services 34
1. Fonctionnalités OM actuellement implémentées 35
a) AdresseLivraisonService 35
b) BanqueService 35
c) CollaborateurService 36
d) CompteAnalytiqueService 36
e) CompteGeneralService 36
f) ContactTiersService 36
g) DocumentService 36
h) EcritureService 37
i) TiersService 37
j) RegistreRevisionService 37
E. Paramétrage des logs 37
1. Emplacement des fichiers 38
2. Paramètres de configuration 38
3. Écriture et gestion des fichiers 39
4. Canal Applicatif (LogAppDiagnostic) 39
5. Canal Performance (LogBenchmarkPerf) 40

6. Canal Santé (LogServiceHealth)....41
7. Champs de corrélation....42
8. Lire un fichier .clef....42
F. Gestion du WAF....44
G. Licence....46
1. Fenêtre de saisie....46
2. Le service refuse de démarrer alors que la licence est valide....48
H. Démarrage du Web Service....48
I. Route Health Check....49
III. Utilisation du Framework....50
A. Prise en main....50
1. Exemple .NET....50
a) Ajouter les références web services....50
b) Ajouter les librairies....51
B. Les services....52
1. Préambule....52
a) Criteria....52
b) Order....54
2. Les services communs....55
a) CalendrierService....55
b) DatabaseStructureService....56
c) FileService....57
d) PreferencesService....66
e) RisqueService....68
f) SqlService....70
g) UserService....71
h) UtilitiesService....73
i) StatistiqueService....78
3. Les services paramètres....80
a) CategorieComptableService....80
b) CategorieTarifaireService....83
c) ConditionLivraisonService....84
d) DeviseService....85
e) DossierService....86

f) GammeService....87
g) InfoLibreService....89
h) ModeExpeditionService....94
i) ModeleReglementService....95
j) NiveauAnalyseService....96
k) PaysService....98
l) PlanAnalytiqueService....99
m) ServiceContactService....100
n) StructureCompteService....101
o) TypeContactService....102
4. Les services de données de gestion commerciale....103
a) AbonnementService....103
b) AdresseLivraisonService....108
c) ArticleService....111
d) CollaborateurService....122
e) ContactTiersService....124
f) DocumentService....129
g) FamilleService....147
h) SoucheService....148
i) StockService....151
j) TarifService....157
k) TiersService....175
5. Les services de données de comptabilité....186
a) BanqueService....186
b) BanqueTiersService....190
c) CompteAnalytiqueService....192
d) CompteBancaireService....194
e) CompteGeneralService....197
f) ContactBanqueService....199
g) EcheanceService....202
h) EcritureService....204
i) ExtraitBancaireService....214
j) ModeReglementService....215
k) JournalService....217

l) TaxeService....219
m) RegistreRevisionService....222
C. Dictionnaire de données....225
1. Section commun....225
a) Calendrier....225
b) ContactTiers....225
c) Criteria....226
d) DocumentFileStream....226
e) DocumentAvecFileStream....226
f) DocumentFileStreamResult....226
g) DocumentSansFileStream....227
h) GetFilesStreamsRequest....227
i) EcritureAvecFileStream....227
j) EcritureFileStreamResult....227
k) EcrituresSansFileStream....227
l) InfoLibre....227
m) JoursCochable....227
n) ModeleReglement....228
o) NiveauAnalyse....228
p) Order....228
q) Pagination....228
r) Pays....228
s) ParametrageCompteGeneralTiers....229
t) ParametresTiers....229
u) Periodicite....229
v) Preferences....229
w) Tiers....230
x) TypeContact....232
y) UtilisateurSage....232
z) WebSqlParameter....233
2. Section Gestion Commerciale....234
a) Abonnement....234
b) AbonnementEnTete....236
c) AbonnementLigne....237

d) AbonnementPeriode 240
e) AdresseLivraison 240
f) Article 241
g) ArticleFournisseur 242
h) Catalogue 243
i) CategorieComptable 243
j) CategorieTarifaire 243
k) Collaborateur 244
l) CombinaisonGammeArticle 245
m) ConditionLivraison 245
n) Conditionnement 245
o) ConditionnementArticle 245
p) ConditionReglementDocument 246
q) Depot 247
r) DetailParametreNomenclature 248
s) Document 249
t) DocumentInterneInfo 252
u) Emplacement 252
v) EnumereConditionnement 252
w) Famille 252
x) FichierLieArticle 253
y) FichierLieDocument 253
z) FraisFixe 253
aa) Gamme 253
bb) GammeArticle 253
cc) InformationsDocumentsDestinationTransformation 254
dd) InformationsDocumentsOrigines 254
ee) InfoDocumentOrigineTransformation 254
ff) InfosLotSerieTransformation 254
gg) LigneDocument 255
hh) LignesDocumentsData 258
ii) ListUpdateEnteteDoc 258
jj) LotSerie 258
kk) ModeExpedition 258

ll) MotifPerteDevis....259
mm) MotifResiliation....259
nn) NomenclatureArticle....259
oo) ParametreEnumereGamme....259
pp) ParametreInsertArticle....259
qq) ParametreNomenclature....260
rr) PrixUnitaireLigneArticle....260
ss) PrixUnitaireLigneArticleParams....260
tt) Reglement....260
uu) RemiseFamilleClient....261
vv) RoutageReceptionFactureElectronique....261
ww) RoutageReceptionFactureElectroniqueInsertDataLight....261
xx) RoutageReceptionFactureElectroniqueInsertDataWithNumeroTiers....261
yy) RoutageReceptionFactureElectroniqueInsertLightWithNumeroTiersCollectionData....262
zz) RoutageReceptionFactureElectroniqueInsertToNumTiersCollectionData....262
aaa) RoutageReceptionFactureElectroniqueUpdateCollectionData....262
bbb) SoucheAchat....262
ccc) SoucheInterne....262
ddd) SoucheVente....262
eee) StatistiqueArticle....263
fff) Stock....263
ggg) StockDepot....263
hhh) StockEmplacement....264
iii) TarifArticle....264
jjj) TarifClient....265
kkk) TarifFournisseur....265
lll) TarifRemise....265
mmm) TransformationDocumentInfos....266
nnn) UniteVente....266
ooo) UpdateEnteteDocument....266
ppp) UpdateEnteteData....266
qqq) ValeurStatistiqueArticle....266
3. Section Comptabilité....267
a) Banque....267

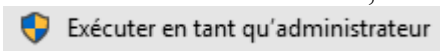
b) BanqueTiers....268
c) CompteAnalytique....269
d) CompteBancaire....270
e) CompteGeneral....271
f) ContactBanque....271
g) DataMajModeReglement....271
h) Devise....272
i) Echeance....273
j) Ecriture....273
k) EcritureAnalytique....275
l) ElementsLettrageRequest....275
m) ElementsLettrageResult....275
n) InfoEcriture....275
o) Journal....275
p) LettrageElement....276
q) LienEcritureEcheance....276
r) MentionExonerationTaxe....276
s) ModeReglement....276
t) Periode....277
u) PlanAnalytique....277
v) StructureCompte....277
w) Taxe....277
x) UpdateModeReglementResult....278
4. Enumérations....279
a) AbonnementCalculEcheance....279
b) AbonnementDateReference....279
c) AbonnementDureeDelaiPreavis....279
d) AbonnementEtatPeriodicite....279
e) AbonnementProrataLigne....279
f) AbonnementReconduction....279
g) AbonnementSourceContact....280
h) AbonnementSourceGeneration....280
i) AbonnementType....280
j) AbonnementTypeDuree....280

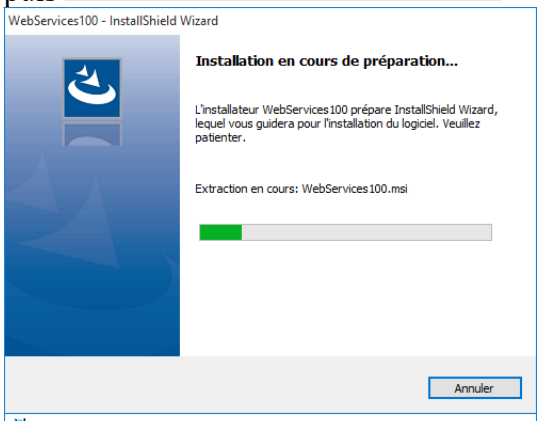
k) AbonnementTypePeriodicite 280
l) AbonnementTypeTiers....280
m) ActionRisque....281
n) CessionCreanceEcriture 281
o) ChampsDocumentUpdate....281
p) Civilite 281
q) ComparisonOperator....281
r) DomaineAffaire 283
s) DomaineDocument....283
t) DbType 284
u) FactureElectroniqueAssujettissement....284
v) FactureElectroniqueAutreIdentifiantType 285
w) FactureElectroniqueControleEmission 285
x) FactureElectroniqueTypeEntite....285
y) LangueSage 285
z) LocalisationPays....285
aa) LogicalOperator 285
bb) ModeControleEnCours 286
cc) ModeFacturationAffaire 286
dd) OrderType 286
ee) ParameterDirection 286
ff) PremierJourSemaine....286
gg) PremiereSemaineAnnee 286
hh) PeriodiciteLigne 286
ii) PrioriteDestockage 287
jj) ProvenanceDocument 287
kk) ReconductionLigne 287
ll) RemiseTypeLigne....287
mm) SensEcriture....287
nn) SensTaxe 287
oo) StatutAffaire....287
pp) StatutDocument 288
qq) StructureCompteBancaire....288
rr) TypeArticle 288

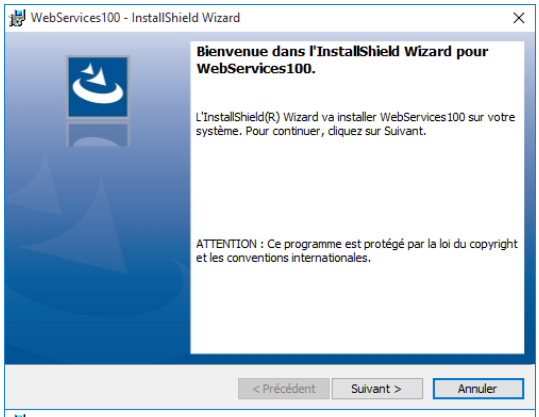
ss) TypeCodeEDI 288
tt) TypeCodeRoutage 288
uu) TypeCompte 288
vv) TypeDocument 289
ww) TypeEmplacement 289
xx) TypeEscompte 289
yy) TypeFamille 290
zz) TypeFrais 290
aaa) TypeFraisModeExpedition 290
bbb) TypeFrancoModeExpedition 290
ccc) TypeInfoLibre 290
ddd) TypeJournal 290
eee) TypeMouvement 290
fff) TypeNatureCompte 291
ggg) TypeNomenclature 291
hhh) TypeNumerotationPieceJournal 291
iii) TypeNumerotationTiers 291
jjj) TypePrix 291
kkk) TypeParametrageComptePrincipal 292
lll) TypeProvenanceTaxe 292
mmm) TypeRemise 292
nnn) TypeRepartitionReglement 292
ooo) TypeReportCompte 292
ppp) TypeSuiviStock 292
qqq) TypeTiers 293
rrr) TypeTarificationFournisseur 293
sss) TypeTarificationQuantiteMontant 293
ttt) TypeTauxTaxe 293
uuu) TypeTaxe 293
vvv) UnitePoids 293
www) ZoneEmplacement 294
xxx) TypeConditionReglement 294
yyy) TypeRepartitionReglement 294

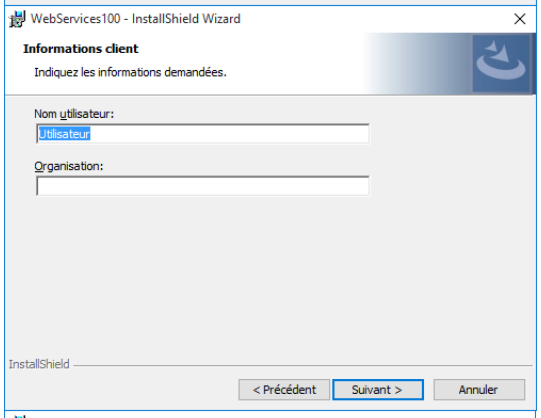
I.Installation

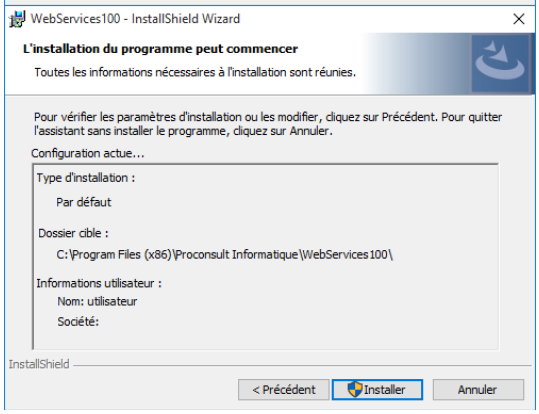
Lancez le setup de WebServices100.

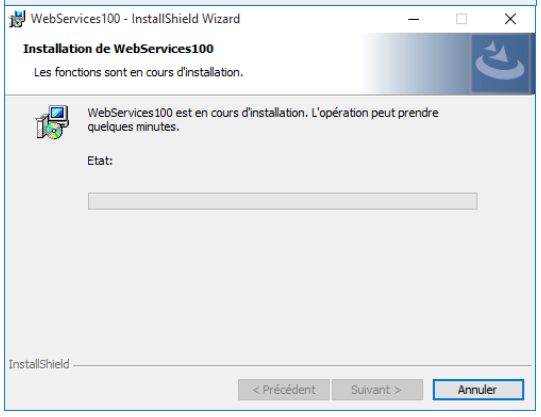
Nota : Vous devez possédez les droits « Administrateur » pour pouvoir installer cette application. Dans le cas contraire, lancez le Setup en tant qu'administrateur par un clic droit puis 

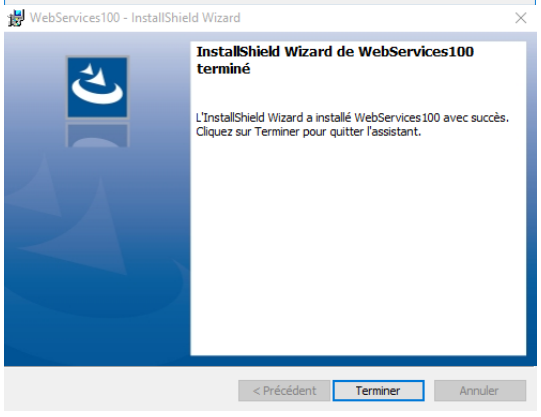












A. Objets Métiers

Les Objets Métiers sont intégrés dans les WebServices.
Pour OnPremise, les 5 dernières versions de Sage sont supportées.
Pour SPC, seule la dernière version est supportée.

Cf. Administration du service
Cf. Fonctionnalités OM actuellement implémentées

Afin de pouvoir utiliser les OM il faudra installer le redistribuable correspondant à la version souhaitée.

Lien de téléchargement des redistribuables :
https://fr-kb.sage.com/portal/app/portlets/results/viewsolution.jsp?solutionid=211010150079696

B. Mise à jour

Certaines versions des Webservices100 sont uniquement publiées sous forme de « Patch ».
Afin d'effectuer la mise à jour via un Patch, il faudra suivre ces étapes :

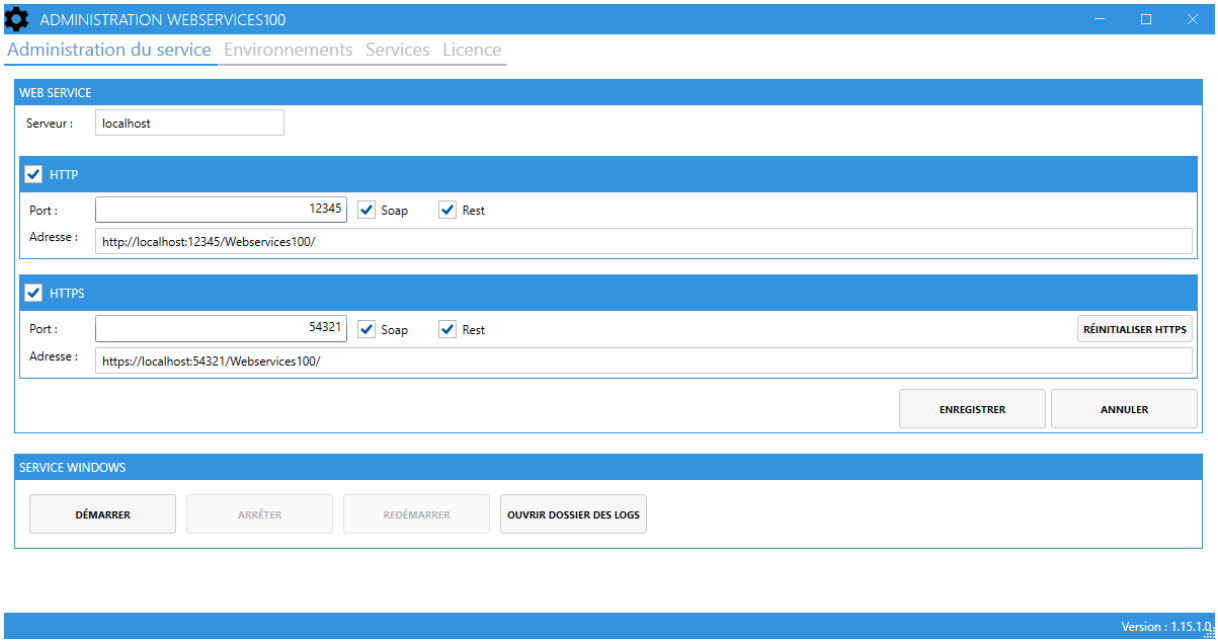
1) Arrêter le service Windows « WebServices100 ».
2) Fermer l'interface d'Administration.
3) Effectuer une copie de votre répertoire d'installation des WebServices100.
4) Copier les éléments du Patch dans votre répertoire d'installation des WebServices100.
5) Ouvrir l'interface d'Administration et vérifier que la version indiquée en bas à droite de la fenêtre correspond à la version mise à jour.
6) Redémarrer le service Windows.

Si après avoir suivi ces étapes, l'interface d'Administration ne s'ouvre pas, il faudra également :

7) Copier les fichiers présents dans les répertoire « Fichiers de config dll » et « Fichiers de configuration par défaut » dans votre répertoire racine WebServices100.
8) Vérifier que l'interface d'Administration s'ouvre.
9) Réappliquer votre paramétrage concernant la gestion des Logs dans le fichier de configuration « WebServices100Service.exe.config ».
10) Redémarrer le service Windows.

II.Configuration

A. Administration du service



1. Définition des champs

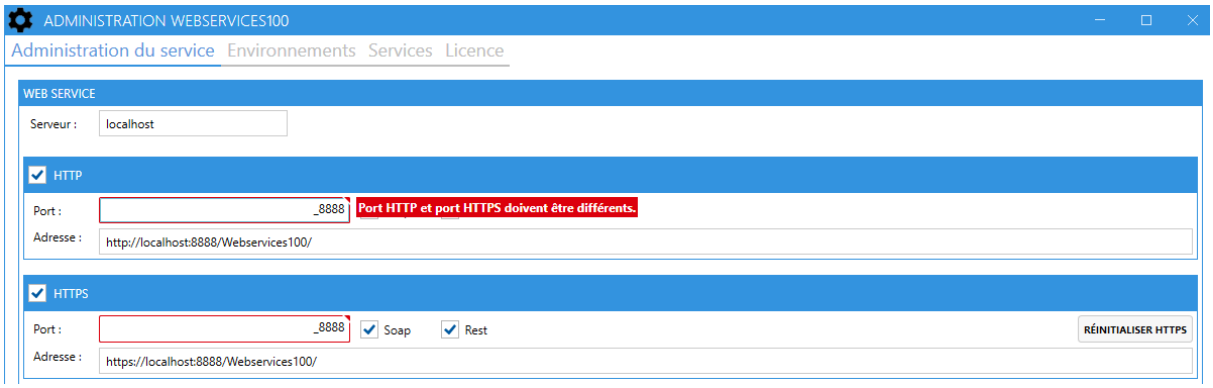
- **Serveur :** Nom ou adresse IP du serveur hébergeant le web service. Généralement, cette valeur sera localhost ou 127.0.0.1.
- **Port :** port d'accès au web service.
- **Soap :** coché pour permettre l'utilisation du protocole SOAP (Simple Object Access Protocol).
- **Rest :** coché pour permettre l'utilisation du protocole REST (REpresentational State Transfert).
- **Adresse :** Le champ Adresse affiche l'URL à utiliser pour accéder au web service. *Champ en lecture seule.*

Enregistrer : Enregistre les modifications.
Annuler : Annule les modifications.

Nota : Suite à toute modification dans l'administration, il faudra redémarrer le service avec le bouton « Redémarrer ».

Les WebServices100 supportent les protocoles HTTP et HTTPS.
Pour activer un protocole, il faut cocher la case correspondante.

Nota : Il est possible de faire fonctionner les 2 protocoles en simultanée. En revanche, les ports HTTP et HTTPS doivent être différents.

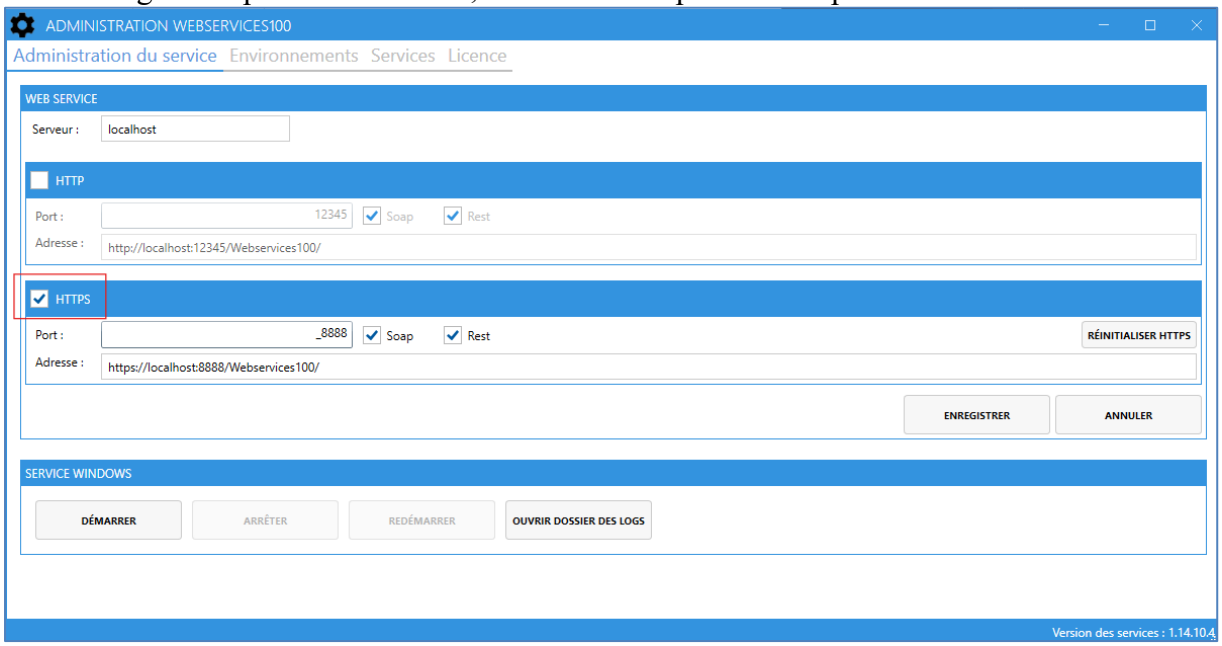


Nota : L'utilisation des 2 protocoles en simultanée multiplie par 2 les ressources utilisées par les WebServices100.

2. Particularité HTTPS

La configuration automatique du mode HTTPS requiert un **accès à Internet**.
Si vous ne disposez pas d'un tel accès, merci de contacter notre **support** qui procédera à une **installation hors-ligne**.

Pour configurer le protocole HTTPS, il faut dans un premier temps cocher la case HTTPS.



Lorsque HTTPS est configuré pour la première fois, lors de l'enregistrement des informations de configuration, **le processus de configuration HTTPS** se déclenche. Celui-ci effectue les opérations suivantes :

1) Création et téléchargement des certificats nécessaires.
2) Réservation de l'URL.
3) Association du port choisi avec le certificat.

En cas de modification du port HTTPS, seul les étapes 2 et 3 seront exécutées.

Réinitialiser HTTPS : les certificats installés seront supprimés, la réservation d'URL et la liaison du port seront annulés.

Lors du prochain enregistrement des informations de configuration, le process complet de configuration HTTPS se déclenchera à nouveau.

a) Les certificats

L'utilisation du protocole HTTPS implique l'utilisation de certificat pour crypter et décrypter la communication. Il vous est possible :

- d'utiliser un certificat auto-généré avec l'administration des WebServices100,
- d'utiliser un de vos certificats.

Utilisation d'un certificat auto-généré par l'administration WebServices100

La génération d'un certificat, via l'administration des WebServices100, nécessite l'utilisation de 2 certificats :

- WebServices100Certificate
- WS-RootCA

Le certificat **WS-RootCA** permet de signer le certificat **WebServices100Certificate** et ainsi éviter les avertissements liés au certificat auto-signé ou signé par une autorité de certification inconnue.

L'utilisation du protocole HTTPS impacte les applications clientes dialoguant avec les WebServices100. En effet, il peut y avoir des avertissements HTTPS empêchant la communication.
Il y a 2 solutions possibles pour résoudre ce problème :

- La première consiste à modifier le code source de l'application pour ignorer ces avertissements.
- La seconde consiste à ajouter le certificat WS-RootCA sur le poste ou est installé l'application cliente. Dans ce cas, il faudra utiliser l'adresse IP du serveur, et non pas son nom, dans l'url des appels WebServices100.

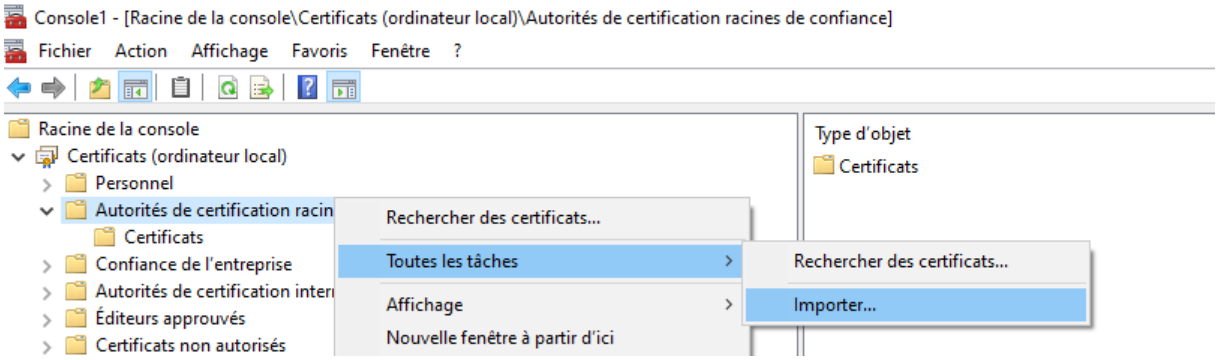
Si l'application cliente est sur le même poste que les WebServices100, il n'est pas nécessaire d'ajouter le certificat puisque cela aura été fait lors du **processus de configuration HTTPS**.

1 Adresse de téléchargement du certificat RootCA

Le certificat WS-RootCA est téléchargeable à l'adresse suivante :
https://webservices-https.azurewebsites.net/api/WSHttps/GetRootCA

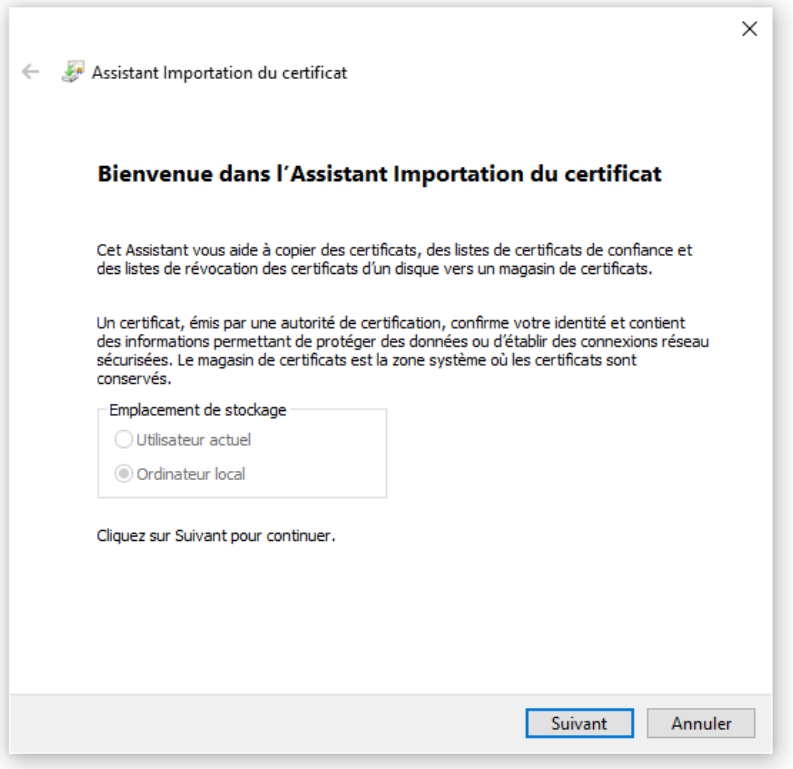
2 Procédure d'installation du certificat

Dans le menu démarrer du poste client, saisir « Gérer les certificats d'ordinateur ».

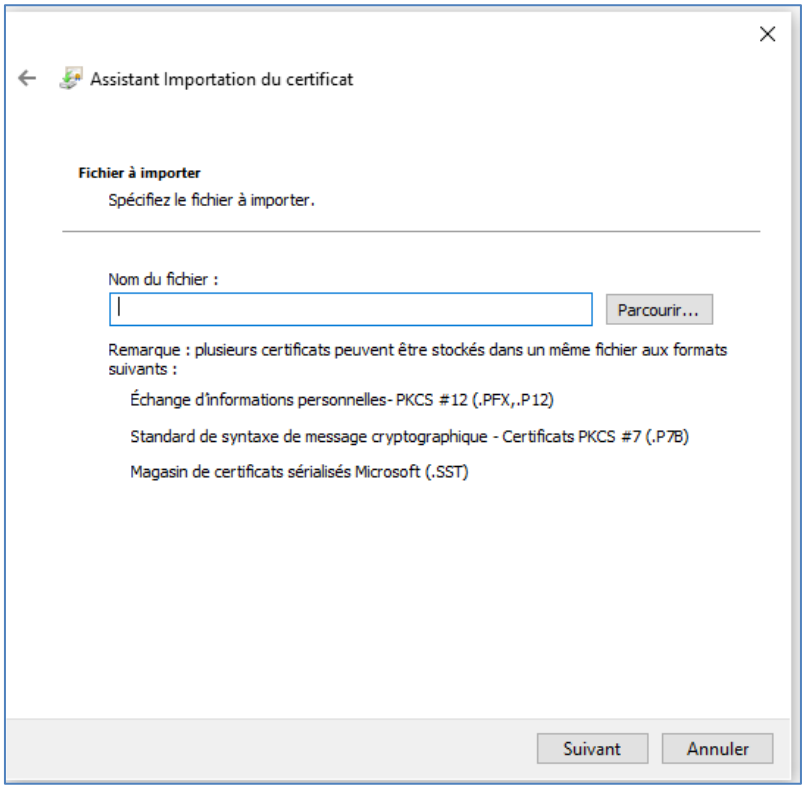


Sélectionner « Certificats (ordinateur local) » → « Autorité de certification racine ».
Clic droit → Toutes les tâches → Importer…

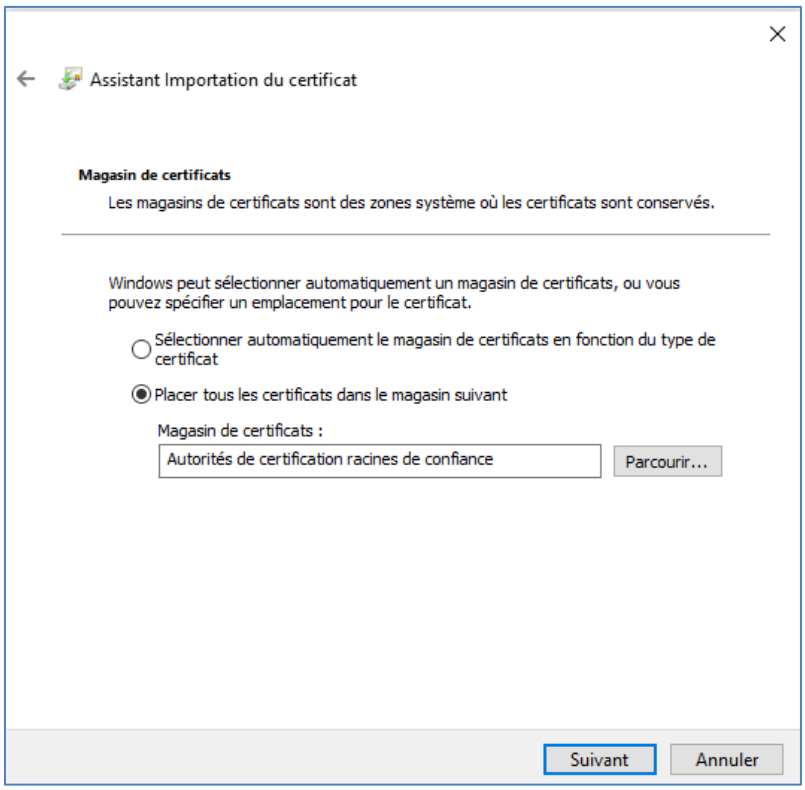
La fenêtre d'assistant d'importation de certificat s'ouvre.



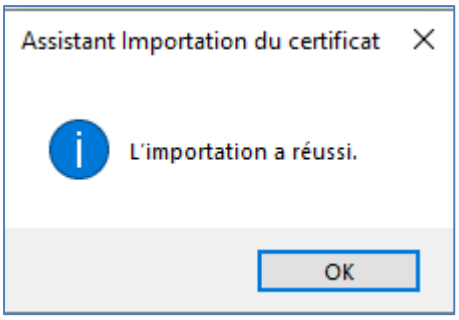
Cliquer sur « Suivant ».



Cliquer sur « Parcourir… »
Sélectionner le certificat WS-RootCA que vous avez téléchargé précédemment.
Cliquer sur « Suivant ».



Si ce n'est pas le cas :
Cocher « **Placer tous les certificats dans le magasin suivant** »
Sélectionner le magasin de certificats « **Autorités de certification racines de confiance** ».
Cliquer sur « Suivant ».



L'import du certificat est réussi.
Cliquer sur « OK » pour terminer le processus d'import de certificat.

Utilisation d'un de vos certificats

Il également possible d'utiliser votre certificat. Ce certificat doit être un fichier PFX (extension .pfx ou .p12).
Pour créer un fichier PFX à partir d'un fichier .crt (certificat publique) et d'un fichier .key (clé privée), il est possible d'utiliser la commande ***OpenSSL*** suivante :

```
> openssl pkcs12 -export -out Certificat.pfx -inkey cle_privee.key -in cle_publique.crt -passout pass:MotDePasse
```

- Certificat.pfx : nom du fichier PFX à créer (extension .pfx requise),
- cle_privee.key : fichier KEY contenant la clé privée (extension .key requise),
- cle_publique.crt : fichier CRT contenant la clé publique (extension .crt requise),
- MotDePasse : mot de passe permettant de sécuriser le fichier PFX.

*Nota : Il est possible de laisser le champ **MotDePasse** vide.*

Si lors de l'ajout du certificat un message d'erreur concernant le mot de passe apparait, alors que celui-ci est correct. Il peut y avoir un souci de version Windows. Dans ce cas, il faut utiliser la commande suivante :

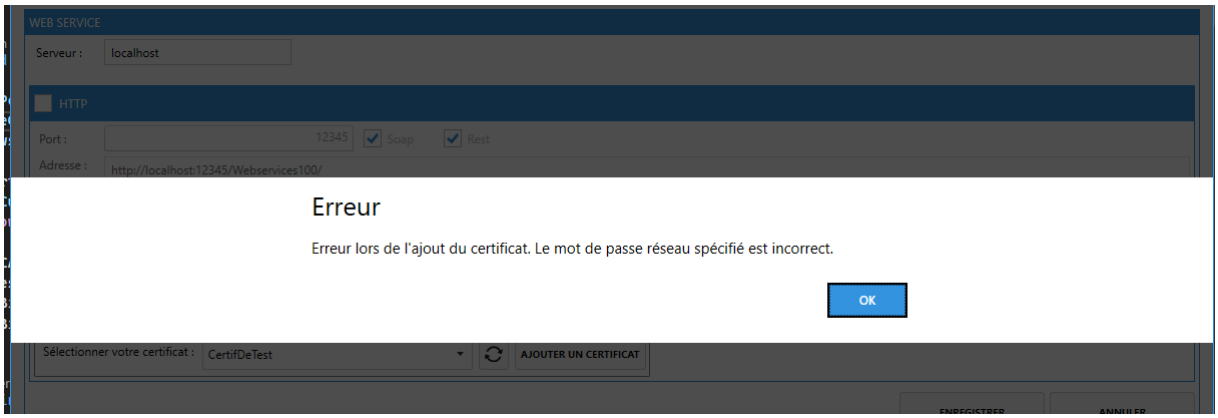
```
> openssl pkcs12 -export -certpbe PBE-SHA1-3DES -keypbe PBE-SHA1-3DES -nomac -inkey cle_privee.key -in cle_publique.crt -export -out Certificat.pfx
```

Pour utiliser votre propre certificat, dans la section HTTPS, il faut cocher la case « Utiliser votre propre certificat ».

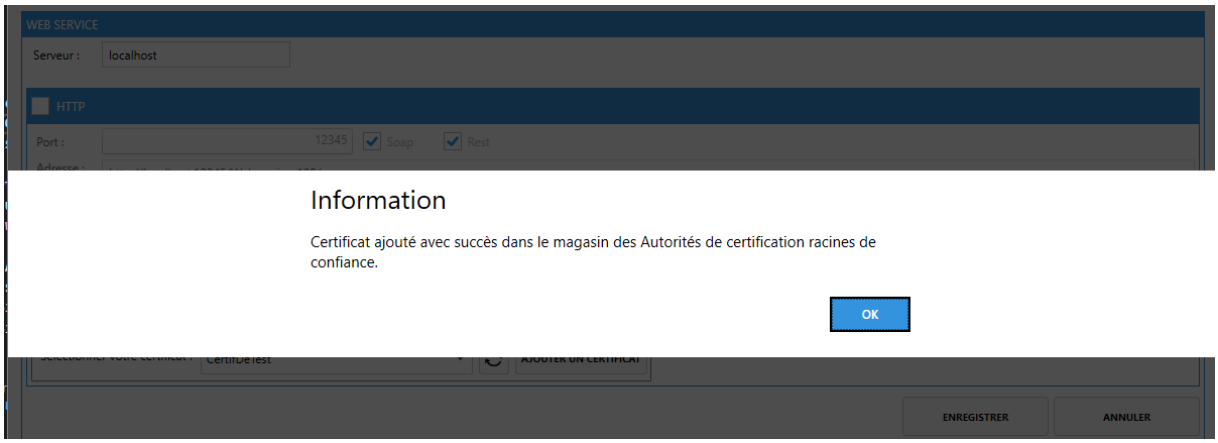
Si votre certificat n'est pas présent dans la liste déroulante :

1. Cliquer sur le bouton ajouter.
2. Sélectionner votre fichier PFX.
3. Saisir le mot de passe (laisser vide si pas de mot de passe).

Si le mot de passe n'est pas correct, l'erreur suivante apparaitra :



Dans le cas où le mot de passe est correct et que l'ajout a réussi :

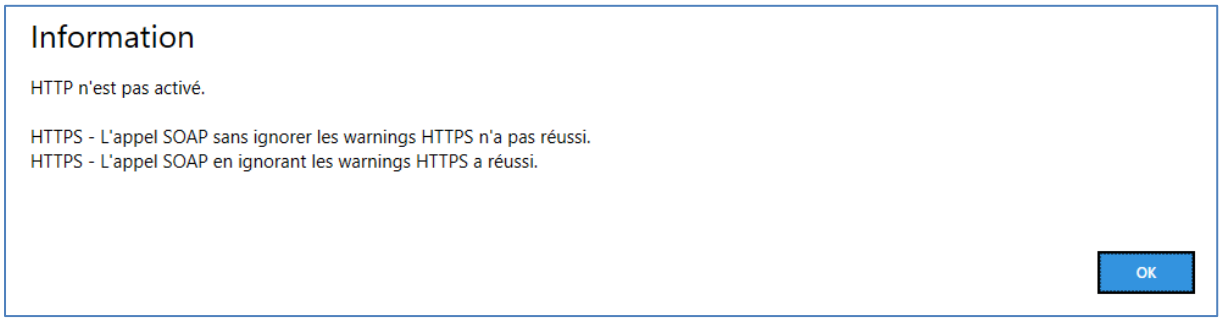


Afin de finaliser le paramétrage HTTPS, sélectionner votre certificat, puis cliquer sur le bouton « Enregistrer ».

3. Test SOAP et REST

Dans le cas où votre certificat ne couvre pas le nom de domaine « localhost », l'exécution des tests SOAP et REST est susceptible de générer des warnings HTTPS. Un deuxième appel de test en ignorant les warnings HTTPS est automatiquement effectué :

La configuration HTTPS pour les appels REST avec votre certificat est opérationnel.



La configuration HTTPS pour les appels SOAP avec votre certificat est opérationnel.

a) Erreurs lors du paramétrage HTTPS

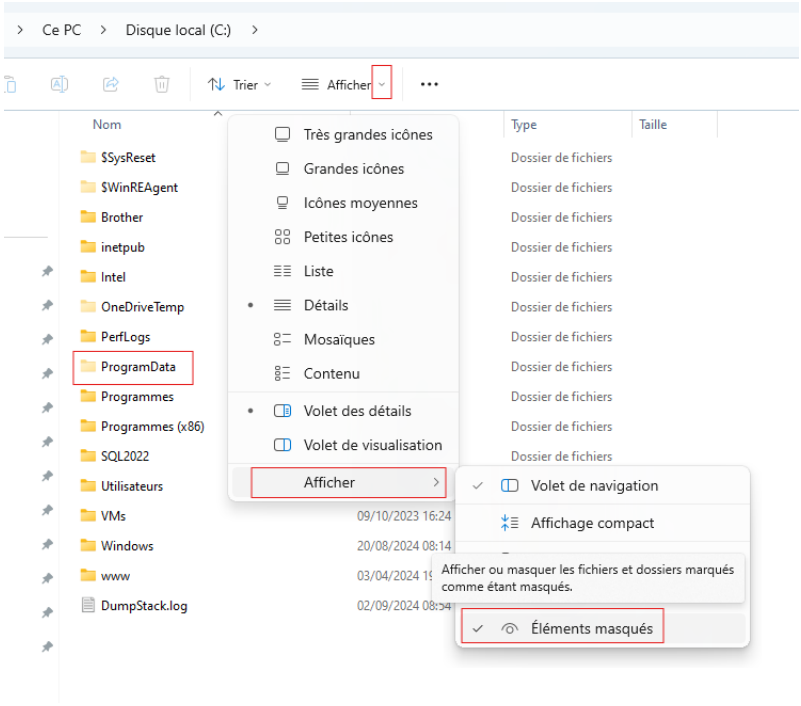
Lors du **processus de configuration HTTPS** il peut arriver que le message d'erreur suivant s'affiche.

« Les IP ne correspondent pas ».

Dans ce cas, il faut :

- Supprimer le fichier de licence WebServices100, nommé WS100.lic, qui se trouve dans le répertoire *C:\ProgramData\PPI-Group,*
- Aller dans l'onglet « License »,
- Cliquer sur le bouton « Rafraîchir »,
- Relancer le **processus de configuration HTTPS**.

Le répertoire ProgramData est un dossier caché. Voici comment l'afficher.



b) Navigateur compatible

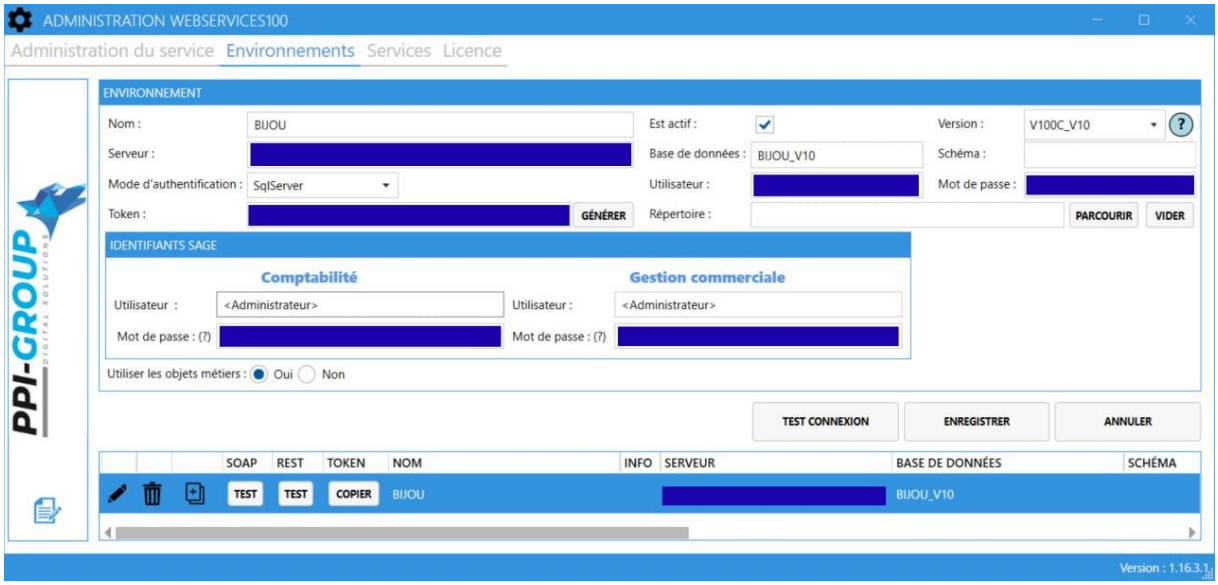
Pour tester les appels Rest via un navigateur, veuillez utiliser le navigateur Firefox.

B. Environnements

C'est dans cet onglet que sont définis les correspondances entre les noms logiques et les bases de données.

Utiliser les Objets Métiers : Cocher « Oui » permet d'activer l'utilisation des OM sur les endpoints pour lesquels une implémentation a été réalisée (cf. II.D.1 Fonctionnalités OM actuellement implémentées).

Nota : L'utilisation des OM aura des répercussions sur les configurations suivantes. Chaque cas — utilisation avec ou sans OM — sera détaillé de manière distincte afin de clarifier les différences de comportement.



Nom : Nom logique de l'environnement du web service.
Est actif : Défini si l'environnement est actif.
Serveur : Serveur de base de données auquel l'environnement est connecté.
Base de données : Nom de la base de données.

Schéma : Ce champ est à remplir uniquement dans un contexte d'utilisation SPC. On précise ici le schéma correspondant à l'entité que l'on veut cibler.
Particularités OM : il faut également indiquer le nom de l'entité.

Version : Version de Sage de la base de données.
Particularités OM : Pour OnPremise, les 5 dernières versions de Sage sont supportées. Pour SPC, seule la dernière version est supportée.

Mode d'authentification : Mode d'authentification à la base de données.

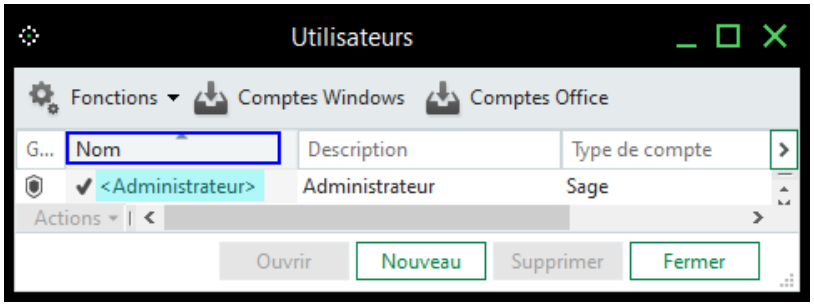
- Windows : Le compte windows courant est utilisé pour l'accès à la base de données.
- Sql Server : Dans ce mode, il faut aussi définir le nom utilisateur et le mot de passe à utiliser pour se connecter à la base de données.

Générer : Permet la génération d'un token d'authentification. On peut aussi saisir son propre token.

Parcourir : Permet de choisir le répertoire où sont stockés les fichiers en rapport avec la base SAGE (les fichiers «. gcm, .mae, .imo » et le repertoire « Mulitimédia »)

Vider : Permet de vider le répertoire sélectionné.

Identifiants Sage :
Le bloc « Identifiants Sage » n'est disponible que si la version de Sage sélectionnée est supérieure ou égale à la version 100C V5.
Concernant le nom d'utilisateur, il faut saisir le **nom tel que celui-ci est inscrit** dans les utilisateurs Sage. S'il y a des caractères spéciaux dans le nom, ils doivent aussi être présent. Dans la capture, il y a les caractères < et >, ils doivent être présent dans le nom.

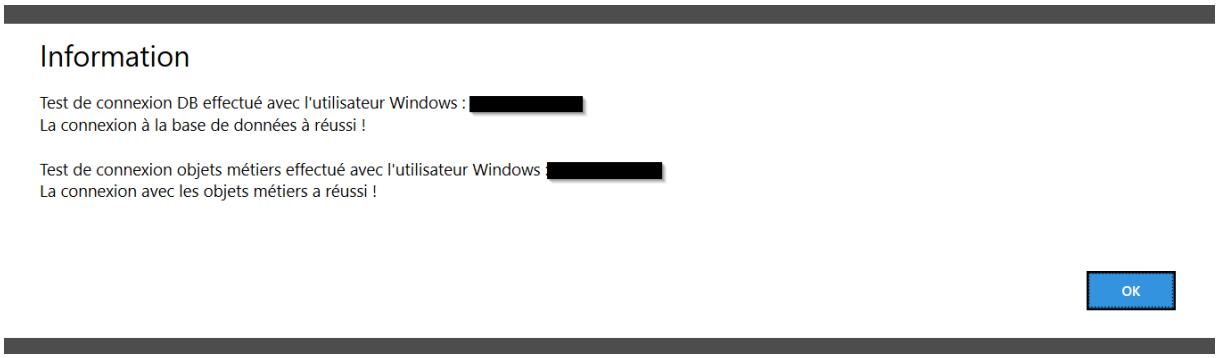


Identifiants Sage – Non OM : Lors des opérations attendant un Guid d'utilisateur Sage, si vous ne fournissez pas un Guid, les WebServices100 utiliseront l'utilisateur par défaut correspondant au contexte (Comptabilité ou Gestion commerciale) pour obtenir le Guid qui lui est associé.

Identifiants Sage – OM : L'utilisation des OM repose sur le mécanisme d'authentification de Sage, avec un identifiant et un mot de passe. Ainsi, lors de l'utilisation des OM, le bloc « Identifiants Sage » précise les comptes utilisés pour exécuter les actions. Par conséquent, lorsqu'un utilisateur agit via les OM, c'est comme si ces actions étaient effectuées directement par cet utilisateur dans l'application Sage.

Nota : sur SPC l'authentification passe par l'authentification SQL Server intégrée. Les champs « Mot de passe » n'apparaissent donc pas.

Test connexion : Permet de tester la connexion à la base de données SAGE.
Particularité OM : Un test supplémentaire de connexion OM via les identifiants Sage est effectué.

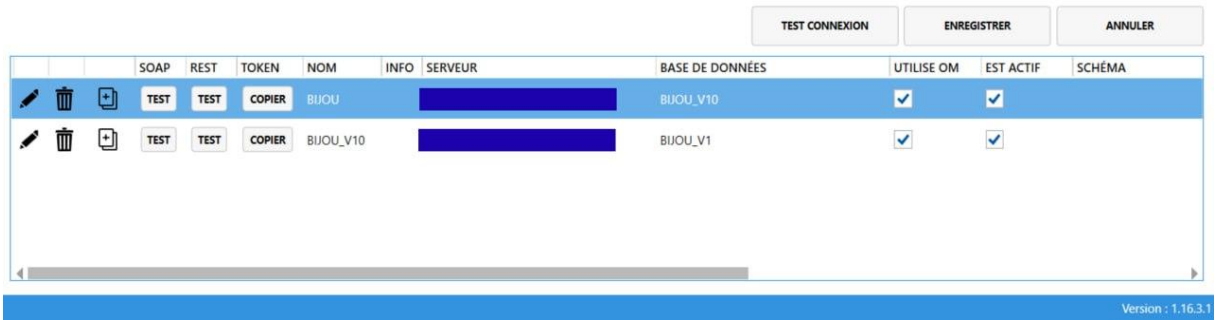


Annuler : Annule les modifications apportées à l'environnement.

Enregistrer : Enregistre les modifications de l'environnement.
Particularité OM : L'utilisation simultanée de plusieurs environnements avec des versions différentes d'objets métiers n'est pas supportée. Cependant l'enregistrement des configurations n'est pas restreint.

Néanmoins dans cette configuration le service ne pourra pas démarrer. Un message d'avertissement sera affiché en changeant d'onglet ou lors du démarrage du service.

En cas de démarrage via les « Services » de Windows, le message sera inscrit dans le fichier de log.



Modification : Un environnement est modifiable par un clic sur l'icône ✎ ou par un double clic.

Suppression : La suppression d'un environnement se fait par un clic sur l'icône 🗑.

Cloner : Permet de dupliquer un environnement ⧉.

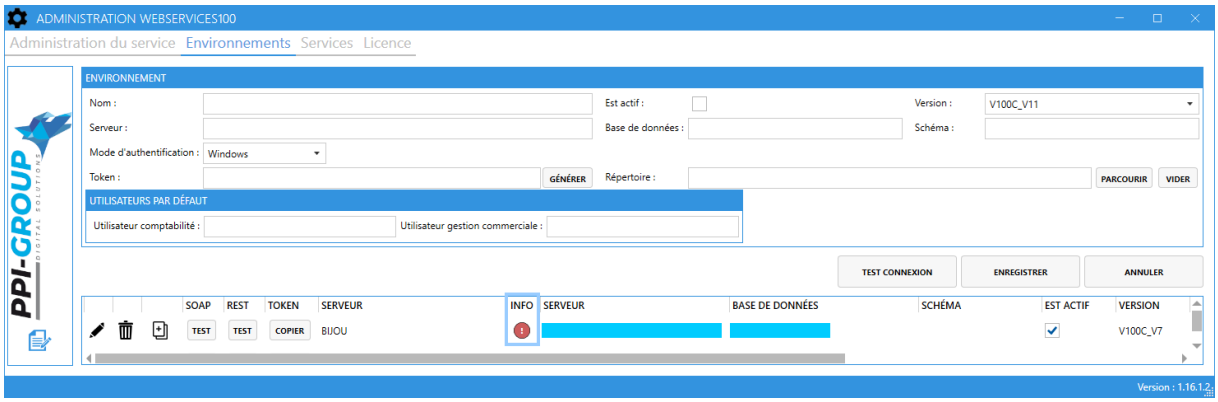
Le bouton « Test » de la colonne « SOAP » : permet de tester un appel SOAP d'une fonction du service TiersService

Le bouton « Test » de la colonne « REST » : permet de tester un appel REST d'une fonction du service TiersService

Le bouton « Copier » de la colonne « TOKEN » : permet de copier le Token dans le presse-papier

Nota : *Suite à toute modification à un environnement, il faut procéder au redémarrage du web service (onglet « Administration du service »).*

Les environnements modifiés, alors que le service WebServices100 est démarré, auront une icône dans la colonne « Info », précisant que les données de cet environnement ont été modifié sans avoir redémarré le service.



Ceci pourrait expliquer pourquoi il y a des différences de comportement entre un appel via une application cliente (middleware, postman, …) et les tests Rest / Soap / de connexion, effectués depuis l'administration WebServices100.

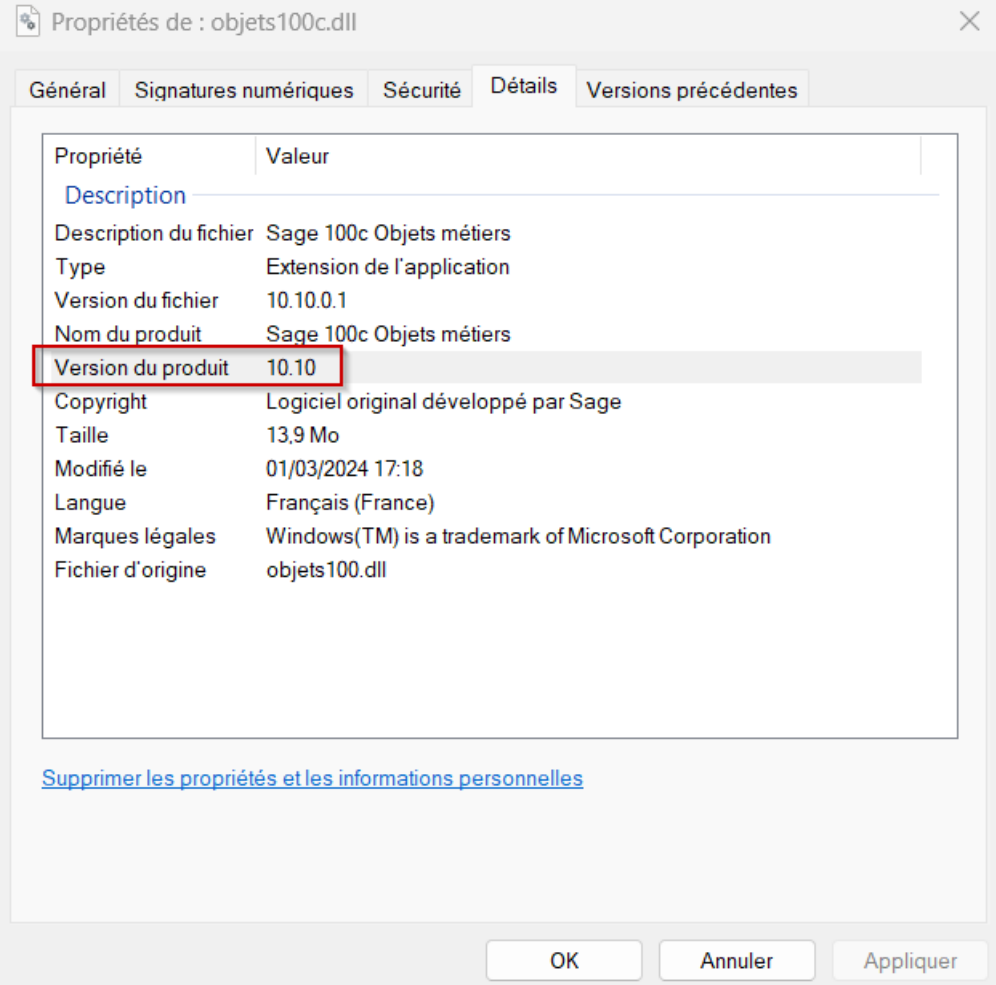
1. Résolution erreurs OM

Des erreurs peuvent survenir lors du test de connexion :

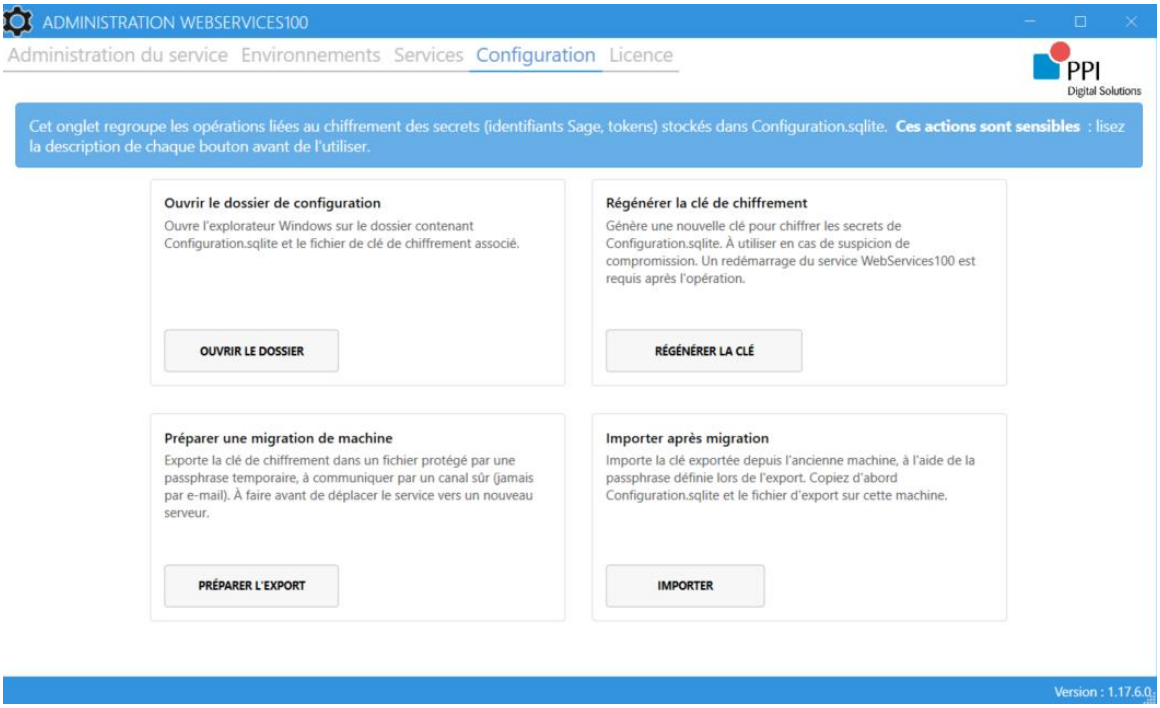
- « Erreur : Ce fichier n'est pas converti, veuillez utiliser la maintenance ! »
- « Erreur : Ce fichier est déjà converti, veuillez mettre à jour votre application ! »

Ces messages indiquent une incohérence entre la version des OM installés et la version de l'environnement sélectionné.

Il faudra vérifier que la version du fichier **objets100c.dll** du répertoire *C:\Program Files (x86)\Common Files\Sage\Objets métiers* corresponde à la version de l'environnement sélectionné.



2. Migration de la configuration vers une nouvelle machine



Onglet « Configuration » de l'administration WebServices100

Depuis la version 1.17.6.0, une nouvelle procédure permet de préparer puis d'importer un export de la configuration (Configuration.sqlite) lors d'un changement de serveur ou de poste hébergeant le service WebServices100. Ces actions sont disponibles dans l'onglet « Configuration » de l'administration, aux côtés de l'ouverture du dossier de configuration et de la régénération de la clé de chiffrement.

Étape 1 — Préparer l'export (sur l'ancienne machine)

1. Dans l'onglet « Configuration », cliquez sur « Préparer l'export ».
2. Saisissez une passphrase temporaire, puis confirmez-la. Cette passphrase doit être communiquée à la personne qui réalisera l'import, par un canal sûr (jamais par e-mail).
3. Choisissez l'emplacement où enregistrer le fichier d'export (extension .dat).
4. Une fois l'export créé, copiez ce fichier ainsi que Configuration.sqlite vers la nouvelle machine.

Étape 2 — Importer après migration (sur la nouvelle machine)

1. Copiez Configuration.sqlite dans le répertoire suivant sur la nouvelle machine : C:\Users\Public\Documents\Proconsult Informatique\WebServices100
2. Copiez également le fichier d'export (.dat) sur la nouvelle machine.
3. Dans l'onglet « Configuration », cliquez sur « Importer ». Si une clé de chiffrement existe déjà sur cette machine, une confirmation est demandée : l'import la remplacera définitivement.
4. Sélectionnez le fichier d'export (.dat) copié précédemment.
5. Saisissez la passphrase définie lors de l'export.
6. Une fois l'import confirmé, démarrez (ou redémarrez) le service WebServices100 pour appliquer le changement.

Une fois l'import validé sur la nouvelle machine, il est recommandé de supprimer le fichier d'export de l'ancienne machine.

Régénérer la clé de chiffrement
Ce bouton, disponible dans le même onglet « Configuration », génère une nouvelle clé de chiffrement pour les secrets stockés dans Configuration.sqlite (identifiants Sage, tokens). Il s'utilise notamment en cas de suspicion de compromission de la clé actuelle. Après la régénération, un redémarrage du service WebServices100 est nécessaire pour que le changement soit pris en compte.

C. Utilisation du token d'authentification

Chaque environnement doit disposer de son propre token. Lors de l'enregistrement d'une configuration d'environnement, plusieurs contrôles sont automatiquement effectués :

- Longueur minimale : le token doit comporter au moins 20 caractères.
- Robustesse : le token doit respecter un niveau de complexité minimal (majuscules, minuscules, chiffres).
- Unicité : le token ne doit pas déjà être utilisé par un autre environnement.

Ces vérifications garantissent qu'un token unique, suffisamment long et robuste est associé à chaque environnement.

Il est possible de définir son propre token manuellement, ou de le générer automatiquement via le bouton "Générer", qui produit un token respectant l'ensemble des conditions évoquées ci-dessus.

Ce dernier doit être indiqué lors de chaque initialisation d'un service.

Initialisation avec token :
Description : Pour faire un appel d'un service avec un token d'authentification, il faut initialiser le service avec le constructeur prenant comme paramètre : l adresse du service et un token

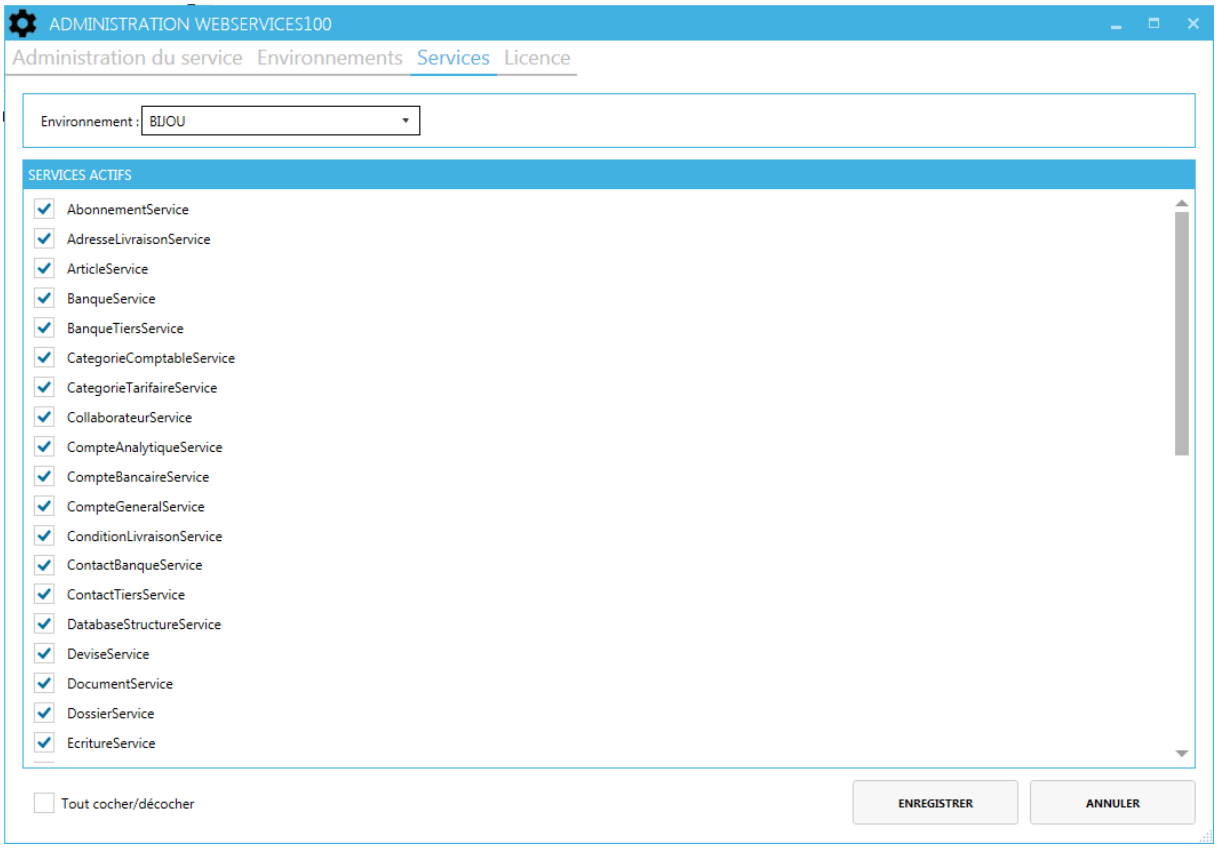
Exemple :
`String token = "TOKENTEST";`
`ITiersService service = new TiersService ("http://localhost:12345/Webservices100/BIJOU/", token);`

Initialisation sans token :
Description : Pour faire un appel d'un service avec un token d'authentification, il faut initialiser le service avec le constructeur ne prennant comme paramètre que l'adresse du service

Exemple:
`String token = "TOKENTEST";`
`ITiersService service = new TiersService ("http://localhost:12345/Webservices100/BIJOU/");`

D. Services

Dans cet onglet sont définis les services des environnements.

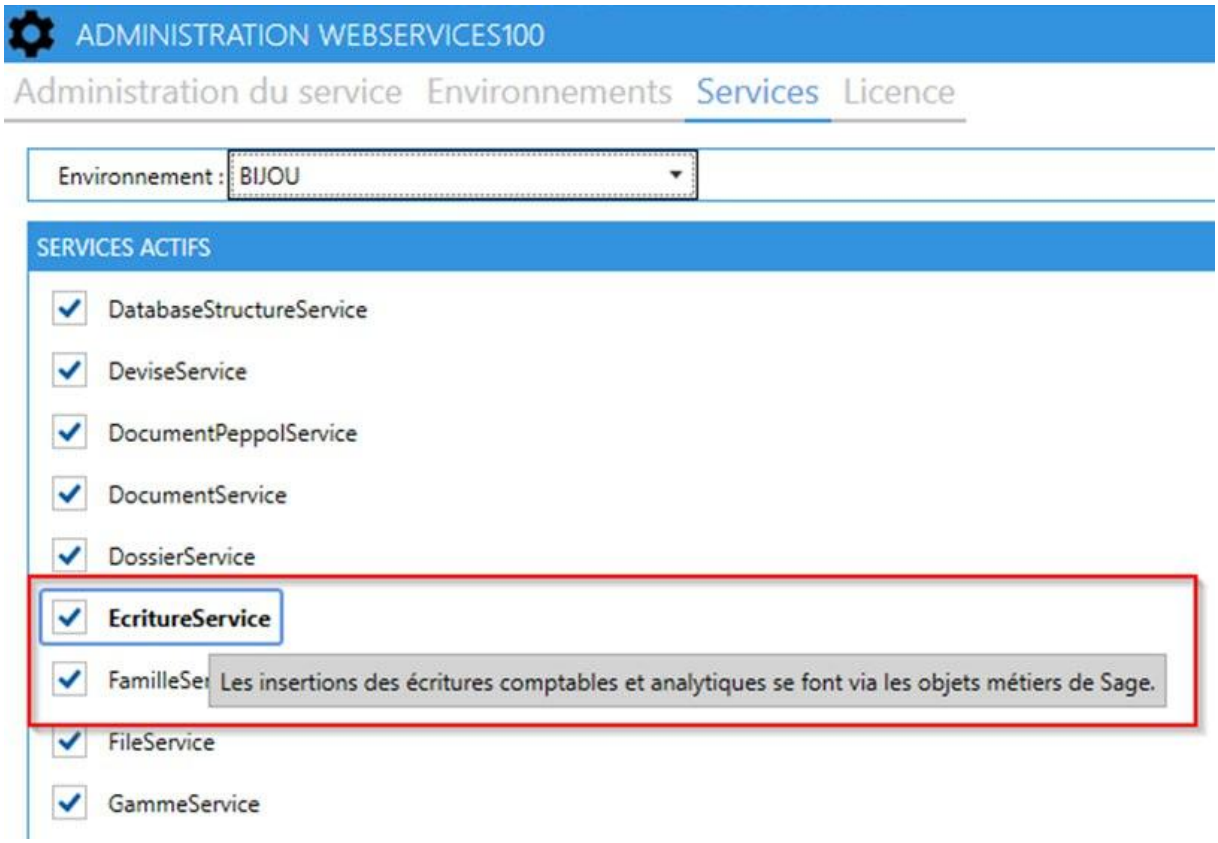


Environnement : Sélection de l'environnement.
Services actifs : Les services cochés seront les services disponibles sur l'environnement sélectionné.

Suite à toute modification sur les services d'un environnement, il faut procéder au redémarrage du web service (onglet « Administration du service »).

Particularités OM :

Les services exploitant les OM sont identifiés et encadrés.
Une description accompagne chaque service, précisant les actions qui mobilisent les objets métiers.



1. Fonctionnalités OM actuellement implémentées

a) AdresseLivraisonService

- Insertion d'une adresse de livraison
  - Insert
  - InsertV2

b) BanqueService

- Insertion, suppression et modification des banques sociétés
  - Insert
  - InsertV2
  - Update
  - UpdateV2
  - DeleteById
  - DeleteByAbrege
  - DeleteByAbregeV2

c) CollaborateurService

- Insertion, suppression et modification des collaborateurs
  - Insert
  - InsertV2
  - Update
  - UpdateV2
  - Delete

d) CompteAnalytiqueService

- Insertion et modification des comptes analytiques
  - Insert
  - InsertV2
  - Update
  - UpdateV2

e) CompteGeneralService

- Insertion et modification des informations libres des comptes généraux
  - Insert
  - InsertV2
  - UpdateInfosLibres
  - UpdateInfosLibresV2

f) ContactTiersService

- Insertion d'un contact de tiers
  - Insert
  - InsertV2

g) DocumentService

- Insertion des documents et lignes de vente, des documents et lignes de mouvement de transfert, ainsi que des documents et lignes de bons de commande achat.
  - InsertDocument
  - InsertDocumentV2
  - InsertDocumentEtLignes
  - InsertDocumentEtLignesV2
  - InsertLigne
  - InsertLigneV2
  - InsertLigneDocumentList
  - InsertLigneDocumentListV2
  - UpdateInfosLibres
  - UpdateInfosLibresV2

h) EcritureService

- Insertion des écritures comptables
  - InsertPieces
  - InsertPiecesV2
  - InsertEcritures
  - InsertEcrituresV2
- Insertion des écritures analytiques
  - InsertAnalytiques
  - InsertAnalytiquesV2
- Mise à jour des modes de règlement
  - UpdateModeReglement
  - UpdateModeReglementV2
- Lettrage
  - LettrerEcritures

i) TiersService

- Insertion d'un Tiers
  - Insert
  - InsertV2
- Modification d'un Tiers
  - Update
  - UpdateV2
- Suppression d'un Tiers
  - Delete
  - DeleteV2

j) RegistreRevisionService

- Insertion des registres de révision
  - Insert
  - InsertV2
- Suppression des registres de révision
  - Delete

E. Paramétrage des logs

Le service WebServices100 produit deux familles de journaux complémentaires :

- Le log texte historique (LogManager) : un fichier « .txt » lisible, généré chaque jour à côté de l'exécutable. Il assure le suivi général et l'écriture des exceptions.
- Trois canaux de journalisation structurée au format CLEF (JSON compact) — Applicatif, Performance et Santé. Chacun dispose de son propre indicateur d'activation, de son sous-dossier et de ses fichiers quotidiens.

Ces trois canaux structurés sont indépendants : on active uniquement ce dont on a besoin. Ils sont tous désactivés par défaut.

Les paramètres se règlent dans le fichier de configuration « WebServices100Service.exe.config », lu au démarrage du service. Une valeur absente retombe sur la valeur par défaut indiquée.

1. Emplacement des fichiers

La racine des canaux structurés est %ProgramData%\PPI-Group\WS100\Logs (soit C:\ProgramData\PPI-Group\WS100\Logs sur une installation standard).

Canal	Fichier	Activé par
Applicatif	…\Logs\App\app-AAAAMMJJ.clef	LogAppDiagnostic
Performance	…\Logs\Perf\benchmark-perf-AAAAMMJJ.clef	LogBenchmarkPerf
Santé	…\Logs\Health\benchmark-sante-AAAAMMJJ.clef	LogServiceHealth
Texte (historique)	Logs\AAAAMMJJ.txt (à côté de l'exécutable)	toujours actif

2. Paramètres de configuration

Chaque paramètre ci-dessous se positionne dans le fichier de configuration ; la capture qui suit en illustre l'emplacement.

Paramètre	Type	Défaut	Rôle
LogAppDiagnostic	bool	false	Active le canal Applicatif (diagnostic structuré → app-*.clef).
LogBenchmarkPerf	bool	false	Active le canal Performance : chronométrage de chaque appel Sage → benchmark-perf-*.clef.
LogServiceHealth	bool	false	Active le canal Santé : heartbeat périodique et événements de cycle de vie → benchmark-sante-*.clef.
LogMinLevel	enum	Information	Niveau minimum du canal Applicatif : Debug, Information, Warning ou Error.
HealthCheckIntervalMinutes	int	5	Intervalle entre deux heartbeats de santé, en minutes (plancher : 1 min).
LogRetentionDays	int	30	Durée d'historique conservée pour les trois canaux structurés, en jours

Paramètre	Type	Défaut	Rôle
			(rétention par âge des fichiers).
LogFileSizeLimitMb	int	50	Taille maximale d'un fichier avant roulement par taille (Mo). Au plafond, un nouveau segment …-AAAAMMJJ_001.clef est créé.
LogException	bool	false	Active l'écriture des exceptions dans le log texte historique (LogManager).
LogStackTrace	bool	false	Si LogException est actif, ajoute la pile d'appels (stack trace) au log texte.

3. Écriture et gestion des fichiers

Les trois canaux structurés partagent la même mécanique d'écriture et de roulement, conçue pour ne pas peser sur le traitement métier ni perdre d'événements silencieusement.
Écriture asynchrone — Chaque canal écrit de façon asynchrone : le thread qui journalise dépose l'événement dans une file en mémoire (jusqu'à 50 000 événements) et rend immédiatement la main ; un thread de fond unique vide cette file vers le disque.
L'entrée/sortie disque et le verrou du fichier quittent ainsi le chemin de la requête — utile pour le canal Performance, qui écrit une ligne par appel Sage.
Important : en cas de saturation disque extrême et prolongée, la file peut se remplir ; les événements excédentaires sont alors abandonnés (et non bufferisés indéfiniment) pour ne pas ralentir le métier. L'arrêt propre du service vide la file, mais un arrêt brutal du process peut perdre les événements encore en attente.
Roulement quotidien et par taille — un nouveau fichier est ouvert chaque jour. À l'intérieur d'une même journée, dès qu'un fichier atteint la taille LogFileSizeLimitMb (50 Mo par défaut), l'écriture bascule sur un segment numéroté (benchmark-perf-AAAAMMJJ_001.clef, _002.clef, …). Aucune donnée n'est perdue : on continue simplement dans un nouveau fichier.
Rétention — les fichiers sont conservés par âge : tout fichier plus vieux que LogRetentionDays jours (30 par défaut) est supprimé automatiquement. La rétention est exprimée en durée et non en nombre de fichiers, car le roulement par taille peut produire plusieurs fichiers dans la même journée.
Note : le log texte historique (LogManager) est indépendant de cette mécanique ; il n'est ni asynchrone ni purgé automatiquement.

4. Canal Applicatif (LogAppDiagnostic)

Diagnostic structuré, destiné à comprendre le déroulement fonctionnel. Chaque ligne porte un niveau (Debug, Information, Warning ou Error) filtré par LogMinLevel. Les propriétés varient selon le message journalisé ; certaines sont systématiques :

Propriété	Signification
(niveau)	Debug, Information, Warning ou Error, selon l'importance du message.
WorkingSetMo	RAM physique (working set) du process en Mo — ajoutée par les messages de suivi mémoire, notamment autour des cycles d'ouverture/fermeture des objets Sage.
ThreadId / CallToken / Environnement	Champs de corrélation (voir « Champs de corrélation »).

5. Canal Performance (LogBenchmarkPerf)

Une ligne par appel à l'API passant par la couche de service, avec chronométrage détaillé et empreinte mémoire. Sert à repérer les appels lents et ceux qui font croître la RAM.

Propriété	Signification
Service	Nom de la classe de service WCF (ex. DocumentService).
Methode	Nom de la méthode appelée (capturé automatiquement).
DureeTotaleMs	Durée totale de l'appel : acquisition du SageManager + opération métier + libération (ms).
DureeOuvertureSageMs	Durée d'acquisition du SageManager (ouverture connexion / COM Sage), en ms.
DureeOperationMs	Durée de l'opération métier elle-même, en ms.
WorkingSetMo	RAM physique du process après l'appel, en Mo.
WorkingSetDeltaMo	Variation du working set due à l'appel (après − avant), en Mo. Met en évidence les appels qui font monter la RAM (fuites, surtout COM natif).

Propriété	Signification
AllocManagedKo	Octets alloués sur le tas managé .NET par le thread pendant l'appel, en Ko.
Succes	true si l'appel s'est terminé sans exception, false sinon.
ThreadId / CallToken / Environnement	Champs de corrélation (voir « Champs de corrélation »).

Note : une ligne de performance pèse environ 500 octets ; un fichier journalier atteint donc la limite de 50 Mo aux alentours de 100 000 appels Sage. Au-delà, un nouveau segment est créé (voir « Écriture et gestion des fichiers »).

6. Canal Santé (LogServiceHealth)

Photographie périodique de l'état du process. Deux types de lignes : HEARTBEAT (émis toutes les HealthCheckIntervalMinutes minutes) et LIFECYCLE (ouverture et arrêt des services).

Propriété	Signification
Type	HEARTBEAT (battement périodique) ou LIFECYCLE (ouverture / arrêt des services).
UptimeSec	Durée de fonctionnement du process depuis son démarrage, en secondes.
WorkingSetMo	RAM physique (working set) du process, en Mo.
GcHeapMo	Taille du tas managé .NET (GC heap), en Mo.
Threads	Nombre de threads du process.
AppelsTotal	Nombre total d'appels Sage depuis le démarrage. Indépendant de LogBenchmarkPerf : compté même si la perf est désactivée.
AppelsDepuisDernierTick	Nombre d'appels Sage depuis le heartbeat précédent (0 sur les lignes LIFECYCLE).

Propriété	Signification
Message	« Services ouverts » / « Arrêt du service » pour LIFECYCLE ; vide pour un heartbeat.

Note : le compteur d'appels est un simple entier en mémoire, remis à zéro à chaque redémarrage du service. AppelsTotal reflète donc le volume depuis le dernier démarrage.

7. Champs de corrélation

Ces champs sont ajoutés automatiquement à chaque ligne des canaux Applicatif et Performance (le canal Santé ne les porte pas). Ils permettent de relier entre elles toutes les lignes d'un même appel.

Propriété	Signification
ThreadId	Identifiant du thread managé courant (toujours présent).
CallToken	Identifiant de corrélation (GUID) de l'appel WCF courant. Relie les lignes Applicatif et Performance d'un même appel.
Environnement	Nom de l'environnement Sage de l'appel courant (ex. BIJOU). Présent si renseigné.

8. Lire un fichier .clef

Les fichiers « .clef » sont au format CLEF (Compact Log Event Format) : une ligne = un événement JSON. Outre les propriétés métier décrites plus haut, chaque ligne contient des champs techniques préfixés par « @ » :

Champ	Signification
@t	Horodatage de l'événement (date / heure UTC, ISO 8601).
@mt	Modèle de message (template, avant substitution des valeurs).
@l	Niveau de l'événement. Absent quand le niveau est Information (niveau par défaut).

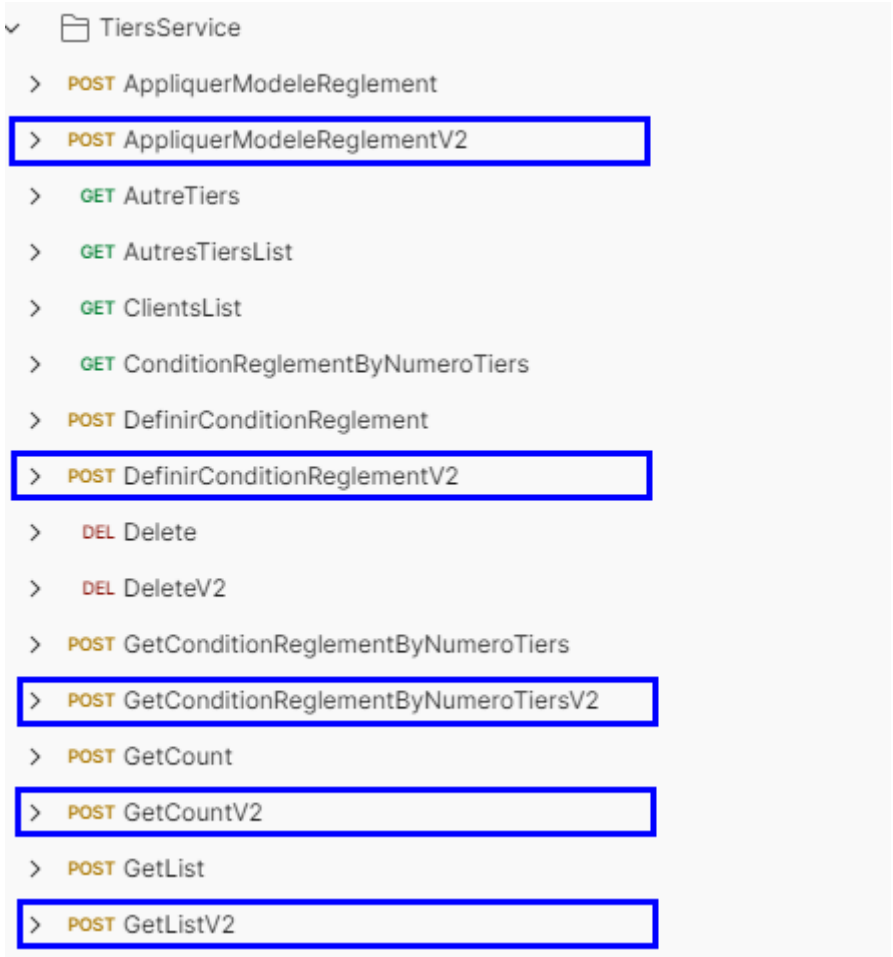
Champ	Signification
@x	Exception complète (message + pile d'appels), si l'événement en porte une.
@i	Identifiant du type d'événement (empreinte du modèle de message).

Ces fichiers se lisent tels quels, mais sont surtout pensés pour être ingérés par un agrégateur (par exemple Seq), qui permet le filtrage et la corrélation par CallToken, Service, Environnement, etc.

F. Gestion du WAF

L'utilisation d'un Web Application Firewall (WAF) se répend de plus en plus.
L'utilisation d'un WAF permet d'améliorer la sécurité des appels web. En effet, le WAF applique des règles de sécurité afin de bloquer des appels qu'il considère comme dangereux.
Les règles du WAF, géré en grande partie par Microsoft, sont en constante évolution afin de garantir un niveau sécurité maximal.
Cependant, certaines règles sont parfois très, voire trop, strictes.
Un appel POST peut être bloqué si le WAF détecte le texte UNION dans le json associé.
Cela rendrait les GetList, avec un criteria contenant le texte Union, inopérants.

Afin de pouvoir contourner ces potentielles restrictions, nous avons ajouté des endpoints. Ces nouveaux endpoints se terminent par V2 (cf notre documentation Postman).



Le fonctionnement de ces endpoints est le suivant :
Ils ont tous un et un seul paramètre nommé data.
Ce paramètre data contient le json des paramètres qui auraient été envoyé au endpoint non V2.

Voici un exemple avec TiersService.GetList.
Ce endpoint peut avoir ces paramètres :

```
{
   "criteria": {
      "__type": "CriteriaComparison:http://www.proconsult.lu/WebServices100",
      "FieldName": "NumeroTiers",
      "Operator": "0",
      "Value": "CARAT"
   },
   "orders": [
      {
         "FieldName":"NumeroTiers",
         "OrderType":"0"
      }
   ],
   "pageNumber": 1,
   "rowsPerPage": 10
}
```

Pour faire l'appel à TiersService.GetListV2, il faudra :

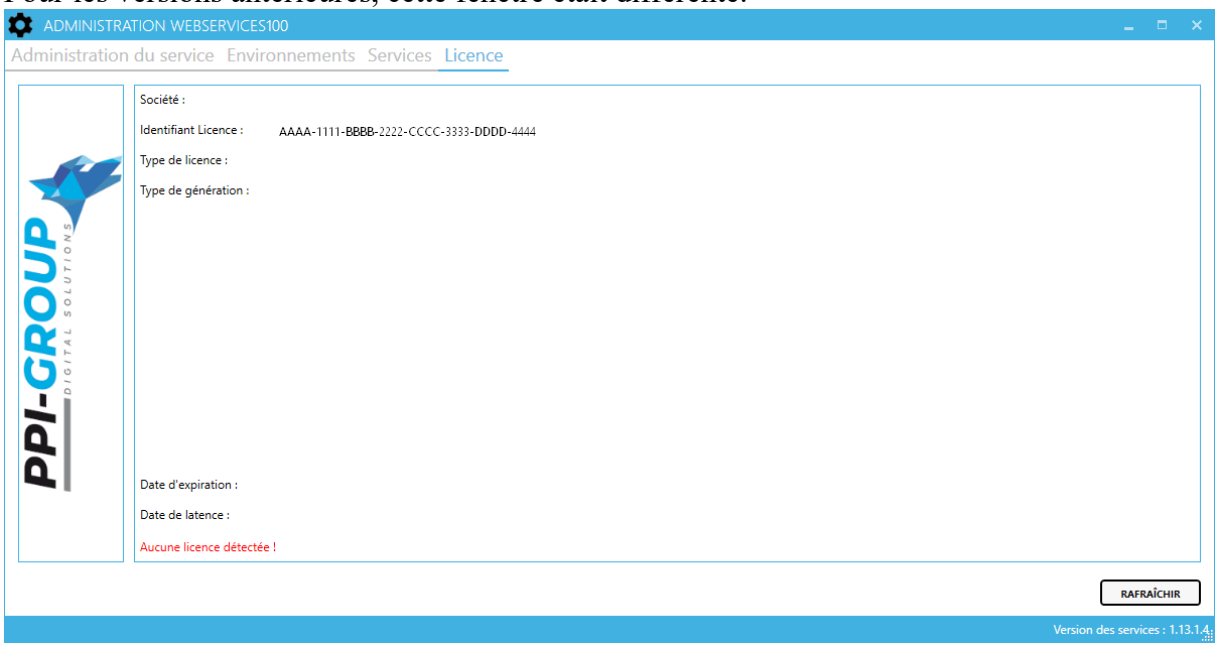
Convertir tout le précédent json, accolades comprises, en base64. Ce qui donne le texte suivant.

ewogICAgImNyaXRlcmlhIjogbnVsbCwKICAgICJvcmRlcnMiOiBudWxsLAogICAgInBhZ2VOdW1iZXIiOiAxLAogICAgInJvd3NQZXJQYWdlIjogMTAKfQ==

Puis mettre ce texte dans le paramètre data du json envoyé.

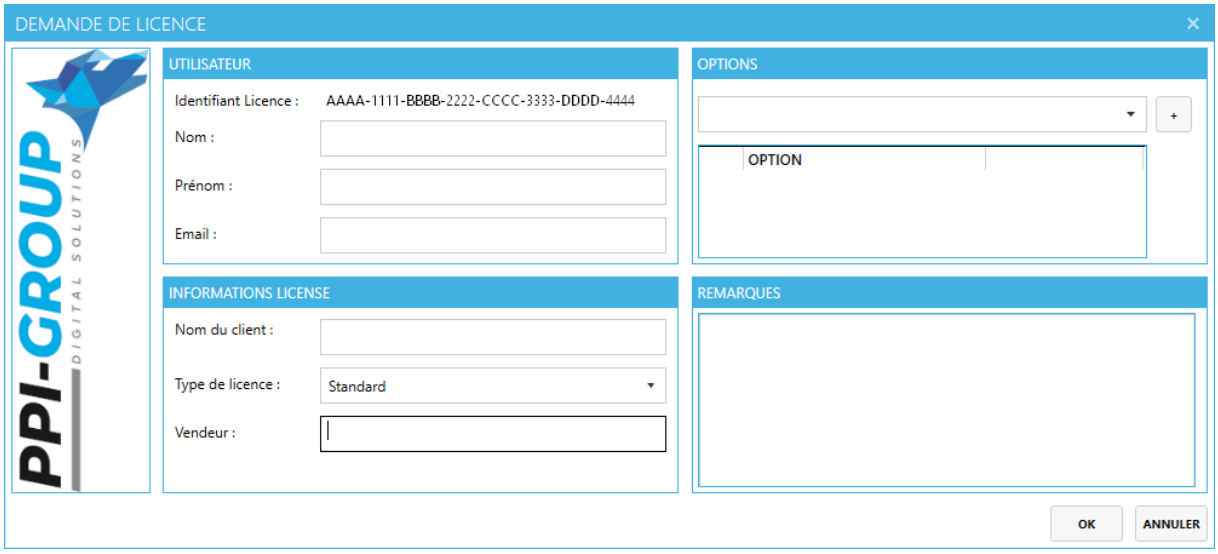
G. Licence

Voici l'onglet de licence. Cette version de cette fenêtre est disponible depuis la version 1.13. Pour les versions antérieures, cette fenêtre était différente.



Au premier lancement, puisque vous n'avez pas de licence connectée, L'administration affichera la fenêtre de saisie.

1. Fenêtre de saisie



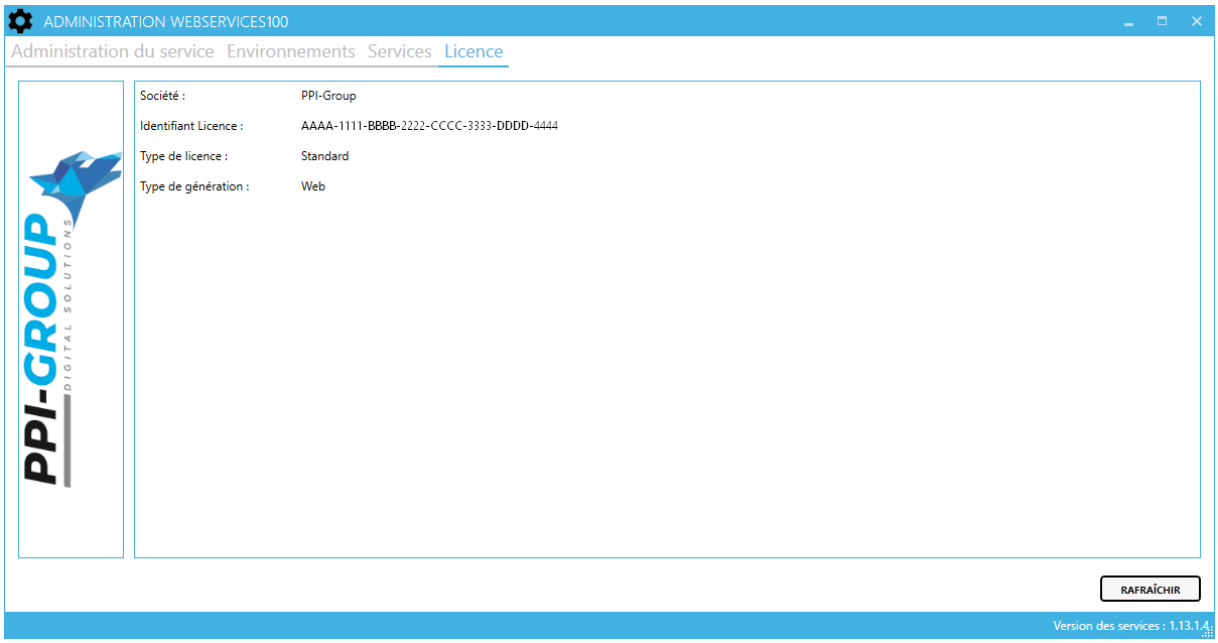
Dans cette fenêtre vous devez saisir les différentes informations :

- Nom, prénom, adresse email du demandeur.
- Nom du client.
- Type de licence : Standard ou démo.
- Nom de la société (revendeur) des WebServices100.

Lorsque ces données seront remplies, cliquer sur « OK ».

Si les données sont correctement remplies, une demande de licence sera envoyée à notre gestionnaire de licences.
Notre équipe sera notifiée de l'arrivée d'une nouvelle demande de licence. Celle-ci sera traitée dans les plus brefs délais.
Lorsque la demande de licence aura été traité, un mail sera envoyé à l'adresse email du demandeur.

Si vous aviez fermé l'administration des WebServices100, il vous suffit de l'ouvrir et la licence se téléchargera automatiquement.
Sinon, il vous suffit de cliquer sur « Rafraîchir ». Cela téléchargera la licence et les informations de licence s'afficheront.



2. Le service refuse de démarrer alors que la licence est valide

Il arrive que la licence soit reconnue comme valide dans l'interface d'administration, mais que le service Windows s'arrête immédiatement au démarrage en signalant une licence invalide.

Origine du problème

La validation de la licence repose sur une empreinte de la machine, calculée à partir de plusieurs informations matérielles. Sur certaines cartes mères, une de ces informations n'est pas une véritable donnée du matériel : Windows la remplace alors par un texte générique (par exemple « Carte de base » ou « Base Board »), traduit selon la langue du compte Windows qui exécute la vérification — et non selon la langue d'installation de Windows.

Or, l'interface d'administration et le service ne s'exécutent pas forcément sous le même compte : l'administration tourne généralement sous un compte utilisateur (souvent en français), tandis que le service tourne sous un compte de service dédié ou un compte système (SYSTEM, Local Service, Network Service), dont la langue par défaut est souvent l'anglais. Si les deux comptes n'ont pas la même langue, l'empreinte calculée diffère pour une seule et même machine, et le service rejette la licence.

Ce décalage est fréquent car un nouveau compte n'hérite pas de la langue du compte qui l'a créé, mais de celle du profil système par défaut — et les comptes système utilisent un profil totalement séparé, jamais affecté par un changement de langue sur un compte utilisateur.

Comment y remédier

Harmoniser la langue d'affichage entre tous les comptes concernés :

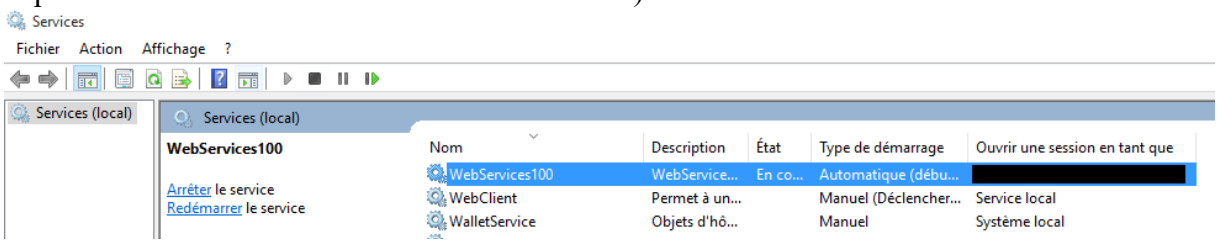
- Compte de service dédié : se connecter à sa session et modifier sa langue via Paramètres → Heure et langue, puis se déconnecter/reconnecter.
- Comptes système : utiliser l'option « Copier les paramètres de l'utilisateur actuel vers l'écran de bienvenue et les comptes système » (Paramètres → Heure et langue → Langue et région), puis redémarrer le poste.

Si le problème persiste malgré cette harmonisation, contactez notre équipe support.

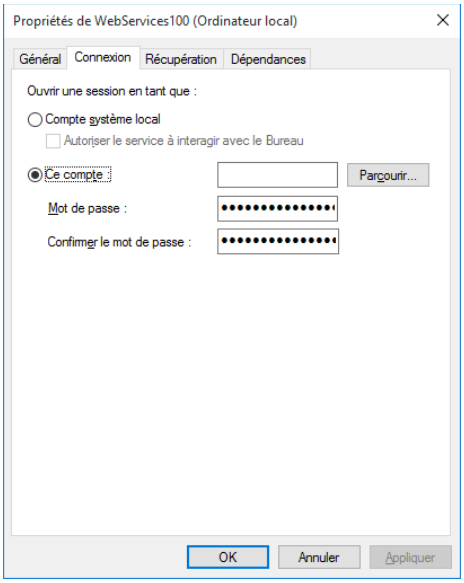
H. Démarrage du Web Service

Pour démarrer le web service, entrez dans l'onglet « Administration du service », puis cliquez sur « Démarrer ».

Si le web service ne démarre pas, cela signifie probablement que le web service n'a pas les droits d'accès. Pour corriger cela, rendez-vous dans les services windows Services (accessible depuis le menu démarrer en recherchant Services).



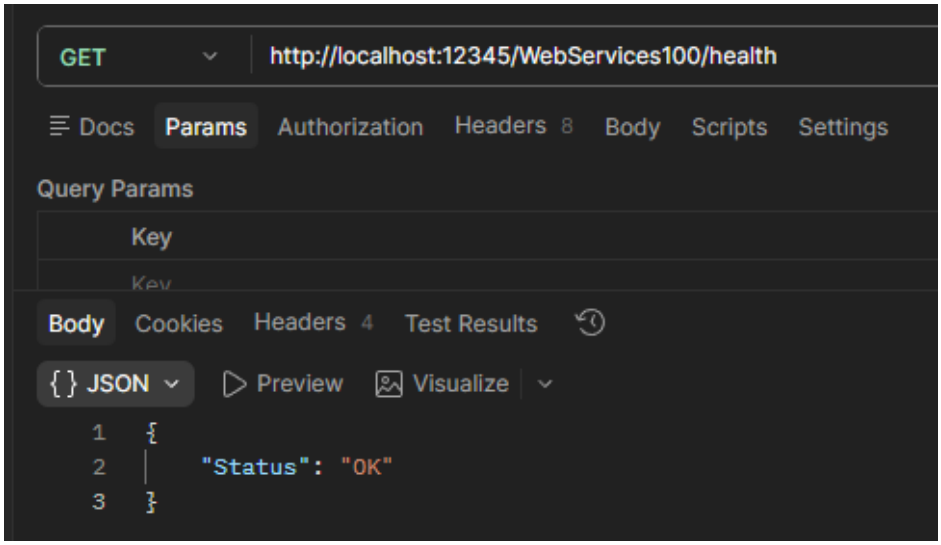
Recherchez, WebServices100, puis clic droit, « Propriétés ».



Dans l'onglet connexion, activez l'option « Ce compte » puis définissez le nom du compte ainsi que le mot de passe.

I. Route Health Check

Afin de vérifier que l'API est opérationnelle, un endpoint REST dédié est disponible : Webservices100/health. Indépendamment des environnements configurés, ce endpoint permet de confirmer que le service est bien démarré et opérationnel.



III. Utilisation du Framework

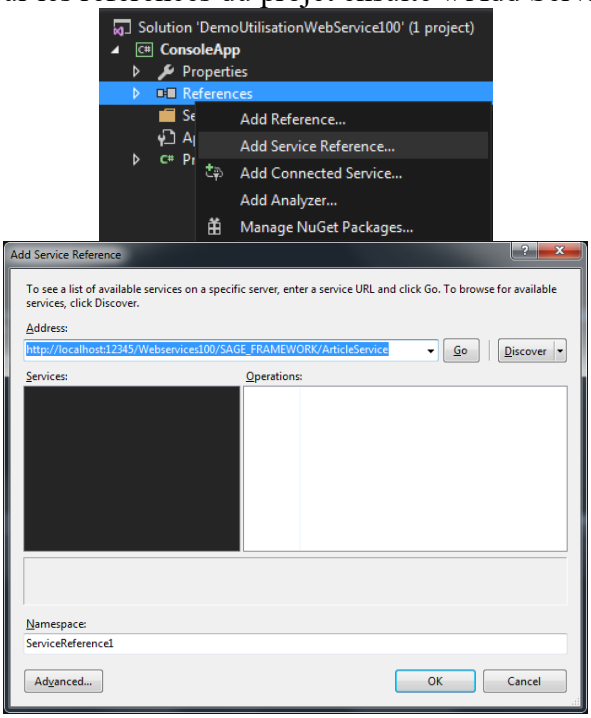
A. Prise en main

1. Exemple .NET

Pour utiliser webServices100 dans les développements .NET il existe 2 solutions. La première en ajoutant des références au service, la seconde en incluant les dll requises.

a) Ajouter les références web services

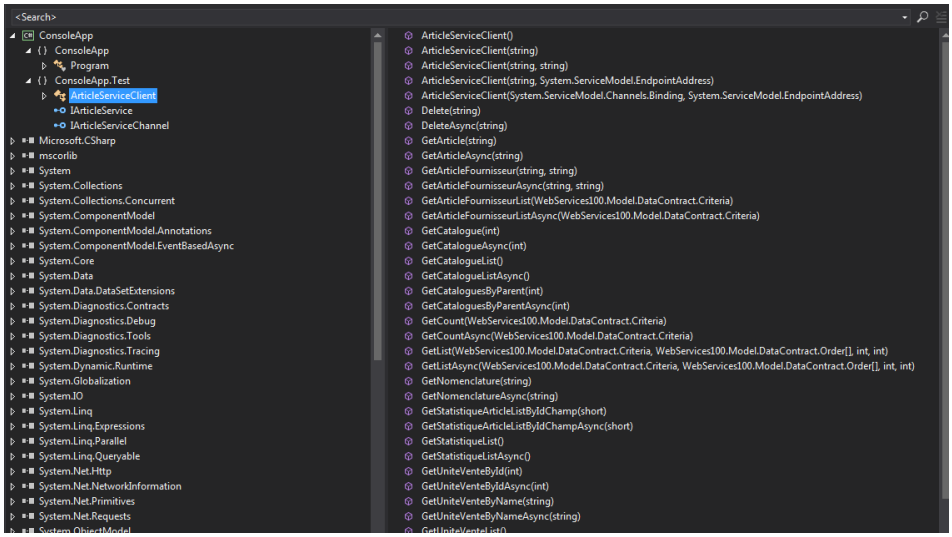
Pour ajouter les références au service, il faut que WebServices100 soit démarré. Puis dans visual Studio clic droit sur les références du projet ensuite « Add Service Référence… ».



Dans la zone de texte « Address », vous saisissez l'Url de WebService100 suivi du nom logique de l'environnement et du service à utiliser.
Dans cet exemple :

- l'url de WebServices100 est « http://localhost:12345/Webservices100/ »,
- le nom logique de l'environnement est « BIJOU »,
- le service utilisé est « ArticleService ».

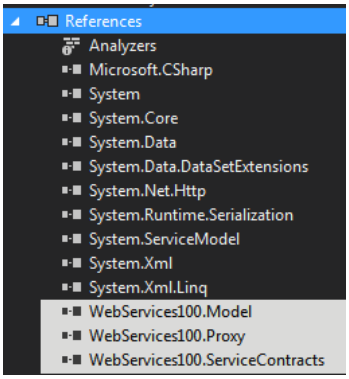
L'adresse est donc « http://localhost:12345/Webservices100/BIJOU/ArticleService ».
Définissez un namespace puis cliquez sur le bouton « Ok ».
Visual studio créé une classe proxy « ArticleServiceClient » contenant l'ensemble des fonctions proposées par le contrat de service « ArticleService ».



b) Ajouter les librairies

La seconde solution consiste à ajouter les librairies nécessaires qui sont :

- WebService100.Model contient les objets DTO
- WebService100.ServiceContracts contient les fonctionnalités
- WebService100.Proxy contient les classes proxy d'appel aux fonctionnalités



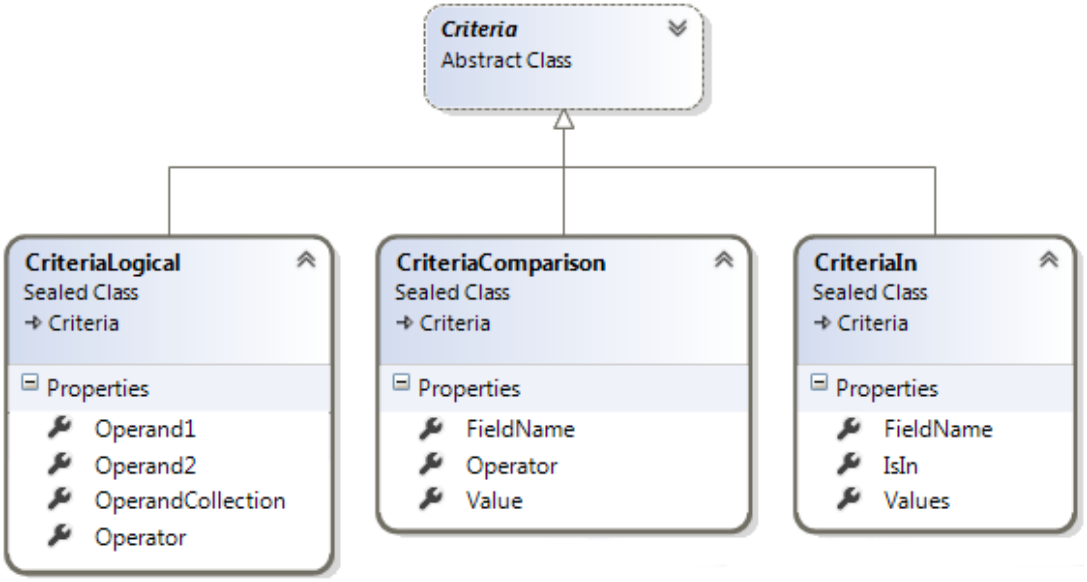
B. Les services

1. Préambule

Afin de bien comprendre comment utiliser les fonctionnalités de lecture depuis Sage, il convient d'expliquer la signification de certains paramètres communs.

a) Criteria

La classe abstraite Criteria permet de définir des filtres de recherche complexe.



CriteriaComparison

La classe CriteriaComparison est destinée à créer un filtre sur un nom de champ.

```
public CriteriaComparison(string fieldName, ComparisonOperator comparisonOperator,
object value);
```

```
public string FieldName { get; set; }
```

La propriété FieldName correspond au nom de la propriété du Dto à utiliser dans le filtre. Par exemple, pour filtrer sur l'email d'un client il faut définir le fieldName sur « Email ».

```
public ComparisonOperator Operator { get; set; }
```

La propriété Operator correspond au type d'opérateur à utiliser (inférieur, inférieur ou égal, égal, différent,…..).

```
public object Value { get; set; }
```

La propriété Value correspond à la valeur de comparaison.

Exemple de criteria pour filtrer les clients dont l'email se termine par « gmail.com ».

```
Criteria criteria = new CriteriaComparison("Email", ComparisonOperator.Like,
"%gmail.com");
```

CriteriaIn

La classe CriteriaIn est destinée à créer un filtre suivant le concept la valeur fait parti, ou ne fait pas parti de la liste des valeurs fournies.

```
public CriteriaIn(string fieldName, bool isIn, IEnumerable<object> values)
```

```
public string FieldName { get; set; }
```

La propriété FieldName correspond au nom de la propriété du Dto à utiliser dans le filtre.

```
public bool IsIn { get; set; }
```

La propriété IsIn permet de préciser si l'on doit conserver, ou exclure, les éléments de la propriété Values.

```
public IEnumerable<object> Values { get; set; }
```

La propriété Values est un tableau contenant l'ensemble des valeurs.

Exemple de criteria pour filtrer les clients dont le pays est un des pays suivants : France, Allemagne,Belgique.

```
Criteria criteria = new CriteriaIn("Pays",true , new Object[]
{"France","Allemagne","Belgique"});
```

CriteriaLogical

La classe CriteriaLogical est destinée à combiner des critères.

```
public CriteriaLogical(LogicalOperator logicalOperator, IEnumerable<Criteria>
operandCollection)
public CriteriaLogical(Criteria operand1, LogicalOperator logicalOperator, Criteria
operand2);                                                    Constructeur obsolète

public Criteria Operand1 { get; set; }                          Propriété obsolète
public Criteria Operand2 { get; set; }                          Propriété obsolète
public IEnumerable<Criteria> OperandCollection { get; set; }
```

La propriété OperandCollection est un tableau contenant l'ensemble des Criteria à combiner.

```
public LogicalOperator Operator { get; set; }
```

La propriété Operator correspond au type d'opérateur (AND ou OR) à appliquer entre les différents criteria de la propriété OperandCollection.

Exemples de criteria pour filtrer les clients dont l'email se termine par « gmail.com » **ET** dont le pays est « France ».

```
Criteria criteria = new CriteriaLogical(
       LogicalOperator.And,
       new Criteria[]
       {
              new CriteriaComparison("Email", ComparisonOperator.Like, "%gmail.com"),
              new CriteriaComparison("Pays", ComparisonOperator.Equals, "France")
       });
```

b) Order

La classe order est destinée à définir un ordre de tri sur la collection obtenu depuis un service.

```
public Order(string fieldName, OrderType type);

public string FieldName { get; set; }
```

La propriété FieldName correspond au nom de la propriété du Dto à utiliser dans le filtre. Par exemple, pour filtrer sur l'email d'un client il faut définir le fieldName sur « Email ».

```
public OrderType OrderType { get; set; }
```

La propriété OrderType définie le sens de tri.

2. Les services communs

a) *CalendrierService*

Fonctions disponibles :

```
▪ Calendrier GetCalendrier(int idCalendrier);
▪ Calendrier Get(string idCalendrier);
▪ List<Calendrier> Calendriers(int pageNumber = 0, int rowsPerPage = 0);
▪ List<Calendrier> GetList(Criteria criteria = null, List<Order> orders = null, int
  pageNumber = 0, int rowsPerPage = 0);
```

Fonction :

```
▪ Calendrier GetCalendrier(int idCalendrier);
```

Description :
Retourne le calendrier par son identifiant.

Exemple :

```
ICalendrierService service = new
CalendrierService("http://localhost:12345/Webservices100/BIJOU/");
Calendrier calendrier = service.GetCalendrier(1);
```

Fonction :

```
▪ Calendrier Get(string idCalendrier);
```

Description :
Retourne le calendrier par son identifiant.

Exemple :

```
ICalendrierService service = new
CalendrierService("http://localhost:12345/Webservices100/BIJOU/");
Calendrier calendrier = service.GetCalendrier("2");
```

Fonction :

```
▪ List<Calendrier> Calendriers(int pageNumber = 0, int rowsPerPage = 0);
```

Description :
Retourne une liste de calendriers correspondant à la page définie par les paramètres pageNumber et rowsPerPage.
Les calendriers sont classés par id croissant.

Exemple :

```
ICalendrierService service = new
CalendrierService("http://localhost:12345/Webservices100/BIJOU/");
List<Calendrier> calendriers = service.Calendriers(2,2);
```

Fonction :

- `List<Calendrier> GetList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);`

Description :

Retourne la liste des calendriers répondants aux critères passés en paramètre.

Exemple :

```
ICalendrierService service = new
CalendrierService("http://localhost:12345/Webservices100/BIJOU/");
Criteria criteria = new CriteriaComparison("JoursOuvres.Samedi",
ComparisonOperator.Equals, true);
List<Calendrier> calendriers = service.GetList(criteria);
```

b) DatabaseStructureService

Fonctions disponibles :

- `bool HaveGescomInstalled();`

Fonction :

`bool HaveGescomInstalled();`

Description :

Retourne true si la gestion commerciale est installée sur la base de données.

Exemple :

```
IDatabaseStructureService service = new DatabaseStructureService
("http://localhost:12345/Webservices100/BIJOU/");
bool gescomInstalled = service.HaveGescomInstalled();
```

Exemple JSON obtenu en retour :

```
true
```

c) FileService

Fonctions disponibles :

- `[Obsolete] Stream GetArticleImage(string refArticle);`
- `Stream PostArticleImage(string refArticle);`
- `bool AddArticleImage(string refArticle, string nomFichier, byte[] fichier);`
- `bool DeleteArticleImage(string refArticle);`
- `Stream GetArticleFichier(int id);`
- `bool AddArticleFichier(string refArticle, string nomFichier, byte[] fichier, string commentaire = "", Guid? guidCreateur = null);`
- `bool DeleteArticleFichier(int id);`
- `Stream GetDocumentFichier(int id);`
- `bool AddDocumentFichier(TypeDocument typeDocument, string numDocument, string nomFichier, byte[] fichier, string intitule, Guid? guidCreateur = null);`
- `bool DeleteDocumentFichier(int id);`
- `FichierLieArticle GetFichierLieArticleById(int id)`
- `List<FichierLieArticle> GetFichiersLiesArticle(string refArticle);`
- `List<FichierLieArticle> GetAllFichiersLiesArticle();`
- `FichierLieDocument GetFichierLieDocumentById(int id);`
- `List<FichierLieDocument> GetFichiersLiesDocument(TypeDocument typeDocument, string numeroDocument);`
- `DocumentFileStream GetDocumentFileStream(Guid guidDocument);`
- `DocumentFileStream GetFactureFileStreamFromDocument(string typeDocument, string numeroDocument);`
- `DocumentFileStreamResult GetFactureFileStreamFromDocuments(GetFilesStreamsRequest getFilesStreamsRequest);`
- `DocumentFileStream GetFactureFileStreamFromEcriture(string idEcriture);`
- `EcritureFileStreamResult GetFactureFileStreamFromEcritures(GetFilesStreamsRequest getFilesStreamsRequest);`

Fonction :

```
Stream GetArticleImage(string refArticle);
```

Description :

Retourne une photo d'un article dont la référence est passée en paramètre.

Cette méthode est désormais obsolète. Utiliser plutôt la méthode PostArticleImage ci-après.

Exemple :

```
IFileService service = new FileService ("http://localhost:12345/Webservices100/BIJOU/");
Stream image = service.GetArticleImage ("REFA1");
```

Fonction :

```
Stream PostArticleImage(string refArticle);
```

Description :

Retourne une photo d'un article dont la référence est passée en paramètre.

Exemple :

```
IFileService service = new FileService ("http://localhost:12345/Webservices100/BIJOU/");
Stream image = service.PostArticleImage("REFA1");
```

Fonction :
```
bool AddArticleImage(string refArticle, string nomFichier, byte[] fichier);
```

Description :
Ajoute une image à un article.

Exemple :
```
IFileService service = new FileService ("http://localhost:12345/Webservices100/BIJOU/");
FileStream stream = File.OpenRead("D:\Images\img1.png");
byte[] fileBytes = new byte[stream.Length];
stream.Read(fileBytes, 0, fileBytes.Length);
stream.Close();
service.AddArticleImage("BAOR01","image1.png", fileBytes);
```

Fonction :
```
bool DeleteArticleImage(string refArticle);
```

Description :
Supprime une photo d'un article dont la référence est passée en paramètre.
Retourne true si la photo a été supprimée.

Exemple :
```
IFileService service = new FileService ("http://localhost:12345/Webservices100/BIJOU/");
service.DeleteArticleImage("REFA1");
```

Fonction :
```
Stream GetArticleFichier(int id);
```

Description :
Retourne un fichier lié à un article. L'identifiant est passé en paramètre.

Exemple :
```
IFileService service = new FileService ("http://localhost:12345/Webservices100/BIJOU/");
Stream fichier = service.GetArticleFichier(1);
```

Fonction :
```
bool AddArticleFichier(string refArticle, string nomFichier, byte[] fichier, string
commentaire = "", Guid? guidCreateur = null);
```

Description :
Ajoute un fichier lié à un article.
Retourne true si l'ajout est réussi.

Exemple :
```
IFileService service = new FileService ("http://localhost:12345/Webservices100/BIJOU/");
FileStream stream = File.OpenRead("D:\Documents\doc1.pdf");
byte[] fileBytes = new byte[stream.Length];
stream.Read(fileBytes, 0, fileBytes.Length);
stream.Close();
Guid? guidCreateur = null;//Ou obtenu via ConnectUser
service. AddArticleFichier("BAOR01","doc1.pdf", fileBytes, "commentaires",
guidCreateur);
```

Fonction :
```
bool DeleteArticleFichier(int id);
```

Description :
Supprime un fichier lié à un article. L'identifiant est passé en paramètre.
Retourne true si le fichier a été supprimé

Exemple :
```
IFileService service = new FileService ("http://localhost:12345/Webservices100/BIJOU/");
service.DeleteArticleFichier(1);
```

Fonction :
```
Stream GetDocumentFichier(int id);
```

Description :
Retourne un fichier lié à un document. L'identifiant est passé en paramètre.

Exemple :
```
IFileService service = new FileService ("http://localhost:12345/Webservices100/BIJOU/");
Stream fichier = service.GetDocumentFichier(1);
```

Fonction :
```
bool AddDocumentFichier(TypeDocument typeDocument, string numDocument, string
nomFichier, byte[] fichier, string intitule, Guid? guidCreateur = null);
```

Description :
Ajoute un fichier lié à un document.

Exemple :
```
IFileService service = new FileService ("http://localhost:12345/Webservices100/BIJOU/");
FileStream stream = File.OpenRead("D:\Documents\doc1.pdf");
byte[] fileBytes = new byte[stream.Length];
stream.Read(fileBytes, 0, fileBytes.Length);
stream.Close();
Guid? guidCreateur = null;//Ou obtenu via ConnectUser
service.AddDocumentFichier(TypeDocument.BonDeCommandeVente, "BC00102","doc1.pdf",
fileBytes, "conditions");
```

Fonction :
```
bool DeleteDocumentFichier(int id);
```

Description :
Supprime un fichier lié à un document. L'identifiant est passé en paramètre.
Retourne true si le fichier a été suprimé

Exemple :
```
IFileService service = new FileService ("http://localhost:12345/Webservices100/BIJOU/");
service.DeleteDocumentFichier(1);
```

Fonction :
```
FichierLieArticle GetFichierLieArticleById(int id)
```

Description :
Retourne un objet décrivant un fichier lié à un article. L'identifiant est passé en paramètre.

Exemple :
```
IFileService service = new FileService ("http://localhost:12345/Webservices100/BIJOU/");
FichierLieArticle fichierLie = service.GetFichierLieArticleById(1);
```

Fonction :
```
List<FichierLieArticle> GetFichiersLiesArticle(string refArticle);
```

Description :
Retourne une liste d'objets décrivant les fichiers liés à un article dont la référence est passée en paramètre.

Exemple :
```
IFileService service = new FileService ("http://localhost:12345/Webservices100/BIJOU/");
List<FichierLieArticle> fichierLieList = service.GetFichiersLiesArticle("REFAA1");
```

Fonction :
```
List<FichierLieArticle> GetAllFichiersLiesArticle();
```

Description :
Retourne une liste d'objets décrivant les fichiers liés à tous les articles.

Exemple :
```
IFileService service = new FileService ("http://localhost:12345/Webservices100/BIJOU/");
List<FichierLieArticle> fichierLieList = service.GetAllFichiersLiesArticle();
```

Fonction :
```
FichierLieDocument GetFichierLieDocumentById(int id)
```

Description :
Retourne un objet décrivant un fichier lié à un document. L'identifiant est passé en paramètre.

Exemple :
```
IFileService service = new FileService ("http://localhost:12345/Webservices100/BIJOU/");
FichierLieDocument fichierLie = service.GetFichierLieDocumentById(1);
```

Fonction :
```
List<FichierLieDocument> GetFichiersLiesDocument(TypeDocument typeDocument, string
numeroDocument);
```

Description :
Retourne une liste d'objets décrivant les fichiers liés à un document dont le type et le numéro sont passés en paramètre.

Exemple :
```
IFileService service = new FileService ("http://localhost:12345/Webservices100/BIJOU/");
List<FichierLieDocument> fichierLieList =
service.GetFichiersLiesDocument(TypeDocument.BonDeCommandeVente,"DOCAA1");
```

Fonction :
```
DocumentFileStream GetDocumentFileStream(Guid guidDocument);
```

Description :
Retourne le document Filestream correspondant à l'identifiant du Filestream passé en paramètre.

Exemple :
```
IFileService fileService = new
FileService("http://localhost:12345/Webservices100/BIJOU/");
DocumentFileStream fileStream = fileService.GetDocumentFileStream(Guid.Parse("8C825748-
E062-4FEC-9032-45B82511A287"));
```

Fonction :
```
DocumentFileStream GetFactureFileStreamFromDocument(string typeDocument, string
numeroDocument);
```

Description :
Retourne le document Filestream correspondant au type de document et numéro de document passés en paramètre.

Exemple :
```
IFileService fileService = new
FileService("http://localhost:12345/Webservices100/BIJOU/");
DocumentFileStream fileStream = fileService.GetFactureFileStreamFromDocument("6",
"FA0007");
```

Fonction :
```
DocumentFileStreamResult GetFactureFileStreamFromDocuments(GetFilesStreamsRequest
getFilesStreamsRequest);
```

Description :
Retourne une liste de FileStream pour les documents qui répondent aux critères de recherche fournis en entrée. Ces critères sont spécifiés dans le DTO d'entrée GetFileStreamsRequest.
Ce DTO d'entrée contient les paramètres suivants :

- Criteria : filtre les documents à traiter ;
- InclureElementsSansFileStreamInResponse : booléen indiquant l'inclusion ou non des éléments sans Filestream dans la réponse ;
- PageNumber : numéro de page (élément de pagination) ;
- RowsPerPage : nombre d'éléments retournés en 1 appel (élément de pagination).
- Orders : détermine le mode de tri des éléments du DocumentFileStreamResult. Les listes d'éléments avec et sans FileStream sont triées selon les critères d'ordre fournis en paramètre.

Selon la taille des Filestream, l'amplitude du Criteria de recherche et la pagination utilisée, l'erreur suivante peut survenir :
"Le nombre maximal d'éléments pouvant être traités est de : XXX ! Adapter la pagination !".
Dans ce cas il faudra adapter la valeur des propriétés "PageNumber" et "RowsPerPage".

Le DTO de retour contient les informations suivantes :

- Le nombre total de documents correspondant au Criteria, sans tenir compte de la pagination (NombreElementsSansPagination) ;

- Le nombre total de documents correspondant au Criteria, en tenant compte des éléments de pagination (NombreElementsAvecPagination) ;
- Les éléments de pagination utilisée ;
- Une liste des documents disposant d'un FileStream ;
- Une liste des documents n'ayant pas de FileStream (regroupés par type de document).

Exemple :

```
IFileService fileService = new
FileService("http://localhost:12345/Webservices100/BIJOU/");
GetFilesStreamsRequest request = new GetFilesStreamsRequest()
{
    Criteria = new CriteriaComparison(nameof(Document.TypeDocument),
ComparisonOperator.Equals, TypeDocument.FactureVente),
    InclureElementsSansFileStreamInResponse = true,
    Orders = new List<Order> { new Order { FieldName = nameof(Document.NumeroDocument),
OrderType = OrderType.Asc}},
    PageNumber = 1,
    RowsPerPage = 50,
};
DocumentFileStreamResult fileStreamResult =
fileService.GetFactureFileStreamFromDocuments(request);
```

Exemple utilisant la pagination :

```
IFileService fileService = new
FileService("http://localhost:12345/Webservices100/BIJOU/");

int pageNumber = 1;
int rowsPerPage = 10;

List<DocumentAvecFileStream> resultsAvecFileStream = new List<DocumentAvecFileStream>();
List<DocumentSansFileStream> resultsSansFileStream = new List<DocumentSansFileStream>();

GetFilesStreamsRequest request = GetDocumentFilesStreamsRequest(pageNumber,
rowsPerPage);
int totalNbOfElementsTraites = 0;

DocumentFileStreamResult result = null;
do
{
    result = fileService.GetFactureFileStreamFromDocuments(request);

    resultsSansFileStream.AddRange(result.ElementsSansFileStream);
    resultsAvecFileStream.AddRange(result.ElementsAvecFileStream);

    totalNbOfElementsTraites += result.NombreElementsAvecPagination;
    request.PageNumber++;

} while (totalNbOfElementsTraites < result.NombreElementsSansPagination);


private GetFilesStreamsRequest GetDocumentFilesStreamsRequest(int pageNumber = 0, int
rowsPerPage = 0)
{
    Criteria criteriaFactureVente = new
CriteriaComparison(nameof(Document.TypeDocument), ComparisonOperator.Equals,
TypeDocument.FactureVente);
    GetFilesStreamsRequest request = new GetFilesStreamsRequest()
```

```
    {
        Criteria = criteriaFactureVente,
        RowsPerPage = rowsPerPage,
        PageNumber = pageNumber,
        InclureElementsSansFileStreamInResponse = true
    };
    return request;
}
```

Note : le Criteria du DTO d'entrée GetFileStreamsRequest ne doit pas être modifié au sein de la boucle while. En effet, le nombre d'éléments retournés sans pagination pourrait être impacté, car le critère de recherche ne serait plus identique au critère initial.

▪

Fonction :

```
DocumentFileStream GetFactureFileStreamFromEcriture(string idEcriture);
```

Description :

Retourne le document Filestream correspondant à un numéro d'écriture comptable passé en paramètre.

Exemple :

```
IFileService fileService = new
FileService("http://localhost:12345/Webservices100/BIJOU/");
DocumentFileStream fileStream = fileService.GetFactureFileStreamFromEcriture("248");
```

Fonction :

```
EcritureFileStreamResult GetFactureFileStreamFromEcritures(GetFilesStreamsRequest
getFilesStreamsRequest);
```

Description :

Retourne une liste de FileStream pour les écritures comptables qui répondent aux critères de recherche fournis en entrée. Ces critères sont spécifiés dans le DTO d'entrée GetFileStreamsRequest.

Ce DTO d'entrée contient les paramètres suivants :

- Criteria : filtre les écritures à traiter ;
- InclureElementsSansFileStreamInResponse : booléen indiquant l'inclusion ou non des éléments sans Filestream dans la réponse ;
- PageNumber : numéro de page (élément de pagination) ;
- RowsPerPage : nombre d'éléments retournés en 1 appel (élément de pagination).
- Orders : détermine le mode de tri des éléments de l'EcritureFileStreamResult. Les listes d'éléments avec et sans FileStream sont triées selon les critères d'ordre fournis en paramètre.

Selon la taille des Filestream, l'amplitude du Criteria de recherche et la pagination utilisée, l'erreur suivante peut survenir :
"Le nombre maximal d'éléments pouvant être traités est de : XXX ! Adapter la pagination !".
Dans ce cas il faudra adapter la valeur des propriétés "PageNumber" et "RowsPerPage".

Le DTO de retour contient les informations suivantes :

- Le nombre total d'écritures correspondant au Criteria, sans tenir compte de la pagination (NombreElementsSansPagination) ;

- Le nombre total d'écritures correspondant au Criteria, en tenant compte des éléments de pagination (NombreElementsAvecPagination) ;
- Les éléments de pagination utilisée ;
- Une liste des écritures disposant d'un FileStream ;
- Une liste des écritures n'ayant pas de FileStream (regroupés par mois, années, code journal et numéro de pièce).

Exemple :

```
IFileService fileService = new
FileService("http://localhost:12345/Webservices100/BIJOU/");
GetFilesStreamsRequest request = new GetFilesStreamsRequest()
{
    Criteria = new CriteriaComparison(nameof(Ecriture.CodeJournal),
ComparisonOperator.Equals, "ACH"),
    InclureElementsSansFileStreamInResponse = true,
    Orders = new List<Order> { new Order { FieldName = nameof(Ecriture.Id),
OrderType = OrderType.Asc}},
    PageNumber = 1,
    RowsPerPage = 50,
};
EcritureFileStreamResult fileStreamResult =
fileService.GetFactureFileStreamFromEcritures(request);

Exemple utilisant la pagination :

IFileService fileService = new
FileService("http://localhost:12345/Webservices100/BIJOU/");

int pageNumber = 0;
int rowsPerPage = 0;
int totalNbOfElementsTraites = 0;

List<EcritureAvecFileStream> resultsAvecFileStream = new
List<EcritureAvecFileStream>();
List<EcrituresSansFileStream> resultsSansFileStream = new
List<EcrituresSansFileStream>();

GetFilesStreamsRequest request = GetEcrituresFilesStreamsRequest(pageNumber,
rowsPerPage);

EcritureFileStreamResult result =
fileService.GetFactureFileStreamFromEcritures(request);
int nbElementsSansPagination = result.NombreElementsSansPagination;

while (totalNbOfElementsTraites < nbElementsSansPagination)
{
    result = fileService.GetFactureFileStreamFromEcritures(request);

    resultsSansFileStream.AddRange(result.ElementsSansFileStream);
    resultsAvecFileStream.AddRange(result.ElementsAvecFileStream);

    totalNbOfElementsTraites += result.NombreElementsAvecPagination;
    request.PageNumber++;
}

private GetFilesStreamsRequest GetEcrituresFilesStreamsRequest(int pageNumber = 0,
int rowsPerPage = 0)
{
    Criteria criteriaCodeJournal = new
CriteriaComparison(nameof(Ecriture.CodeJournal), ComparisonOperator.Equals, "VTE");
```

```
    GetFilesStreamsRequest request = new GetFilesStreamsRequest()
    {
        Criteria = criteriaCodeJournal,
        RowsPerPage = rowsPerPage,
        PageNumber = pageNumber,
        InclureElementsSansFileStreamInResponse = true
    };
    return request;
}
```

Note : le Criteria du DTO d'entrée GetFileStreamsRequest ne doit pas être modifié au sein de la boucle while. En effet, le nombre d'éléments retournés sans pagination pourrait être impacté, car le critère de recherche ne serait plus identique au critère initial.

d) PreferencesService

Fonctions disponibles :

- `Preferences GetPreferences();`
- `Preferences Preferences();`

Fonction :

- `Preferences GetPreferences ();`

Description :

Retourne une instance des préférences.

Exemple :

```
IPreferencesService service = new
PreferencesService("http://localhost:12345/Webservices100/BIJOU/");
Preferences preferences= service.GetPreferences();
```

Exemple JSON obtenu en retour :

```
{
    "UnitePoids": 3
}
```

Fonction :

- `Preferences Preferences ();`

Description :

Retourne une instance des préférences.

Exemple :

```
IPreferencesService service = new
PreferencesService("http://localhost:12345/Webservices100/BIJOU/");
```

```
Preferences preferences= service.GetPreferences();
```

Exemple JSON obtenu en retour :


```
{
    "UnitePoids": 3
}
```

e) RisqueService

Fonctions disponibles :

- `Risque GetByIndice(short indice);`
- `List<Risque> GetList();`

Fonction :

- `Risque GetByIndice(short indice);`

Description :

Retourne un risque par son indice.

Exemple :

```
IRisqueService service = new
RiqueService("http://localhost:12345/Webservices100/BIJOU/");
Risque risque = service.GetByIndice(1);
```

Exemple JSON obtenu en retour :

```
{
    "Id": 1,
    "Intitule": "Bonne cote crédit",
    "Min": 0,
    "Max": 1.000000,
    "Action": 0
}
```

Fonction :

- `List<Risque> GetList();`

Description :

Retourne la liste des risques.

Exemple :

```
IRisqueService service = new
RisqueService("http://localhost:12345/Webservices100/BIJOU/");
List<Risque> list = service.GetList();
```

Exemple JSON obtenu en retour :


```
[
    {
        "Id": 1,
        "Intitule": "Bonne cote crédit",
        "Min": 0,
        "Max": 1.000000,
        "Action": 0
    },
    {
        "Id": 2,
        "Intitule": "En cours normal",
        "Min": 2.000000,
        "Max": 9999.000000,
        "Action": 0
    },
    {
        "Id": 3,
        "Intitule": "En dépassement",
        "Min": 10000.000000,
        "Max": 74999.000000,
        "Action": 1
    },
    {
        "Id": 4,
        "Intitule": "Insolvable",
        "Min": 75000.000000,
        "Max": 999999999999.000000,
        "Action": 2
    }
]
```

f) SqlService

Fonctions disponibles :

- `DataTable GetData(string query, List<SqlParameter> parameters = null);`
- `void ExecuteQuery(string query, List<SqlParameter> parameters = null);`

Fonction :

```
DataTable GetData(string query);
```

Description :

Permet de récupérer les données d'une requête SQL sous forme d'une DataTable.
Lève une ArgumentException si le paramètre query est nul.

Exemple :

```
ISqlService service = new SqlService("http://localhost:12345/Webservices100/BIJOU/");
string query = "SELECT DO_DATE AS CA_DATE, SUM(DO_TotalHT) AS CA_AMOUNT FROM F_DOCENTETE
WHERE DO_Domaine = @Domaine AND YEAR(DO_DATE) = @Date GROUP BY DO_DATE ORDER BY DO_DATE;
List<SqlParameter> parameters = New List<SqlParameter>()
{
    New SqlParameter("@Domaine", SqlDbType.SmallInt, 0),
    New SqlParameter("@Date", SqlDbType.Int, 2021)
};
DataTable table = service.GetData(query, parameters);
```

Retourne un objet DataTable sérialisé en XML sour forme d'une chaîne de caractères.

Fonction :

```
void ExecuteQuery(string query, List<SqlParameter> parameters = null);
```

Description :

Permet d'exécuter une requête sur une base de données SQL.
Lève une ArgumentException si le paramètre query est nul.

Exemple :

```
ISqlService service = new SqlService("http://localhost:12345/Webservices100/BIJOU/");
string query = "UPDATE F_ARTICLE SET AR_SOMMEIL = @EstSomeil WHERE AR_REF NOT IN (SELECT
DISTINCT AR_REF FROM F_DOCLIGNE WHERE DO_TYPE IN (@Type1,@Type2) AND YEAR(DO_DATE) >
@Date)";
List<SqlParameter> parameters = New List<SqlParameter>()
{
    New SqlParameter("@EstSomeil", SqlDbType.SmallInt, 1),
    New SqlParameter("@Type1", SqlDbType.Int, 6),
    New SqlParameter("@Type2", SqlDbType.Int, 7)
    New SqlParameter("@Date", SqlDbType.Int, 2021)
};
service.ExecuteQuery(query, parameters);
```

g) UserService

Fonctions disponibles :

- `bool CheckUserPassword(TypeApplicationSage typeApplicationSage, string user, string password);`
- `void ChangeUserPassword(TypeApplicationSage typeApplicationSage, string user, string oldPassword, string newPassword);`
- `UtilisateurSage ConnectUser(TypeApplicationSage typeApplicationSage, string user, string password);`

Fonction :

```
bool CheckUserPassword(TypeApplicationSage typeApplicationSage, string user, string password);
```

Description :

Vérifie le mot de passe de l'utilisateur dans le type d'application Sage passé en paramètre.
Retourne 'true' si le mot de passe est correct ; sinon 'false'.
Lève une ArgumentException si le paramètre user est nul.

Exemple :

```
string login = "<Administrateur>";
string pwd = "test";
IUserService service = new UserService("http://localhost:12345/Webservices100/BIJOU/");
bool result = service.CheckUserPassword(TypeApplicationSage.GestionCommerciale, login, pwd);
```

Fonction :

```
void ChangeUserPassword(TypeApplicationSage typeApplicationSage, string user, string oldPassword, string newPassword);
```

Description :

Modifie le mot de passe de l'utilisateur dans l'application typeApplicationSage.
Lève une exception si le paramètre user est nul.
Lève une exception s'il n'existe pas d'utilisateur avec les paramètres user et oldPassword.

Exemple :

```
string login = "<Administrateur>";
string pwd = "test";
string newPwd = "test2";
IUserService service = new UserService("http://localhost:12345/Webservices100/BIJOU/");
service.ChangeUserPassword(TypeApplicationSage.Comptabilite, login, pwd, newPwd);
```

Fonction :

```
UtilisateurSage ConnectUser(TypeApplicationSage typeApplicationSage, string user, string password);
```

Description :

Renvoie une instance de `UtilisateurSage` si la paire {user, password) est valide pour le type d'application Sage spécifié.
Renvoi null dans le cas contraire.

Exemple :

```
string login = "<Administrateur>";
string pwd = "test";
IUserService service = new UserService("http://localhost:12345/Webservices100/BIJOU/");
UtilisateurSage utilisateur = service.ConnectUser(TypeApplicationSage.Comptabilite, login, pwd);
```

h) UtilitiesService

Fonctions disponibles :

- `Statistique GetByIndice(int indice);`
- `bool IsAbonnementEnteteInUse(int idAbonnement);`
- `bool IsArticleInUse(string refArticle);`
- `bool IsCollaborateurInUse(string nom, string prenom);`
- `bool IsCompteAnalytiqueInUse(short idPlanAnalytique, string numCompteAnalytique);`
- `bool IsCompteGeneralInUse(string numCompte);`
- `bool IsDepotInUse(int idDepot);`
- `bool IsDocumentInUse(TypeDocument typeDocument, string numeroDocument);`
- `bool IsFamilleInUse(string codeFamille);`
- `bool IsJournalInUse(string codeJournal);`
- `bool IsJournalSaisieInUse(string codeJournal, int mois, int annee);`
- `bool IsTaxeInUse(string codeTaxe);`
- `bool IsTiersInUse(string numeroTiers);`

Fonction :

```
bool IsAbonnementInUse (int idAbonnement);
```

Description :

Retourne True si l'abonnement est en cours d'utilisation.
Lève une ArgumentException si l'identifiant d'abonnement n'existe pas.

Exemple :

```
IUtilitiesService service = new
UtilitiesService("http://localhost:12345/Webservices100/BIJOU/");
bool abonnementInUse = service.IsAbonnementInUse(12);
```

Fonction :

```
bool IsAbonnementEnteteInUse (int idAbonnement);
```

Description :

Retourne True si l'entête d'abonnement est en cours d'utilisation.
Lève une ArgumentException si l'identifiant d'abonnement n'existe pas.

Exemple :

```
IUtilitiesService service = new
UtilitiesService("http://localhost:12345/Webservices100/BIJOU/");
bool enteteAbonnementInUse = service.IsAbonnementEnteteInUse(12);
```

Fonction :
```
bool IsArticleInUse(string refArticle);
```

Description :
Retourne True si l'article est en cours d'utilisation.
Lève une ArgumentException si la référence de l'article n'existe pas ou si le paramètre refArticle est nul.

Exemple :
```
IUtilitiesService service = new
UtilitiesService("http://localhost:12345/Webservices100/BIJOU/");
bool articleInUse = service.IsArticleInUse("BAOR01");
```

Fonction :
```
bool IsCollaborateurInUse(string nom, string prenom);
```

Description :
Retourne True si le collaborateur est en cours d'utilisation.
Lève une ArgumentException si le collaborateur n'existe pas ou si le paramètre nom est nul.

Exemple :
```
IUtilitiesService service = new
UtilitiesService("http://localhost:12345/Webservices100/BIJOU/");
bool collaborateurInUse = service.IsCollaborateurInUse ("MARTIN", "Didier");
```

Fonction :
```
bool IsCompteAnalytiqueInUse(short idPlanAnalytique, string numCompteAnalytique);
```

Description :
Retourne True si le compte analytique est en cours d'utilisation.
Lève une ArgumentException si le compte analytique n'existe pas ou si le paramètre numCompteAnalytique est nul.

Exemple :
```
IUtilitiesService service = new
UtilitiesService("http://localhost:12345/Webservices100/BIJOU/");
bool compteAnalytiqueInUse = service.IsCompteAnalytiqueInUse(1, "ENF30");
```

Fonction :
```
bool IsCompteGeneralInUse(string numCompte);
```

Description :
Retourne True si le compte général est en cours d'utilisation.
Lève une ArgumentException si le compte général n'existe pas ou si le paramètre numCompte est nul.

Exemple :
```
IUtilitiesService service = new
UtilitiesService("http://localhost:12345/Webservices100/BIJOU/");
bool compteGeneralInUse = service.IsCompteGeneralInUse("44111000");
```

Fonction :
```
bool IsDepotInUse(int idDepot);
```

Description :
Retourne True si le dépôt est en cours d'utilisation.
Lève une ArgumentException si le dépôt est en cours d'utilisation.

Exemple :
```
IUtilitiesService service = new
UtilitiesService("http://localhost:12345/Webservices100/BIJOU/");
bool depotInUse = service.IsDepotInUse(1);
```

Fonction :
```
bool IsDocumentInUse(TypeDocument typeDocument, string numeroDocument);
```

Description :
Retourne True si le document est en cours d'utilisation.
Lève une ArgumentException si le document n'existe pas ou si le paramètre numeroDocument est nul.

Exemple :
```
IUtilitiesService service = new
UtilitiesService("http://localhost:12345/Webservices100/BIJOU/");
bool docInUse = service.IsDocumentInUse(TypeDocument.PreparationDeLivraisonVente,
"PL00006");
```

Fonction :
```
bool IsFamilleInUse(string codeFamille);
```

Description :
Retourne True si la famille est en cours d'utilisation.
Lève une ArgumentException si la famille n'existe pas ou si le paramètre codeFamille est nul.

Exemple :
```
IUtilitiesService service = new
UtilitiesService("http://localhost:12345/Webservices100/BIJOU/");
bool familleInUse = service.IsFamilleInUse("BIJOUXARG");
```

Fonction :
```
bool IsJournalInUse(string codeJournal);
```

Description :
Retourne True si le journal est en cours d'utilisation.
Lève une ArgumentException si le journal n'existe pas ou si le paramètre codeJournal est nul.

Exemple :
```
IUtilitiesService service = new
UtilitiesService("http://localhost:12345/Webservices100/BIJOU/");
bool journalInUse = service.IsJournalInUse("ACH");
```

Fonction :
```
bool IsJournalSaisieInUse(string codeJournal, int mois, int annee);
```

Description :
Retourne True si le journal de saisie est en cours d'utilisation.
Lève une ArgumentException si l'index de mois est incorrect ou si le codeJournal name est nul.

Exemple :
```
IUtilitiesService service = new
UtilitiesService("http://localhost:12345/Webservices100/BIJOU/");
bool journalSaisieInUse = service.IsJournalSaisieInUse("ACH",3,2015);
```

Fonction :
```
bool IsTaxeInUse(string codeTaxe);
```

Description :
Retourne True si la taxe est en cours d'utilisation.
Lève une ArgumentException si la taxe n'existe pas.

Exemple :
```
IUtilitiesService service = new
UtilitiesService("http://localhost:12345/Webservices100/BIJOU/");
bool taxeInUse = service.IsTaxeInUse("D21");
```

Fonction :
```
bool IsTiersInUse(string numeroTiers);
```

Description :
Retourne True si le tiers est en cours d'utilisation.
Lève une ArgumentException si le tiers n'existe pas.

Exemple :
```
IUtilitiesService service = new
UtilitiesService("http://localhost:12345/Webservices100/BIJOU/");
bool tiersInUse = service.IsTiersInUse("BRELLO");
```

i) StatistiqueService

Fonctions disponibles :

- `List<Statistique> GetList();`
- `Statistique GetByName();`
- `Statistique Insert(Statistique statistique, Guid? guidCreateur = null);`
- `List<Statistique> Statistique();`
- `Statistique StatistiqueIntitule(string intitule);`

Fonction :
- `List<Statistique> GetList();`

Description :
Retourne la liste des statistiques.

Exemple :
```
IStatistiqueService service = new.
StatistiqueService("http://localhost:12345/Webservices100/BIJOU/");
List<Statistique> statistiques = service.GetList();
```

Fonction :
- `Statistique GetByName();`

Description :
Retourne la statistique par son nom.
Lève une ArgumentException si le paramètre name est nul.

Exemple :
```
IStatistiqueService service = new.
StatistiqueService("http://localhost:12345/Webservices100/BIJOU/");
Statistique statistique = service.GetByName("Ouest");
```

Fonction :
- `Statistique Statistiqueintitule();`

Description :
Retourne la statistique par son intitulé.
Lève une ArgumentException si le paramètre name est nul.

Exemple :
```
IStatistiqueService service = new.
StatistiqueService("http://localhost:12345/Webservices100/BIJOU/");
Statistique statistique = service.Statistiqueintitule("Ouest");
```

Fonction :

- `Statistique Insert(Statistique statistique, Guid? guidCreateur = null);`

Description :

Ajoute une nouvelle statistique en respectant les conditions suivantes :
Les champs ID et Intitule sont obligatoires.

Retourne la statistique inserée.
Lève une ArgumentException si la statistique est nulle.

Exemple :

```
Statistique statistique = new Statistique();
statistique.Id = 1;
statistique.Intitule = "France";

IStatistiqueService service = new.
StatistiqueService("http://localhost:12345/Webservices100/BIJOU/");
Statistique statistiqueInserted = service.Insert(statistique, utilisateurConnecte.Guid);
```

3. Les services paramètres

a) *CategorieComptableService*

Fonctions disponibles :

- `CategorieComptable GetByIndiceAchat(short indice);`
- `CategorieComptable GetByNameAchat(string name);`
- `List<CategorieComptable> GetListAchat();`
- `CategorieComptable GetByIndiceVente(short indice);`
- `CategorieComptable GetByNameVente(string name);`
- `List<CategorieComptable> GetListVente();`
- `CategorieComptable GetByIndiceStock(short indice);`
- `CategorieComptable GetByNameStock(string name);`
- `List<CategorieComptable> GetListStock();`

Fonction :
```
CategorieComptable GetByIndiceAchat(short indice);
```

Description :
Retourne une catégorie comptable d'achat par son indice.

Exemple :
```
ICategorieComptableService service = new.
CategorieComptableService("http://localhost:12345/Webservices100/BIJOU/");
CategorieComptable categorie = service.GetByIndiceAchat(1);
```

Fonction :
```
CategorieComptable GetByNameAchat(string name);
```

Description :
Retourne une catégorie comptable d'achat par son nom.
Lève une ArgumentException si le paramètre name est nul.

Exemple :
```
ICategorieComptableService service = new.
CategorieComptableService("http://localhost:12345/Webservices100/BIJOU/");
CategorieComptable categorie = service.GetByNameAchat("Achats import");
```

Fonction :
```
List<CategorieComptable> GetListAchat();
```

Description :
Retourne la liste des catégories comptable d'achat.

Exemple :
```
ICategorieComptableService service = new.
CategorieComptableService("http://localhost:12345/Webservices100/BIJOU/");
List<CategorieComptable> list = service.GetListAchat();
```

Fonction :
```
CategorieComptable GetByIndiceVente(short indice);
```

Description :
Retourne une catégorie comptable de vente par son indice.

Exemple :
```
ICategorieComptableService service = new.
CategorieComptableService("http://localhost:12345/Webservices100/BIJOU/");
CategorieComptable categorie = service.GetByIndiceVente(1);
```

Fonction :
```
CategorieComptable GetByNameVente(string name);
```

Description :
Retourne une catégorie comptable de vente par son nom.
Lève une ArgumentException si le paramètre name est nul.

Exemple :
```
ICategorieComptableService service = new.
CategorieComptableService("http://localhost:12345/Webservices100/BIJOU/");
CategorieComptable categorie = service.GetByNameVente("Ventes export");
```

Fonction :
```
List<CategorieComptable> GetListVente();
```

Description :
Retourne la liste des catégories comptable de vente.

Exemple :
```
ICategorieComptableService service = new.
CategorieComptableService("http://localhost:12345/Webservices100/BIJOU/");
List<CategorieComptable> list = service.GetListVente();
```

Fonction :
```
CategorieComptable GetByIndiceStock(short indice);
```

Description :
Retourne une catégorie comptable de stock par son indice.

Exemple :
```
ICategorieComptableService service = new.
CategorieComptableService("http://localhost:12345/Webservices100/BIJOU/");
CategorieComptable categorie = service.GetByIndiceStock(1);
```

Fonction :
```
CategorieComptable GetByNameStock(string name);
```

Description :
Retourne une catégorie comptable de stock par son nom.
Lève une ArgumentException si le paramètre name est nul.

Exemple :
```
ICategorieComptableService service = new.
CategorieComptableService("http://localhost:12345/Webservices100/BIJOU/");
CategorieComptable categorie = service.GetByNameStock("Stock France");
```

Fonction :
```
List<CategorieComptable> GetListStock();
```

Description :
Retourne la liste des catégories comptable de stock.

Exemple :
```
ICategorieComptableService service = new.
CategorieComptableService("http://localhost:12345/Webservices100/BIJOU/");
List<CategorieComptable> list = service.GetListStock();
```

b) CategorieTarifaireService

Fonctions disponibles :

- `CategorieTarifaire GetByIndice(short indice);`
- `CategorieTarifaire GetByName(string name);`
- `List<CategorieTarifaire> GetList();`

Fonction :
```
CategorieTarifaire GetByIndice(short indice);
```

Description :
Retourne une catégorie tarifaire par son indice.

Exemple :
```
ICategorieTarifaireService service = new.
CategorieTarifaireService("http://localhost:12345/Webservices100/BIJOU/");
CategorieTarifaire categorie = service.GetByIndice(1);
```

Fonction :
```
CategorieTarifaire GetByName(string name);
```

Description :
Retourne une catégorie tarifaire par son nom.
Lève une ArgumentException si le paramètre name est nul.

Exemple :
```
ICategorieTarifaireService service = new.
CategorieTarifaireService("http://localhost:12345/Webservices100/BIJOU/");
CategorieTarifaire categorie = service.GetByName("Clients comptoir");
```

Fonction :
```
List<CategorieTarifaire> GetList();
```

Description :
Retourne la liste des catégories tarifaires.

Exemple :
```
ICategorieTarifaireService service = new.
CategorieTarifaireService("http://localhost:12345/Webservices100/BIJOU/");
List<CategorieTarifaire> list = service.GetList();
```

c) ConditionLivraisonService

Fonctions disponibles :

- `ConditionLivraison GetByIndice(short indice);`
- `ConditionLivraison GetByName(string name);`
- `List<ConditionLivraison> GetList();`

Fonction :
`ConditionLivraison GetByIndice(short indice);`

Description :
Retourne une condition de livraison par son indice.

Exemple :
```
IConditionLivraisonService service = new
ConditionLivraisonService("http://localhost:12345/Webservices100/BIJOU/");
ConditionLivraison condition = service.GetByIndice(1);
```

Fonction :
`ConditionLivraison GetByName(string name);`

Description :
Retourne une condition de livraison par son nom.
Lève une ArgumentException si le paramètre name est nul.

Exemple :
```
IConditionLivraisonService service = new
ConditionLivraisonService("http://localhost:12345/Webservices100/BIJOU/");
ConditionLivraison condition = service.GetByName("Exp. Franco transporteur");
```

Fonction :
`List<ConditionLivraison> GetList();`

Description :
Retourne la liste des conditions de livraison.

Exemple :
```
IConditionLivraisonService service = new
ConditionLivraisonService("http://localhost:12345/Webservices100/BIJOU/");
List<ConditionLivraison> list = service.GetList();
```

d) DeviseService

Fonctions disponibles :

- `Devise GetByIndice(short indice);`
- `List<Devise> GetList();`

Fonction :
`Devise GetByIndice(short indice);`

Description :
Retourne une devise par son indice.

Exemple :
```
IDeviseService service = new
DeviseService("http://localhost:12345/Webservices100/BIJOU/");
Devise devise = service.GetByIndice(1);
```

Fonction :
`List<Devise> GetList();`

Description :
Retourne la liste des devises.

Exemple :
```
IDeviseService service = new
DeviseService("http://localhost:12345/Webservices100/BIJOU/");
List<Devise> list = service.GetList();
```

e) DossierService

Fonctions disponibles :

- `Periode GetExerciceByDate(DateTime date);`
- `List<Periode> GetExercices();`
- `Devise GetDeviseDossier();`

Fonction :
`Periode GetExerciceByDate(DateTime date);`

Description :
Retourne un exercice comptable en fonction de la date passée.
La classe `Periode` contient la date de début et la date de fin de l'exercice et indique également si l'exercice est clôturé.

Exemple :
```
IDossierService service = new
DossierService("http://localhost:12345/Webservices100/BIJOU/");
Periode exercice = service.GetExerciceByDate(DateTime.Now);
```

Fonction :
`List<Periode> GetExercices();`

Description :
Retourne la liste des exercices comptables.

Exemple :
```
IDossierService service = new
DossierService("http://localhost:12345/Webservices100/BIJOU/");
List<Periode> list = service.GetExercices();
```

Fonction :
`Devise GetDeviseDossier();`

Description :
Retourne la devise du dossier.

Exemple :
```
IDossierService service = new
DossierService("http://localhost:12345/Webservices100/BIJOU/");
Devise devise = service.GetDeviseDossier();
```

f) GammeService

Fonctions disponibles :

- `List<Gamme> GetList();`
- `List<GammeArticle> GetListGamme1ByRefArticle(string refArticle);`
- `List<GammeArticle> GetListGamme2ByRefArticle(string refArticle);`
- `List<CombinaisonGammeArticle> GetCombinaisonGammeList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);`
- `List<CombinaisonGammeArticle> GetCombinaisonGammeListByRefArticle(string refArticle);`

Fonction :

```
List<Gamme> GetList();
```

Description :

Retourne la liste des gammes.

Exemple :

```
IGammeService service = new
GammeService("http://localhost:12345/Webservices100/BIJOU/");
List<Gamme> list = service.GetList();
```

Fonction :

```
List<GammeArticle> GetListGamme1ByRefArticle(string refArticle);
```

Description :

Retourne la liste des gammes d'article 1 de la référence d'article.
Lève une ArgumentException si le paramètre refArticle est nul.

Exemple :

```
IGammeService service = new
GammeService("http://localhost:12345/Webservices100/BIJOU/");
List<GammeArticle> list = service.GetListGamme1ByRefArticle("BAOR01");
```

Fonction :

```
List<GammeArticle> GetListGamme2ByRefArticle(string refArticle);
```

Description :

Retourne la liste des gammes d'article 2 de la référence d'article.
Lève une ArgumentException si le paramètre refArticle est nul

Exemple :

```
IGammeService service = new
GammeService("http://localhost:12345/Webservices100/BIJOU/");
List<GammeArticle> list = service.GetListGamme2ByRefArticle("CHAAR/VAR");
```

Fonction :
```
List<CombinaisonGammeArticle> GetCombinaisonGammeList(Criteria criteria = null,
List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
```

Description :
Retourne la liste de combinaisons des gammes et d'articles 1

Exemple :
```
IGammeService service = new
GammeService("http://localhost:12345/Webservices100/BIJOU/");
List< CombinaisonGammeArticle > list = service. GetCombinaisonGammeList ()
```

Fonction :
```
List<CombinaisonGammeArticle> GetCombinaisonGammeListByRefArticle (string refArticle);
```

Description :
Retourne la liste de combinaisons des gammes et d'articles dont la référence est passé en paramètre.
Lève une ArgumentException si le paramètre refArticle est nul

Exemple :
```
IGammeService service = new
GammeService("http://localhost:12345/Webservices100/BIJOU/");
List< CombinaisonGammeArticle > list =
service.GetCombinaisonGammeListByRefArticle("BAOR01");
```

g) InfoLibreService

Fonctions disponibles :

- `ListInfoLibre GetInfosLibresTiers();`
- `ListInfoLibre GetInfosLibresCompteGeneral();`
- `ListInfoLibre GetInfosLibresCompteAnalytique();`
- `ListInfoLibre GetInfosLibresEcriture();`
- `ListInfoLibre GetInfosLibresArticle();`
- `ListInfoLibre GetInfosLibresDocument();`
- `ListInfoLibre GetInfosLibresLigneDocument();`
- `List<string> GetListIntituleInfoLibreTypeTableArticle(string nomInfoLibre);`
- `List<string> GetListIntituleInfoLibreTypeTableLigneDocument(string nomInfoLibre);`
- `List<string> GetListIntituleInfoLibreTypeTableDocument(string nomInfoLibre);`
- `List<string> GetListIntituleInfoLibreTypeTableTiers(string nomInfoLibre);`
- `List<string> GetListIntituleInfoLibreTypeTableCompteGeneral(string nomInfoLibre);`
- `List<string> GetListIntituleInfoLibreTypeTableCompteAnalytique(string nomInfoLibre);`
- `List<string> GetListIntituleInfoLibreTypeTableEcriture(string nomInfoLibre);`
- `List<string> GetListIntituleInfoLibreTypeTableRessources(string nomInfoLibre);`

Fonction :
`ListInfoLibre GetInfosLibresTiers();`

Description :
Renvoie la liste des informations libres des tiers.

Exemple :
```
IInfoLibreService service = new
InfoLibreService("http://localhost:12345/Webservices100/BIJOU/");
IList<InfoLibre> list = service.GetInfosLibresTiers();
```

Fonction :
`ListInfoLibre GetInfosLibresCompteGeneral();`

Description :
Renvoie la liste des informations libres des comptes généraux.

Exemple :
```
IInfoLibreService service = new
InfoLibreService("http://localhost:12345/Webservices100/BIJOU/");
IList<InfoLibre> list = service.GetInfosLibresCompteGeneral();
```

Fonction :
```
ListInfoLibre GetInfosLibresCompteAnalytique();
```

Description :
Renvoie la liste des informations libres des comptes analytiques.

Exemple :
```
IInfoLibreService service = new
InfoLibreService("http://localhost:12345/Webservices100/BIJOU/");
IList<InfoLibre> list = service.GetInfosLibresCompteAnalytique();
```

Fonction :
```
ListInfoLibre GetInfosLibresEcriture();
```

Description :
Renvoie la liste des informations libres des écritures comptables.

Exemple :
```
IInfoLibreService service = new
InfoLibreService("http://localhost:12345/Webservices100/BIJOU/");
IList<InfoLibre> list = service.GetInfosLibresEcriture();
```

Fonction :
```
ListInfoLibre GetInfosLibresArticle();
```

Description :
Renvoie la liste des informations libres des articles.

Exemple :
```
IInfoLibreService service = new
InfoLibreService("http://localhost:12345/Webservices100/BIJOU/");
IList<InfoLibre> list = service.GetInfosLibresArticle();
```

Fonction :
```
ListInfoLibre GetInfosLibresDocument();
```

Description :
Renvoie la liste des informations libres des documents.

Exemple :
```
IInfoLibreService service = new
InfoLibreService("http://localhost:12345/Webservices100/BIJOU/");
IList<InfoLibre> list = service.GetInfosLibresDocument();
```

Fonction :
```
ListInfoLibre GetInfosLibresLigneDocument();
```

Description :
Renvoie la liste des informations libres des lignes de document.

Exemple :
```
IInfoLibreService service = new
InfoLibreService("http://localhost:12345/Webservices100/BIJOU/");
IList<InfoLibre> list = service.GetInfosLibresLigneDocument();
```

Fonction :
```
List<string> GetListIntituleInfoLibreTypeTableArticle (string nomInfoLibre);
```

Description :
Renvoie la liste des intitulés des informations libres de la table des articles et dont le nom est passé en paramètre.
Lève une ArgumentException si le paramètre nomInfoLibre est nul.

Exemple :
```
IInfoLibreService service = new
InfoLibreService("http://localhost:12345/Webservices100/BIJOU/");
List< string > list = service. GetListIntituleInfoLibreTypeTableArticle ("Status");
```

Fonction :
```
List<string> GetListIntituleInfoLibreTypeTableLigneDocument(string nomInfoLibre);
```

Description :
Renvoie la liste des intitulés des informations libres de la table des lignes de documents et dont le nom est passé en paramètre.
Lève une ArgumentException si le paramètre nomInfoLibre est nul.

Exemple :
```
IInfoLibreService service = new
InfoLibreService("http://localhost:12345/Webservices100/BIJOU/");
List< string > list = service. GetListIntituleInfoLibreTypeTableLigneDocument
("Status");
```

Fonction :
```
List<string> GetListIntituleInfoLibreTypeTableDocument (string nomInfoLibre);
```

Description :
Renvoie la liste des intitulés des informations libres de la table des documents et dont le nom est passé en paramètre
Lève une ArgumentException si le paramètre nomInfoLibre est nul.

Exemple :
```
IInfoLibreService service = new
InfoLibreService("http://localhost:12345/Webservices100/BIJOU/");
List< string > list = service. GetListIntituleInfoLibreTypeTableDocument ("Status");
```

Fonction :
```
List<string> GetListIntituleInfoLibreTypeTableTiers (string nomInfoLibre);
```

Description :
Renvoie la liste des intitulés des informations libres de la table des tiers et dont le nom est passé en paramètre.
Lève une ArgumentException si le paramètre nomInfoLibre est nul.

Exemple :
```
IInfoLibreService service = new
InfoLibreService("http://localhost:12345/Webservices100/BIJOU/");
List< string > list = service. GetListIntituleInfoLibreTypeTableTiers ("Status");
```

Fonction :
```
List<string> GetListIntituleInfoLibreTypeTableCompteGeneral (string nomInfoLibre);
```

Description :
Renvoie la liste des intitulés des informations libres de la table des comptes généraux et dont le nom est passé en paramètre.
Lève une ArgumentException si le paramètre nomInfoLibre est nul.

Exemple :
```
IInfoLibreService service = new
InfoLibreService("http://localhost:12345/Webservices100/BIJOU/");
List< string > list = service. GetListIntituleInfoLibreTypeTableCompteGeneral
("Status");
```

Fonction :
```
List<string> GetListIntituleInfoLibreTypeTableCompteAnalytique (string nomInfoLibre);
```

Description :
Renvoie la liste des intitulés des informations libres de la table des comptes analytiques et dont le nom est passé en paramètre.
Lève une ArgumentException si le paramètre nomInfoLibre est nul.

Exemple :
```
IInfoLibreService service = new
InfoLibreService("http://localhost:12345/Webservices100/BIJOU/");
List< string > list = service. GetListIntituleInfoLibreTypeTableCompteAnalytique
("Status");
```

Fonction :
```
List<string> GetListIntituleInfoLibreTypeTableEcriture (string nomInfoLibre);
```

Description :
Renvoie la liste des intitulés des informations libres de la table des écritures et dont le nom est passé en paramètre.
Lève une ArgumentException si le paramètre nomInfoLibre est nul.

Exemple :
```
IInfoLibreService service = new
InfoLibreService("http://localhost:12345/Webservices100/BIJOU/");
List< string > list = service. GetListIntituleInfoLibreTypeTableEcriture ("Status");
```

Fonction :
```
List<string> GetListIntituleInfoLibreTypeTableRessources (string nomInfoLibre);
```

Description :
Renvoie la liste des intitulés des informations libres de la table des ressources et dont le nom est passé en paramètre.
Lève une ArgumentException si le paramètre nomInfoLibre est nul.

Exemple :
```
IInfoLibreService service = new
InfoLibreService("http://localhost:12345/Webservices100/BIJOU/");
List< string > list = service. GetListIntituleInfoLibreTypeTableRessources ("status") ;
```

h) ModeExpeditionService

Fonctions disponibles :

- `ModeExpedition GetByIndice(short indice);`
- `ModeExpedition GetByName(string name);`
- `List<ModeExpedition> GetList();`

Fonction :
`ModeExpedition GetByIndice(short indice);`

Description :
Retourne un mode d'expédition par son indice.

Exemple :
```
IModeExpeditionService service = new
ModeExpeditionService("http://localhost:12345/Webservices100/BIJOU/");
ModeExpedition expedition = service.GetByIndice(1);
```

Fonction :
`ModeExpedition GetByName(string name);`

Description :
Retourne un mode d'expédition par son nom.
Lève une ArgumentException si le paramètre name est nul.

Exemple :
```
IModeExpeditionService service = new
ModeExpeditionService("http://localhost:12345/Webservices100/BIJOU/");
ModeExpedition expedition = service.GetByName("Chronoposte");
```

Fonction :
`List<ModeExpedition> GetList();`

Description :
Retourne la liste des modes d'expédition.

Exemple :
```
IModeExpeditionService service = new
ModeExpeditionService("http://localhost:12345/Webservices100/BIJOU/");
List<ModeExpedition> list = service.GetList();
```

i) ModeleReglementService

Fonctions disponibles :

- `ModeleReglement GetByIndice(int indice);`
- `ModeleReglement GetByName(string name);`
- `List<ModeleReglement> GetList();`

Fonction :
`ModeleReglement GetByIndice(int indice);`

Description :
Retourne un modèle de règlement par son indice.

Exemple :
```
IModeleReglementService service = new
ModeleReglementService("http://localhost:12345/Webservices100/BIJOU/");
ModeleReglement modele = service.GetByIndice(1);
```

Fonction :
`ModeleReglement GetByName(string name);`

Description :
Retourne un modèle de règlement par son nom.
Lève une ArgumentException si le paramètre name est nul.

Exemple :
```
IModeleReglementService service = new
ModeleReglementService("http://localhost:12345/Webservices100/BIJOU/");
ModeleReglement modele = service.GetByName("Paiement en 3 fois");
```

Fonction :
`List<ModeleReglement> GetList();`

Description :
Retourne la liste des modèles de règlement.

Exemple :
```
IModeleReglementService service = new
ModeleReglementService("http://localhost:12345/Webservices100/BIJOU/");
List<ModeleReglement> list = service.GetList();
```

j) NiveauAnalyseService

Fonctions disponibles :

- `List<NiveauAnalyse> GetList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);`
- `List < NiveauAnalyse > NiveauxAnalyses(int pageNumber = 0, int rowsPerPage = 0);`
- `NiveauAnalyse NiveauAnalyseById(short id);`
- `NiveauAnalyse NiveauAnalyseId(string id);`

Fonction :

- `List<NiveauAnalyse> GetList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0)`

Description :

Retourne la liste des niveaux d'analyse répondant aux critères.

Exemple :

```
INiveauAnalyseService service = new
NiveauAnalyseService("http://localhost:12345/Webservices100/BIJOU/");
List<NiveauAnalyse> list = service.GetList();
```

Fonction :

- `List<NiveauAnalyse> NiveauxAnalyses (int pageNumber = 0, int rowsPerPage = 0)`

Description :

Retourne la liste des niveaux d'analyse suivant la pagination fournie.

Exemple :

```
INiveauAnalyseService service = new
NiveauAnalyseService("http://localhost:12345/Webservices100/BIJOU/");
List<NiveauAnalyse> list = service.NiveauxAnalyses(2, 4);
```

Fonction :

- `NiveauAnalyse> NiveauAnalyseById (short id)`

Description :

Retourne le niveau d'analyse par son id.

Exemple :

```
INiveauAnalyseService service = new
NiveauAnalyseService("http://localhost:12345/Webservices100/BIJOU/");
NiveauAnalyse niveau= service.NiveauAnalyseById(2);
```

Fonction :
- `NiveauAnalyse NiveauAnalyseId (string id)`

Description :
Retourne le niveau d'analyse par son id.

Exemple :
```
INiveauAnalyseService service = new
NiveauAnalyseService("http://localhost:12345/Webservices100/BIJOU/");
NiveauAnalyse niveau = service.NiveauAnalyseId("5");
```

k) PaysService

Fonctions disponibles :
▪ `List<Pays> GetList();`

Fonction : `List<Pays> GetList()`
Description : Retourne la liste des pays.

Exemple :
`IPaysService service = new PaysService("http://localhost:12345/Webservices100/BIJOU/");`
`List<Pays> list = service.GetList();`

I) PlanAnalytiqueService

Fonctions disponibles :

- `PlanAnalytique GetByIndice(short indice);`
- `PlanAnalytique GetByName(string name);`
- `List<PlanAnalytique> GetList();`

Fonction :
```
PlanAnalytique GetByIndice(short indice);
```

Description :
Retourne un plan analytique par son indice.

Exemple :
```
IPlanAnalytiqueService service = new
PlanAnalytiqueService("http://localhost:12345/Webservices100/BIJOU/");
PlanAnalytique plan = service.GetByIndice(1);
```

Fonction :
```
PlanAnalytique GetByName(string name);
```

Description :
Retourne un plan analytique par son nom.
Lève une ArgumentException si le paramètre name est nul.

Exemple :
```
IPlanAnalytiqueService service = new
PlanAnalytiqueService("http://localhost:12345/Webservices100/BIJOU/");
PlanAnalytique plan = service.GetByName("Activité");
```

Fonction :
```
List<PlanAnalytique> GetList();
```

Description :
Retourne la liste des plans analytiques.

Exemple :
```
IPlanAnalytiqueService service = new
PlanAnalytiqueService("http://localhost:12345/Webservices100/BIJOU/");
List<PlanAnalytique> list = service.GetList();
```

m) ServiceContactService

Fonctions disponibles :

- `ServiceContact GetByIndice(short indice);`
- `ServiceContact GetByName(string name);`
- `List<ServiceContact> GetList();`

Fonction :
`ServiceContact GetByIndice(short indice);`

Description :
Retourne un service de contact par son indice.

Exemple :
```
IServiceContactService service = new
ServiceContactService("http://localhost:12345/Webservices100/BIJOU/");
ServiceContact serviceContact = service.GetByIndice(1);
```

Fonction :
`ServiceContact GetByName(string name);`

Description :
Retourne un service de contact par son nom.
Lève une ArgumentException si le paramètre name est nul.

Exemple :
```
IServiceContactService service = new
ServiceContactService("http://localhost:12345/Webservices100/BIJOU/");
ServiceContact serviceContact = service.GetByName("Commerciale");
```

Fonction :
`List<ServiceContact> GetList();`

Description :
Retourne la liste des services de contact.

Exemple :
```
IServiceContactService service = new
ServiceContactService("http://localhost:12345/Webservices100/BIJOU/");
List<ServiceContact> list = service.GetList();
```

n) StructureCompteService

Fonctions disponibles :

- `List<StructureCompte> GetAll();`
- `StructureCompte Get(StructureCompteBancaire structure);`

Fonction :
`List<StructureCompte> GetAll();`

Description :
Retourne la liste des structures de comptes bancaires.

Exemple :
```
IStructureCompteService service = new
StructureCompteService("http://localhost:12345/Webservices100/BIJOU/");
List<StructureCompte> structureCompteList = service.GetAll();
```

Fonction :
- `StructureCompte Get(StructureCompteBancaire structure);`

Description :
Retourne une structure de compte bancaire par son type.

Exemple :
```
IStructureCompteService service = new
StructureCompteService("http://localhost:12345/Webservices100/BIJOU/");
StructureCompte structureCompte = service.Get(StructureCompteBancaire.Autre);
```

o) TypeContactService

Fonctions disponibles :

- `TypeContact GetByIndice(short indice);`
- `TypeContact GetByName(string name);`
- `List<TypeContact> GetList();`

Fonction :
`TypeContact GetByIndice(short indice);`

Description :
Retourne un type de contact par son indice.

Exemple :
```
ITypeContactService service = new
TypeContactService("http://localhost:12345/Webservices100/BIJOU/");
TypeContact type = service.GetByIndice(1);
```

Fonction :
`TypeContact GetByName(string name);`

Description :
Retourne un type de contact par son nom.
Lève une ArgumentException si le paramètre name est nul.

Exemple :
```
ITypeContactService service = new
TypeContactService("http://localhost:12345/Webservices100/BIJOU/");
TypeContact type = service.GetByName("Huissier");
```

Fonction :
`List<TypeContact> GetList();`

Description :
Retourne la liste des types de contact.

Exemple :
```
ITypeContactService service = new
TypeContactService("http://localhost:12345/Webservices100/BIJOU/");
List<TypeContact> list = service.GetList();
```

4. Les services de données de gestion commerciale

a) AbonnementService

Fonctions disponibles :

- `IList<Abonnement> GetAbonnementList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);`
- `Abonnement GetAbonnementById(int idAbonnement);`
- `IList<Abonnement> GetModeleAbonnementListByTypeTiers(AbonnementTypeTiers abonnementTypeTiers);`
- `int Insert(Abonnement abonnement, AbonnementEntete entete, List<AbonnementLigne> ligneList, Guid? guidCreateur = null);`
- `AbonnementEntete GetEnteteByIdAbonnement(int idAbonnement);`
- `IList<AbonnementLigne> GetListAbonnementLigneByIdAbonnement(int idAbonnement);`
- `IList<MotifResiliation> GetMotifResiliationList();`
- `IList<AbonnementPeriode> GetHistoriqueByIdAbonnement(int idAbonnement);`
- `void DefinirConditionReglement(int idAbonnement, List<ConditionReglement> listConditionReglement);`
- `void DefinirHistorique(int idAbonnement, List <AbonnementPeriode> periodes);`
- `void DeleteAbonnementsPeriodes(List<int> idPeriodeAbonnementList);`
- `void UpdateLignesAbonnement(List<AbonnementLigne> lignes, Guid? guidUtilisateur = null);`
- `AbonnementEntete UpdateEnteteAbonnement(AbonnementEntete entete, Guid? guidUtilisateur = null);`

Fonction :

```
IList<Abonnement> GetAbonnementList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
```

Description :

Retourne une collection d'abonnement répondant aux critères passés en paramètre

Exemple :

```
IAbonnementService service = new AbonnementService("http://localhost:12345/Webservices100/BIJOU/");
IList<Abonnement> abonnementList = service.GetAbonnementList(new CriteriaComparison(nameof(Abonnement.AbonnementType), ComparisonOperator.Equals, 1));
```

Fonction :

```
Abonnement GetAbonnementById(int idAbonnement);
```

Description :

Retourne l'abonnement ayant l'identifiant idAbonnement passé en paramètre

Exemple :

```
IAbonnementService service = new AbonnementService("http://localhost:12345/Webservices100/BIJOU/");
Abonnement abonnement = service.GetAbonnementById(12);
```

Fonction :
```
IList<Abonnement> GetModeleAbonnementListByTypeTiers(AbonnementTypeTiers
abonnementTypeTiers);
```

Description :
Retourne une collection d'abonnement modèle ayant le type de tiers passé en paramètre

Exemple :
```
IAbonnementService service = new
AbonnementService("http://localhost:12345/Webservices100/BIJOU/");
IList<Abonnement> abonnementModeleFournisseurList =
service.GetModeleAbonnementListByTypeTiers(AbonnementTypeTiers.Fournisseur);
```

Fonction :
```
int Insert(Abonnement abonnement, AbonnementEntete entete, List<AbonnementLigne>
ligneList, Guid? guidCreateur = null);
```

Description :
Insère un abonnement complet (abonnement, entête d'abonnement, lignes d'abonnement).
Retourne l'identifiant de l'abonnement inséré.
Lève différentes exceptions si les données ne sont pas cohérentes (guidCreateur invalide, tiers inexistant, dates invalides, type de pièce générée incorrect, modèle inexistant, souche inexistante,…)

Exemple :
```
UtilisateurSage utilisateurSage = ConnectUser(…);
Abonnement abonnement = CreateNewAbonnement(…);
AbonnementEntete enteteAbonnement = CreateNewAbonnementEntete(…);
List<AbonnementLigne> ligneAbonnementCollection = CreateListeLigneAbonnement(…);
IAbonnementService service = new
AbonnementService("http://localhost:12345/Webservices100/BIJOU/");
int idAbonnement = service.Insert(abonnement, enteteAbonnement,
ligneAbonnementCollection, utilisateurSage.Guid);
```

Fonction :
```
AbonnementEntete GetEnteteByIdAbonnement(int idAbonnement)
```

Description :
Retourne l'instance d'AbonnementEntete associée à l'abonnement ayant l'identifiant idAbonnement passé en paramètre.

Exemple :
```
IAbonnementService service = new
AbonnementService("http://localhost:12345/Webservices100/BIJOU/");
AbonnementEntete enteteAbonnement = service.GetEnteteByIdAbonnement(12);
```

Fonction :
```
IList<AbonnementLigne> GetListAbonnementLigneByIdAbonnement(int idAbonnement)
```

Description :
Retourne la collection des lignes d'abonnement associées à l'abonnement ayant l'identifiant idAbonnement passé en paramètre.

Exemple :
```
IAbonnementService service = new
AbonnementService("http://localhost:12345/Webservices100/BIJOU/");
IList<AbonnementLigne> ligneList = service. GetListAbonnementLigneByIdAbonnement(12);
```

Fonction :
```
IList<MotifResiliation> GetMotifResiliationList()
```

Description :
Retourne la liste des motifs de résiliation.

Exemple :
```
IAbonnementService service = new
AbonnementService("http://localhost:12345/Webservices100/BIJOU/");
IList< MotifResiliation > motifList = service. GetMotifResiliationList();
```

Fonction :
```
IList<AbonnementPeriode> GetHistoriqueByIdAbonnement(int idAbonnement)
```

Description :
Retourne l'historique (collection `AbonnementPeriode`) associé à l'abonnement ayant l'identifiant idAbonnement passé en paramètre.

Exemple :
```
IAbonnementService service = new
AbonnementService("http://localhost:12345/Webservices100/BIJOU/");
IList<AbonnementPeriode> historiqueList = service. GetHistoriqueByIdAbonnement(5);
```

Fonction :
```
void DefinirConditionReglement(int idAbonnement, List<ConditionReglement>
listConditionReglement)
```

Description :
Définit une liste de `ConditionReglement` à associer à l'abonnement ayant l'identifiant idAbonnement passé en paramètre.

Exemple :
```
IAbonnementService service = new
AbonnementService("http://localhost:12345/Webservices100/BIJOU/");
IList<ConditionReglement> conditions = new List<ConditionReglement>()
{
   new ConditionReglement() {IdReglement = 1, TypeConditionReglement =
TypeConditionReglement.JourNet, NbJour = 0, TypeRepartition =
TypeRepartitionReglement.Pourcentage, ValeurRepartition = 30},
   new ConditionReglement () {IdReglement = 4, TypeConditionReglement =
TypeConditionReglement.JourNet, NbJour = 15, JourTombee1 = 11, TypeRepartition =
TypeRepartitionReglement.Pourcentage, ValeurRepartition = 30},
   new ConditionReglement () {IdReglement = 1, TypeConditionReglement =
TypeConditionReglement.FinMois, NbJour = 45, TypeRepartition =
TypeRepartitionReglement.Equilibre, ValeurRepartition = 0},
                };
service.DefinirConditionReglement(abonnement.Id, conditions);
```

Fonction :
```
void DefinirHistorique(int idAbonnement, List <AbonnementPeriode> periodes)
```

Description :
Définit la périodicité de l'abonnement idAbonnement passé en paramètre.

Exemple :
```
IAbonnementService service = new
AbonnementService("http://localhost:12345/Webservices100/BIJOU/");
List<AbonnementPeriode> abonnementPeriodes = new List<AbonnementPeriode>()
   {
       new AbonnementPeriode() { DateDebut =
DateTime.Parse($"01/01/{DateTime.Now.Year}"), DateFin =
DateTime.Parse($"01/02/{DateTime.Now.Year}"),
       AbonnementEtatPeriodicite = AbonnementEtatPeriodicite.NonGeneree, DateLivraison =
DateTime.Parse($"01/01/{DateTime.Now.Year}") }
                };
IList<AbonnementPeriode> historiqueList = service.GetHistoriqueByIdAbonnement(5);
```

Fonction :

```
void UpdateLignesAbonnement(List<AbonnementLigne> lignes, Guid? guidUtilisateur = null)
```

Description :

Met à jour un abonnement existant en respectant les règles suivantes :

- l'ID de la ligne doit exister.
- Le type d'abonnement ne peut pas être modifié.
- L'abonnement associé à la ligne ne doit pas être en cours d'utilisation dans SAGE.
- Si la traçabilité des données est activée, les modifications de données sont historisées.

Lève une ArgumentException si le paramètre lignes est nul ou vide.

Exemple :

```
IAbonnementService service = new
AbonnementService("http://localhost:12345/Webservices100/BIJOU/");
IList<AbonnementLigne> lignes = service.GetListAbonnementLigneByIdAbonnement(5);
//Modification des lignes d'abonnement
service.UpdateLignesAbonnement(lignes, guidUtilisateur);
```

Fonction :

```
AbonnementEntete UpdateEnteteAbonnement(AbonnementEntete entete, Guid? guidUtilisateur =
null)
```

Description :

Met à jour un entête d'abonnement existant en respectant les règles suivantes :

- l'ID de l'abonnement doit exister.
- Le type d'abonnement ne peut pas être modifié.
- L'abonnement ne doit pas être en cours d'utilisation dans SAGE.
- Si la traçabilité des données est activée, les modifications de données sont historisées.

Lève une ArgumentException si le paramètre lignes est nul ou vide.
Retourne l'entête d'abonnement.

Exemple :

```
IAbonnementService service = new
AbonnementService("http://localhost:12345/Webservices100/BIJOU/");
Abonnement abonnement = service.GetAbonnementById(5);
//Modification de l'entête d'abonnement.
service.UpdateEnteteAbonnement(AbonnementEntete entete, Guid? guidUtilisateur = null) ;
```

b) AdresseLivraisonService

Fonctions disponibles :

```
AdresseLivraison GetAdresseLivraison(string numeroTiers, int idAdresse);
List<AdresseLivraison> GetList(Criteria criteria = null, List<Order> orders = null,
int pageNumber = 0, int rowsPerPage = 0);
List<AdresseLivraison> GetListByTiers(string numeroTiers, List<Order> orders = null);
int GetCount(Criteria criteria = null);
AdresseLivraison Insert(AdresseLivraison adresse , Guid? guidCreateur = null);
AdresseLivraison Update(AdresseLivraison adresse);
bool Delete(string numeroTiers, int idAdresse);
```

Fonction :

```
AdresseLivraison GetAdresseLivraison(string numeroTiers, int idAdresse);
```

Description :

Retourne une adresse de livraison d'un tiers.
Lève une ArgumentException si le paramètre numeroTiers est nul

Exemple :

```
IAdresseLivraisonService service = new
AdresseLivraisonService("http://localhost:12345/Webservices100/BIJOU/");
AdresseLivraison adresse = service.GetAdresseLivraison("BAGUES", 1);
```

Fonction :

```
List<AdresseLivraison> GetList(Criteria criteria = null, List<Order> orders = null, int
pageNumber = 0, int rowsPerPage = 0);
```

Description :

Retourne la liste des adresses de livraison répondant aux critères passés en paramètre.

Exemple :

```
IAdresseLivraisonService service = new
AdresseLivraisonService("http://localhost:12345/Webservices100/BIJOU/");
List<AdresseLivraison> AdresseLivraison = service.GetList();
```

Fonction :

```
List<AdresseLivraison> GetListByTiers(string numeroTiers, List<Order> orders = null);
```

Description :

Retourne la liste des adresses de livraison d'un tiers.
Lève une ArgumentException si le paramètre numeroTiers est nul

Exemple :

```
IAdresseLivraisonService service = new
AdresseLivraisonService("http://localhost:12345/Webservices100/BIJOU/");
List<AdresseLivraison> list = service.GetListByTiers("BAGUES");
```

Fonction :
```
int GetCount(Criteria criteria = null);
```

Description :
Retourne le nombre d'adresses de livraison répondant aux critères passés en paramètre.

Exemple :
```
IAdresseLivraisonService service = new
AdresseLivraisonService("http://localhost:12345/Webservices100/BIJOU/");
Criteria criteria = new CriteriaComparison("Pays", ComparisonOperator.Like, "France");
int nb = service.GetCount(criteria);
```

Fonction :
```
AdresseLivraison Insert(AdresseLivraison adresse , Guid? guidCreateur = null);
```

Description :
Ajoute une nouvelle adresse de livraison.
Lève une ArgumentException si le paramètre adresse est nul

Exemple :
```
AdresseLivraison adresse = new AdresseLivraison();
adresse.Id = 0;
adresse.NumeroTiers = "BAGUES";
adresse.Intitule = "Adresse Principale";
adresse.Contact = "Martin Eric";
adresse.Adresse = "1, Rue de l'adresse";
adresse.Complement = "";
adresse.CodePostal = "L-1010";
adresse.Ville = "Lange";
adresse.Pays = "Luxembourg";
adresse.Region = "Luxembourg";
adresse.Fax = "+352 17 22 23 42";
adresse.Telephone = "+352 17 22 23 42";
adresse.Email = "info@Test.lu";

IAdresseLivraisonService service = new
AdresseLivraisonService("http://localhost:12345/Webservices100/BIJOU/");
AdresseLivraison adresseInserted = service.Insert(adresse,utilisateurConnecte.Guid);
```

Fonction :
```
AdresseLivraison Update(AdresseLivraison adresse);
```

Description :
Met à jour une adresse de livraison existante.
Lève une ArgumentException si le paramètre adresse est nul

Exemple :
```
IAdresseLivraisonService service = new
AdresseLivraisonService("http://localhost:12345/Webservices100/BIJOU/");
AdresseLivraison adresse = service.GetAdresseLivraison("BAGUES", 1);
//Modification de l'adresse
AdresseLivraison adresseUpdated = service.Update(adresse);
```

Fonction :
```
bool Delete(string idTiers, int idAdresse);
```

Description :
Supprime une adresse de livraison existante.
Retourne true si l'adresse a été suprimée

Exemple :
```
IAdresseLivraisonService service = new
AdresseLivraisonService("http://localhost:12345/Webservices100/BIJOU/");
AdresseLivraison adresse = service.GetAdresseLivraison("BAGUES", 1);
bool deleted = service.Delete(adresse.NumeroTiers, adresse.Id);
```

c) ArticleService

Fonctions disponibles :

```
Article GetArticle(string reference);
List<Article> GetList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
int GetCount(Criteria criteria = null);
Article Insert(Article article, bool useSageProcess = true, Guid? guidCreateur = null);
Article Insert(ParametreInsertArticle data);
Article Update(Article article, Guid? guidUtilisateur = null);
bool Delete(string reference);

//--Unité de vente
List<UniteVente> GetUniteVenteList();
UniteVente GetUniteVenteById(int id);
UniteVente GetUniteVenteByName(string name);

//Catalogue
List<Catalogue> GetCatalogueList();
Catalogue GetCatalogue(int id);
List<Catalogue> GetCataloguesByParent(int parentId);

//Statistique article
List<StatistiqueArticle> GetStatistiqueList();
List<StatistiqueArticle> GetStatistiqueArticleListByIdChamp(short idChamp);

//Article fournisseur
List<ArticleFournisseur> GetArticleFournisseurList(Criteria criteria = null);
List<ArticleFournisseur> GetArticleFournisseur (string refArticle, string numeroTiers);

//Nomenclature
List<NomenclatureArticle> GetNomenclature(string refArticle, int idGamme1 = 0, int idGamme2 = 0);

//Enuméré de conditionnement
IList<EnumereConditionnement> GetEnumereConditionnementList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
IList<EnumereConditionnement> EnumereConditionnementByRefArticle(string refArticle);
IList<EnumereConditionnement> GetEnumereConditionnementByRefArticle(string refArticle);
IList<EnumereConditionnement> EnumereConditionnementByIdConditionnement(string idConditionnement);
IList<EnumereConditionnement> GetEnumereConditionnementByIdConditionnement(int idConditionnement);

//conditionnement d'article
IList<ConditionnementArticle> GetConditionnementArticleList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
IList<ConditionnementArticle> GetConditionnementArticleListByRefArticle(string refArticle);
```

```
▪ IList<ConditionnementArticle> ConditionnementArticleListByRefArticle(string
  refArticle);

//Enuméré de conditionnement
▪ IList<Conditionnement> ConditionnementList();
▪ IList<Conditionnement> GetConditionnementList();
```

Fonction :

```
Article GetArticle(string reference)
```

Description :

Retourne l'article par sa référence.
Renvoi null si la référence d'article n'existe pas dans la base de données.
Si le paramètre reference est null, une ArgumentException est levée.

Exemple :

```
IArticleService service = new
ArticleService("http://localhost:12345/Webservices100/BIJOU/");
Article article = service.GetArticle("CHAOR42");
```

Fonction :

```
List<Article> GetList(Criteria criteria = null, List<Order> orders = null, int
pageNumber = 0, int rowsPerPage = 0)
```

Description :

Retourne la liste des articles.

Exemples :

```
IArticleService service =
new.ArticleService("http://localhost:12345/Webservices100/BIJOU/");
List <Article> articleList = service.GetList();
```

Fonction :

```
int GetCount(Criteria criteria = null)
```

Description :

Retourne le nombre d'articles correspondant au critère passé en paramètre.

Exemple :

```
IArticleService service =
new.ArticleService("http://localhost:12345/Webservices100/BIJOU/");
int nombreArticlesPrixVenteSup50 = service.GetCount( new CriteriaComparison("PrixVente",
ComparisonOperator.GreaterThan, 50));
```

Fonction :

```
Article Insert(Article article, bool useSageProcess = true, Guid? guidCreateur = null);
```

Description :

Ajoute un nouvel article en respectant les règles suivantes :

- Il n'existe pas d'article ayant la même référence dans la base de données.
- Les champs Reference, CodeFamille, Intitule sont définis.
- La famille associée existe.
- L'unité de vente associée existe.
- Les valeurs par défaut sont les valeurs provenant de la Famille associée.

Si le paramètre article est null, une ArgumentException est levée.

Exemple :

```
Article article = new Article ();
article.Reference = "CBLALU";
article.Intitule= "Cable aluminium";
article.CodeFamille = "CBL";
IArticleService service =
new.ArticleService("http://localhost:12345/Webservices100/BIJOU/");
Article articleInserted = service.Insert(article, utilisateurConnecte.Guid);
```

Fonction :

```
Article Insert(ParametreInsertArticle data);
```

Description :

Ajoute un nouvel article en respectant les règles suivantes :

- Il n'existe pas d'article ayant la même référence dans la base de données.
- Les champs Reference, CodeFamille, Intitule sont définis.
- La famille associée existe.
- L'unité de vente associée existe.
- Les valeurs par défaut sont les valeurs provenant de la Famille associée.

Si le paramètre data est null, une ArgumentException est levée.
Si la propriété Article de data est nulle, une ArgumentException est levée.

Exemple :

```
Article article = new ArticleGamme ();
article.Reference = "CBLALU";
article.Intitule= "Cable aluminium";
article.CodeFamille = "CBL";
article.IdGamme1 = 5 ;
article.IdGamme2 = 6 ;
……
ParametreEnumereGamme paramEnumGamme1 = new ParametreEnumereGamme()
{
        Inclure = true,
        Enumeres = new string[]{ “Rouge” , ”Vert” }
};
ParametreEnumereGamme paramEnumGamme2 = new ParametreEnumereGamme()
{
        Inclure = false,
        Enumeres = new string[]{ “XXXL” }
};

ParametresInsertArticle donnees = new ParametresInsertArticle()
{
        Article = article,
        UseSageProcess = true,
        GuidCreateur = utilisateurConnecte.Guid,
        ParametresEnumereGamme1 = paramEnumGamme1,
        ParametresEnumereGamme2 = paramEnumGamme2
};

IArticleService service =
new.ArticleService("http://localhost:12345/Webservices100/BIJOU/");
Article articleInserted = service.Insert(donnees);
```

Fonction :

- `Article Update(Article article, Guid? guidUtilisateur = null)`

Description :

Met à jour un article existant.
Si le paramètre article est null, une ArgumentException est levée.

Exemple :

```
IArticleService service =
new.ArticleService("http://localhost:12345/Webservices100/BIJOU/");
Article article = service.GetArticle("BDJ");
//Modification de l'article
article.Intitule = "Boîte de jonction";
Guid? guidUtilisateur = null;//Ou obtenu via ConnectUser
Article articleUpdated = service.Update(article, guidUtilisateur);
```

Fonction :

```
bool Delete(string reference)
```

Description :

Supprime un article existant en respectant les règles suivantes :

- Il n'existe pas de référence de l'article dans :
  - Les stocks d'article
  - Les stocks de gamme d'article
  - Les nomenclatures
  - Les lignes de documents
  - Les lignes d'abonnement
  - Les lots FIFO
  - Les tarifs
  - Les tarifs SELECT
  - Les plannings de projet
  - Les agendas
  - Les lots-série
  - Les historiques de projet

Renvoie true si l'article a été suprimé

Exemple :

```
IArticleService service =
new.ArticleService("http://localhost:12345/Webservices100/BIJOU/");
bool deleted = service.Delete("BDJ");
```

Fonction :
```
List<UniteVente> GetUniteVenteList()
```

Description :
Retourne la liste des unités de vente.

Exemple :
```
IArticleService service =
new.ArticleService("http://localhost:12345/Webservices100/BIJOU/");
List<UniteVente> unites = service.GetUniteVenteList();
```

Fonction :
```
UniteVente GetUniteVenteById(int id)
```

Description :
Retourne l'unité de vente ayant l'identifiant passé en paramètre.

Exemple :
```
IArticleService service =
new.ArticleService("http://localhost:12345/Webservices100/BIJOU/");
UniteVente uniteDeVente = service.GetUniteVenteById(3) ;
```

Fonction :
```
UniteVente GetUniteVenteByName(string name)
```

Description :
Retourne l'unité de vente portant le nom passé en paramètre.
Si le paramètre name est null, une ArgumentException est levée.

Exemple :
```
IArticleService service =
new.ArticleService("http://localhost:12345/Webservices100/BIJOU/");
UniteVente uniteDeVente = service.GetUniteVenteByName("Heure");
```

Fonction :
```
List<Catalogue> GetCatalogueList()
```

Description :
Retourne la liste des catalogues.
Nota : La propriété SousCatalogues contient les catalogues enfants.

Exemple :
```
IArticleService service =
new.ArticleService("http://localhost:12345/Webservices100/BIJOU/");
List<Catalogue> catalogues = service.GetCatalogueList();
```

Fonction :
```
Catalogue GetCatalogue(int id)
```

Description :
Retourne le catalogue par son identifiant.

Exemple :
```
IArticleService service =
new.ArticleService("http://localhost:12345/Webservices100/BIJOU/");
Catalogue catalogue = service.GetCatalogue(17);
```

Fonction :
```
List<Catalogue> GetCataloguesByParent(int parentId)
```

Description :
Retourne les catalogues enfant du catalogue ayant comme identifiant la valeur du paramètre parentId.

Exemple :
```
IArticleService service =
new.ArticleService("http://localhost:12345/Webservices100/BIJOU/");
List<Catalogue> catalogues = service.GetCataloguesByParent(12);
```

Fonction :
```
List<StatistiqueArticle> GetStatistiqueList()
```

Description :
Retourne la liste des statistiques d'article.

Exemple :
```
IArticleService service =
new.ArticleService("http://localhost:12345/Webservices100/BIJOU/");
List<StatistiqueArticle> statistiques = service.GetStatistiqueList();
```

Fonction :
```
List<StatistiqueArticle> GetStatistiqueArticleListByIdChamp(short idChamp)
```

Description :
Retourne la liste des statistiques d'article ayant l'index de champ passé en paramètre.

Exemple :
```
IArticleService service =
new.ArticleService("http://localhost:12345/Webservices100/BIJOU/");
List<StatistiqueArticle> statistiques = service.GetStatistiqueArticleListByIdChamp(9);
```

Fonction :
```
List<ArticleFournisseur> GetArticleFournisseurList()
```

Description :
Retourne la liste des articles fournisseur.

Exemple :
```
IArticleService service =
new.ArticleService("http://localhost:12345/Webservices100/BIJOU/");
List<ArticleFournisseur> statistiques = service.GetArticleFournisseurList();
```

Fonction :
```
List<ArticleFournisseur> GetArticleFournisseur(string refArticle , string numeroTiers)
```

Description :
Retourne la liste des articles fournisseur en relation avec la référence d'article passée en paramètre.
Si le paramètre refArticle ou numeroTiers est null, une ArgumentException est levée.

Exemple :
```
IArticleService service =
new.ArticleService("http://localhost:12345/Webservices100/BIJOU/");
List<ArticleFournisseur> statistiques = service.GetArticleFournisseur (" CHAOR42 ",
"ECLAT");
```

Fonction :
```
List<NomenclatureArticle> GetNomenclature(string refArticle, int idGamme1 = 0, int
idGamme2 = 0)
```

Description :
Retourne la nomenclature associée à une référence d'article.

Exemple :
```
IArticleService service =
new.ArticleService("http://localhost:12345/Webservices100/BIJOU/");
IList< NomenclatureArticle > compositionNomenclature =
service.GetNomenclature("CHAOR42");
```

Fonction :
```
IList<EnumereConditionnement> GetEnumereConditionnementList (Criteria criteria = null,
List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0)
```

Description :
Retourne la liste des énumérés de conditionnement.

Exemple :
```
IArticleService service =
new.ArticleService("http://localhost:12345/Webservices100/BIJOU/");
IList<EnumereConditionnement> conditionnements =
service.GetEnumereConditionnementList();
```

Fonction :
```
IList<EnumereConditionnement> EnumereConditionnementByRefArticle(string refArticle)
```

Description :
Retourne la liste des énumérés de conditionnement du conditionnement de l'article ciblé.

Exemple :
```
IArticleService service =
new.ArticleService("http://localhost:12345/Webservices100/BIJOU/");
IList<EnumereConditionnement> conditionnements =
service.EnumereConditionnementList("BRAAR10");
```

Fonction :
```
IList<EnumereConditionnement> EnumereConditionnementByIdConditionnement (string
idConditionnement)
```

Description :
Retourne la liste des énumérés de conditionnement de l'identifiant de conditionnement passé en paramètre.
Une exception est levée si le paramètre idCondtionnement n'est pas convertissable en une valeur de type Entier.

Exemple :
```
IArticleService service =
new.ArticleService("http://localhost:12345/Webservices100/BIJOU/");
IList<EnumereConditionnement> enumeresConditionnement3 =
service.EnumereConditionnementbyIdConditionnement("3");
```

Fonction :
```
IList<EnumereConditionnement> GetEnumereConditionnementByIdConditionnement (int idConditionnement)
```

Description :
Retourne la liste des énumérés de conditionnement de l'identifiant de conditionnement passé en paramètre.

Exemple :
```
IArticleService service = new.ArticleService("http://localhost:12345/Webservices100/BIJOU/");
IList<EnumereConditionnement> enumeresConditionnement3 = service.GetEnumereConditionnementByIdConditionnement (3);
```

Fonction :
```
IList<ConditionnementArticle> GetConditionnementArticleList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0)
```

Description :
Retourne la liste des conditionnements d'article.

Exemple :
```
IArticleService service = new.ArticleService("http://localhost:12345/Webservices100/BIJOU/");
IList< ConditionnementArticle > conditionnementArticleList = service.GetConditionnementArticleList();
```

Fonction :
```
IList<ConditionnementArticle> GetConditionnementArticleListByRefArticle (string refArticle)
```

Description :
Retourne la liste des conditionnements d'article associés à la référence d'article passée en paramètre.

Exemple :
```
IArticleService service = new.ArticleService("http://localhost:12345/Webservices100/BIJOU/");
IList< ConditionnementArticle > conditionnementArticleList = service.GetConditionnementArticleListByRefArticle("EM040");
```

Fonction :
```
IList<Conditionnement> ConditionnementList ()
```

Description :
Retourne la liste des conditionnements configuré dans Sage.

Exemple :
```
IArticleService service =
new.ArticleService("http://localhost:12345/Webservices100/BIJOU/");
IList< Conditionnement > conditionnements = service.ConditionnementList();
```

Fonction :
```
IList<Conditionnement> GetConditionnementList ()
```

Description :
Retourne la liste des conditionnements configuré dans Sage.

Exemple :
```
IArticleService service =
new.ArticleService("http://localhost:12345/Webservices100/BIJOU/");
IList<Conditionnement> conditionnements = service.GetConditionnementList();
```

d) CollaborateurService

Fonctions disponibles :

- `Collaborateur GetCollaborateur(int id);`
- `int GetCount(Criteria criteria = null);`
- `List<Collaborateur> GetList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);`
- `Collaborateur Insert(Collaborateur collaborateur, Guid? guidCreateur = null);`
- `Collaborateur Update(Collaborateur collaborateur);`
- `bool Delete(int id);`

Fonction :

```
Collaborateur GetCollaborateur(int id);
```

Description :

Retourne un collaborateur par son indice.

Exemple :

```
ICollaborateurService service = new.
CollaborateurService("http://localhost:12345/Webservices100/BIJOU/");
Collaborateur collaborateur = service.GetCollaborateur(1);
```

Fonction :

```
int GetCount(Criteria criteria = null);
```

Description :

Retourne le nombre de collaborateurs correspondants au critère passé en paramètre.

Exemple :

```
ICollaborateurService service = new.
CollaborateurService("http://localhost:12345/Webservices100/BIJOU/");
int nombrecollaborateur = service.GetCount();
```

Fonction :

```
List<Collaborateur> GetList(Criteria criteria = null, List<Order> orders = null, int
pageNumber = 0, int rowsPerPage = 0);
```

Description :

Retourne la liste des collaborateurs correspondants au critère passé en paramètre.

Exemple :

```
ICollaborateurService service = new.
CollaborateurService("http://localhost:12345/Webservices100/BIJOU/");
List<Collaborateur> list = service.GetList();
```

Fonction :
```
Collaborateur Insert(Collaborateur collaborateur, Guid? guidCreateur = null);
```

Description :
Insère un collaborateur.
Si le paramètre collaborateur est nul, une ArgumentException est levée.

Exemple :
```
ICollaborateurService service = new.
CollaborateurService("http://localhost:12345/Webservices100/BIJOU/");
Collaborateur collaborateur = new Collaborateur();
//Alimentation des propriétés du collaborateur
Collaborateur collaborateurInserted = service.Insert(collaborateur,
utilisateurConnecte.Guid);
```

Fonction :
```
Collaborateur Update(Collaborateur collaborateur);
```

Description :
Met à jour un collaborateur.
Si le paramètre collaborateur est nul, une ArgumentException est levée.

Exemple :
```
ICollaborateurService service = new.
CollaborateurService("http://localhost:12345/Webservices100/BIJOU/");
Collaborateur collaborateur = service. GetCollaborateur(1);
//Modification des propriétés du collaborateur
Collaborateur collaborateurUpdated = service.Update(collaborateur);
```

Fonction :
```
bool Delete(int id);
```

Description :
Supprime un collaborateur.
Retourne true si le collaborateur a été supprimé.

Exemple :
```
ICollaborateurService service = new.
CollaborateurService("http://localhost:12345/Webservices100/BIJOU/");
bool collaborateurDeleted = service.Delete(1);
```

e) ContactTiersService

Fonctions disponibles :

- `ContactTiers GetContactTiers(string numeroTiers, int idContact);`
- `List<ContactTiers> GetList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);`
- `List<ContactTiers> GetListByTiers(string numeroTiers, List<Order> orders = null);`
- `int GetCount(Criteria criteria = null);`
- `ContactTiers Insert(ContactTiers contact, Guid? guidCreateur = null);`
- `ContactTiers Update(ContactTiers contact);`
- `bool Delete(string numeroTiers, int idContact);`
- `ContactTiers CheckContactTiers(string numeroTiers, string contactName, string contactFirstName);`

Fonction :

```
ContactTiers GetContactTiers(string numeroTiers, int idContact);
```

Description :

Retourne un contact par son Id et par l'identifiant du tiers.
Lève une ArgumentException si le paramètre numeroTiers est nul

Exemple :

```
IContactTiersService service = new
ContactTiersService("http://localhost:12345/Webservices100/BIJOU/");
ContactTiers contact = service.GetContactTiers("BAGUES", 1);
```

Fonction :

```
List<ContactTiers> GetList(Criteria criteria = null, List<Order> orders = null, int
pageNumber = 0, int rowsPerPage = 0);
```

Description :

Retourne la liste des contacts répondant aux critères passés en paramètre.

Exemple :

```
IContactTiersService service = new
ContactTiersService("http://localhost:12345/Webservices100/BIJOU/");
Criteria criteria = new CriteriaComparison("Fonction", ComparisonOperator.Like,
"Gérant");
List<ContactTiers> list = service.GetList(criteria);
```

Fonction :
```
List<ContactTiers> GetListByTiers(string numeroTiers, List<Order> orders = null);
```

Description :
Retourne la liste des contacts d'un tiers.
Lève une ArgumentException si le paramètre numeroTiers est nul

Exemple :
```
IContactTiersService service = new
ContactTiersService("http://localhost:12345/Webservices100/BIJOU/");
List<ContactTiers> list = service.GetListByTiers("BAGUES");
```

Fonction :
```
int GetCount(Criteria criteria = null);
```

Description :
Retourne le nombre de contacts répondant aux critères passés en paramètre.

Exemple :
```
IContactTiersService service = new
ContactTiersService("http://localhost:12345/Webservices100/BIJOU/");
int nb = service.GetCount();
```

Fonction :

```
ContactTiers Insert(ContactTiers contact, Guid? guidCreateur = null);
```

Description :

Ajoute un nouveau contact en respectant les règles suivantes :

- Le tiers associé doit être renseigné et exister.
- Le champ 'Nom' doit être renseigné.
- Le type de contact renseigné doit exister.
- Le service de contact renseigné doit exister.
- Un contact avec le même 'Nom-Prenom' pour le tiers associé ne doit pas exister.

Retourne le contact ajouté.
Lève une ArgumentException si le paramètre contact est nul

Exemple :

```
ContactTiers contact = new ContactTiers();
contact.Id = 0;
contact.NumeroTiers = "BAGUES";
contact.Nom = "Dal Cengio";
contact.Prenom = "Eric";
contact.Fonction = "Gérant";
contact.Civilite = Civilite.M;
contact.Fax = "+352 27 30 33 59";
contact.Telephone = "+352 27 30 33 69";
contact.Gsm = "+352 691 48 24 61";
contact.Email = "info@test.lu";

IContactTiersService service = new
ContactTiersService("http://localhost:12345/Webservices100/BIJOU/");
ContactTiers contactInserted = service.Insert(contact, utilisateurConnecte.Guid);
```

Fonction :
```
ContactTiers Update(ContactTiers contact);
```

Description :
Met à jour un contact existant en respectant les règles suivantes :

- Le tiers associé doit être renseigné et exister.
- Le champ 'Nom' doit être renseigné.
- Le type de contact renseigné doit exister.
- Le service de contact renseigné doit exister.
- Un contact avec le même 'Nom-Prenom' pour le tiers associé ne doit pas exister.
- Le contact doit exister.

Retourne le contact mis à jour.
Lève une ArgumentException si le paramètre contact est nul

Exemple :
```
IContactTiersService service = new
ContactTiersService("http://localhost:12345/Webservices100/BIJOU/");
ContactTiers contact = service.GetContact("BAGUES", 1);
//Modification du contact
service.Update(contact);
```

Fonction :
```
bool Delete(string numeroTiers, int idContact);
```

Description :
Supprime un contact existant en respectant les règles suivantes :

- Aucuns dossiers de recouvrement ne doivent être liés au contact.

Retourne true si le contact a été supprimée

Exemple :
```
IContactTiersService service = new
ContactTiersService("http://localhost:12345/Webservices100/BIJOU/");
ContactTiers contact = service.GetContact("BAGUES", 1);
Bool deleted = Service.CommonService.ContactTiersService.Delete(contact.NumeroTiers,
contact.Id);
```

Fonction :

```
ContactTiers CheckContactTiers(string numeroTiers, string contactName, string contactFirstName);
```

Description :

Détermine si un contact existe pour un tiers en utilisant le numéro du tiers, le nom et le prénom du contact.
Retourne null si le contact n'existe pas sur le numeroTiers définit en paramètre.

Exemple :

```
IContactTiersService service = new ContactTiersService("http://localhost:12345/Webservices100/BIJOU/");
ContactTiers contact = service.CheckContactTiers("BAGUES", "DUPONT", "JEAN");
```

f) DocumentService

Fonctions disponibles :

- IList<Document> GetList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
- IList<DocumentAchat> GetListDocumentAchat(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
- IList<DocumentVente> GetListDocumentVente(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
- IList<DocumentStock> GetListDocumentStock(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
- IList<DocumentInterne> GetListDocumentInterne(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
- IList<Document> GetListDocumentEnCoursByDomaine(DomaineDocument domaine, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
- Document GetDocument(TypeDocument typeDocument, string numeroDocument);
- IList<LigneDocument> GetListLigneDocument(TypeDocument typeDocument, string numeroDocument);
- LigneDocument GetLigneDocument(int idLigne);
- IList<DocumentTypeLibelle> GetListTypeDocumentLibelle();
- DocumentTypeLibelle GetTypeDocumentLibelle(TypeDocument typeDocument);
- DocumentTypeLibelle GetTypeDocumentInterneByLibelle(string libelle);
- string InsertDocument(Document document, bool useSageProcess = true, Guid? guidCreateur = null);
- string InsertEntete(TypeDocument typeDocument, DateTime date, string numeroDocument, string numeroTiers, Guid? guidCreateur = null);
- string InsertDocumentEtLignes(Document document, List<LigneDocument> ligneList, bool useSageProcess = true, Guid? guidCreateur = null, List<ParametreNomenclature> parametreNomenclatureList = null);
- LigneDocument InsertLigne(LigneDocument ligneDocument, bool useSageProcess = true, Guid? guidCreateur = null, List<DetailParametreNomenclature> detailParametreNomenclatureList = null);
- List<LigneDocument> InsertLigneDocumentList(List<LigneDocument> ligneList, bool useSageProcess = true, Guid? guidCreateur = null, List<ParametreNomenclature> parametreNomenclatureList = null);
- bool DeleteLigneDocument(int idLigne, Guid? guidUtilisateur = null);
- bool DeleteLigneDocumentList(List<int> idLigneList, Guid? guidUtilisateur = null);
- bool DeleteLignesDocument(TypeDocument typeDocument,string numeroDocument, Guid? guidUtilisateur = null);
- List<LigneDocument> ReplaceDocumentLines(TypeDocument typeDocument, string numeroDocument, List<LigneDocument> ligneDocumentList, bool useSageProcess = true, List<ParametreNomenclature> parametreNomenclatureList = null);
- IList<Reglement> InsertReglements(String numDocument, TypeDocument typeDocument, IList<Reglement> reglementList, Guid? guidCreateur = null);
- IList<Reglement> GetReglementsDocument(string numeroDocument, TypeDocument typeDocument);
- bool UpdateInfosLibres(TypeDocument typeDocument, string numeroDocument, List<InfoLibre> infosLibresList);
- bool UpdateChampsDocument(TypeDocument typeDocument, string numeroDocument, ChampsDocumentUpdate champs, object value, Guid? guidUtilisateur = null);
- bool UpdateInfosLibresLigne(int idLigne, List<InfoLibre> infoLibresList);
- List<DocumentInterneInfo> GetListDocumentInterneInfo();
- DocumentInterneInfo GetDocumentInterneInfoByType(TypeDocument typeDocument);

```
DocumentInterneInfo GetDocumentInterneInfo(string intitule);
void AppliquerModeleReglement(TypeDocument typeDocument, string numeroDocument, int idModeleReglement, Guid? guidUtilisateur);
IList<ConditionReglementDocument> DefinirConditionReglement(TypeDocument typeDocument, string numeroDocument, List<ConditionReglementDocument> listConditionReglement);
IList<ConditionReglementDocument> GetConditionReglementDocuments(TypeDocument typeDocument, string numeroDocument);
IList<MotifPerteDevis> GetMotifPerteDevisList();
void AppliquerFraisPort(TypeDocument typeDocument, string numeroDocument, decimal montantPort, TypePrix? typePrix);
void AppliquerExpeditionDefaut(TypeDocument typeDocument, string numeroDocument);
void AppliquerModeExpedition(TypeDocument typeDocument, string numeroDocument, short idModeExpedition);
bool UpdateListChampsDocument(ListUpdateEnteteDoc updateInfoEnteteData);
```

Fonction :
```
IList<Document> GetList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
```

Description :
Retourne la liste des documents correspondants au critère passé en paramètre.

Exemple :
```
IDocumentService service = new DocumentService("http://localhost:12345/Webservices100/BIJOU/");
IList<Document> list = service.GetList();
```

Fonction :
```
IList<DocumentAchat> GetListDocumentAchat(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
```

Description :
Retourne la liste des documents d'achat correspondants au critère passé en paramètre.

Exemple :
```
IDocumentService service = new DocumentService("http://localhost:12345/Webservices100/BIJOU/");
IList<DocumentAchat> list = service.GetListDocumentAchat();
```

Fonction :
```
IList<DocumentVente> GetListDocumentVente(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
```

Description :
Retourne la liste des documents de vente correspondants au critère passé en paramètre.

Exemple :
```
IDocumentService service = new DocumentService("http://localhost:12345/Webservices100/BIJOU/");
IList<DocumentVente> list = service.GetListDocumentVente();
```

Fonction :
```
IList<DocumentStock> GetListDocumentStock(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
```

Description :
Retourne la liste des documents de stock correspondants au critère passé en paramètre.

Exemple :
```
IDocumentService service = new DocumentService("http://localhost:12345/Webservices100/BIJOU/");
IList<DocumentStock> list = service.GetListDocumentStock();
```

Fonction :
```
IList<DocumentInterne> GetListDocumentInterne(Criteria criteria = null, List<Order>
orders = null, int pageNumber = 0, int rowsPerPage = 0);
```

Description :
Retourne la liste des documents internes correspondants au critère passé en paramètre.

Exemple :
```
IDocumentService service = new
DocumentService("http://localhost:12345/Webservices100/BIJOU/");
IList<DocumentInterne> list = service.GetListDocumentInterne();
```

Fonction :
```
IList<Document> GetListDocumentEnCoursByDomaine(DomaineDocument domaine, List<Order>
orders = null, int pageNumber = 0, int rowsPerPage = 0);
```

Description :
Retourne la liste des documents en cours du domaine passé en paramètre.

Exemple :
```
IDocumentService service = new
DocumentService("http://localhost:12345/Webservices100/BIJOU/");
IList<Document> list = service.GetListDocumentEnCoursByDomaine(DomaineDocument.Achat);
```

Fonction :
```
Document GetDocument(TypeDocument typeDocument, string numeroDocument);
```

Description :
Retourne un document par son type de document et son numéro.

Exemple :
```
IDocumentService service = new
DocumentService("http://localhost:12345/Webservices100/BIJOU/");
Document document = service.GetDocument(TypeDocument.BonDeCommandeAchat, "FBC00003");
```

Fonction :
```
IList<LigneDocument> GetListLigneDocument(TypeDocument typeDocument, string
numeroDocument);
```

Description :
Retourne la liste des lignes d'un document par son type de document et son numéro.

Exemple :
```
IDocumentService service = new
DocumentService("http://localhost:12345/Webservices100/BIJOU/");
IList<LigneDocument> list = service.GetListLigneDocument(TypeDocument.BonAvoirVente,
"BA00003");
```

Fonction :
```
LigneDocument GetLigneDocument(int idLigne);
```

Description :
Retourne une ligne de document par son identifiant.

Exemple :
```
IDocumentService service = new
DocumentService("http://localhost:12345/Webservices100/BIJOU/");
LigneDocument ligne = service.GetLigneDocument(2884);
```

Fonction :
```
IList<DocumentTypeLibelle> GetListTypeDocumentLibelle();
```

Description :
Retourne la liste des DocumentTypeLibelle.

Exemple :
```
IDocumentService service = new
DocumentService("http://localhost:12345/Webservices100/BIJOU/");
IList<DocumentTypeLibelle> list = service.GetListTypeDocumentLibelle();
```

Fonction :
```
DocumentTypeLibelle GetTypeDocumentLibelle(TypeDocument typeDocument);
```

Description :
Retourne le DocumentTypeLibelle correspondant au type de document passé en paramètre.

Exemple :
```
IDocumentService service = new
DocumentService("http://localhost:12345/Webservices100/BIJOU/");
DocumentTypeLibelle docTypeLibelle =
service.GetTypeDocumentLibelle(TypeDocument.FactureAchat);
```

Fonction :
```
DocumentTypeLibelle GetTypeDocumentInterneByLibelle(string libelle);
```

Description :
Retourne le DocumentTypeLibelle de document interne correspondant au libellé passé en paramètre.

Exemple :
```
IDocumentService service = new
DocumentService("http://localhost:12345/Webservices100/BIJOU/");
DocumentTypeLibelle docTypeLibelle =
service.GetTypeDocumentInterneByLibelle("Reversement commission");
```

Fonction :
```
string InsertDocument(Document document, bool useSageProcess = true, Guid? guidCreateur = null);
```

Description :
Insère l'entete de document.
Renvoie le numéro de document inséré.

Exemple :
```
IDocumentService service = new DocumentService("http://localhost:12345/Webservices100/BIJOU/");
Document doc = new DocumentVente(TypeDocument.BonDeCommandeVente);
//Initialisation du document
String numDoc = service.InsertDocument(doc, guidCreateur :utilisateurConnecte.Guid);
```

Fonction :
```
string InsertEntete(TypeDocument typeDocument, DateTime date, string numeroDocument, string numeroTiers, Guid? guidCreateur = null);
```

Description :
Insère un entete de document avec les paramètres fournis.
Renvoie le numéro de document inséré.

Exemple :
```
IDocumentService service = new DocumentService("http://localhost:12345/Webservices100/BIJOU/");
String numDocInserted = service.InsertEntete(TypeDocument.FactureVente, DateTime.Now, "FACTURE1", "CARAT", utilisateurConnecte.Guid);
```

Fonction :

```
string InsertDocumentEtLignes(Document document, List<LigneDocument> ligneList, bool
useSageProcess = true, Guid? guidCreateur = null, List<ParametreNomenclature>
parametreNomenclatureList = null);
```

Description :

Insère un entete de document avec ses lignes passées en paramètre.
Le paramètre parametreNomenclatureList fonctionne comme un dictionnaire dont la clé est {ReferenceArticle, IdGamme1, IdGamme2} et la valeur associée est une liste de DetailParametreNomenclature.
Lorsque les webServices100 traitent l'injection d'une ligne d'article dont l'article est un article à nomenclature. Ils regardent dans le paramètre parametreNomenclatureList, s'il existe une clé identique aux informations de la ligne d'article traitée. S'ils trouvent cette correspondance, il utilise la liste de DetailParametreNomenclature associée à cette clé pour composer la nomenclature (liste de lignes d'article) à insérer.
Si aucune clé correspondante n'est trouvée, la composition de la nomenclature sera faite en fonction de useSageProcess.
Si useSageProcess = true, la nomenclature sera faite à partir de la nomenclature de l'article.
Si useSageProcess = false, il n'y aura aucune nomenclature de générer.
Renvoie le numéro de document inséré.

Exemple :

```
IDocumentService service = new
DocumentService("http://localhost:12345/Webservices100/BIJOU/");
Document doc = new DocumentVente(TypeDocument.BonDeCommandeVente);
//Initialisation du document
List<LigneDocument> ligneList = new List<LigneDocument>();
//Ajout des lignes de document à la liste ligneList
String numDocInserted = service.InsertDocumentEtLignes(doc, ligneList,
guidCreateur :utilisateurConnecte.Guid);
```

Fonction :

```
LigneDocument InsertLigne(LigneDocument ligneDocument, bool useSageProcess = true, Guid?
guidCreateur = null, List<DetailParametreNomenclature> detailParametreNomenclatureList =
null);
```

Description :

Insère une ligne de document.
Le paramètre detailParametreNomenclatureList sera utilisé uniquement si la ligne insérée est une ligne d'article et que l'article est un article à nomenclature.
Il permet de définir la composition de la nomenclature (liste de lignes d'article) à insérer.
Retourne la ligne de document insérée
Si le paramètre ligneDocument est nul, une ArgumentException est levée.

Exemple :

```
IDocumentService service = new
DocumentService("http://localhost:12345/Webservices100/BIJOU/");
LigneDocument ligne = new LigneArticle(TypeDocument.FactureVente);
LigneDocument ligneInserted = service.InsertLigne(ligne,
guidCreateur :utilisateurConnecte.Guid);
```

Fonction :

```
List<LigneDocument> InsertLigneDocumentList(List<LigneDocument> ligneList, bool
useSageProcess = true, Guid? guidCreateur = null, List<ParametreNomenclature>
parametreNomenclatureList = null);
```

Description :

Insère une liste de lignes de documents.
Le paramètre parametreNomenclatureList fonctionne comme un dictionnaire dont la clé est {ReferenceArticle, IdGamme1, IdGamme2} et la valeur associée est une liste de DetailParametreNomenclature.
Lorsque les webServices100 traitent l'injection d'une ligne d'article dont l'article est un article à nomenclature. Ils regardent dans le paramètre parametreNomenclatureList, s'il existe une clé identique aux informations de la ligne d'article traitée. S'ils trouvent cette correspondance, il utilise la liste de DetailParametreNomenclature associée à cette clé pour composer la nomenclature (liste de lignes d'article) à insérer.
Si aucune clé correspondante n'est trouvée, la composition de la nomenclature sera faite en fonction de useSageProcess.
Si useSageProcess = true, la nomenclature sera faite à partir de la nomenclature de l'article.
Si useSageProcess = false, il n'y aura aucune nomenclature de générer.

Retourne la liste de lignes de documents insérées
Si le paramètre ligneList est nul, une ArgumentException est levée.

Exemple :

```
IDocumentService service = new
DocumentService("http://localhost:12345/Webservices100/BIJOU/");
List<LigneDocument> ligneList = new List<LigneDocument>();
List<LigneDocument> ligneListInserted = service. InsertLigneDocumentList (ligneList,
guidCreateur : utilisateurConnecte.Guid);
```

Fonction :

```
bool DeleteLigneDocument(int idLigne, Guid? guidUtilisateur = null);
```

Description :

Supprime une ligne de document.
Retourne true si la ligne de document a été supprimée.

Exemple :

```
IDocumentService service = new
DocumentService("http://localhost:12345/Webservices100/BIJOU/");
Guid? guidUtilisateur = null;//Ou obtenu via ConnectUser
bool ligneDeleted = service. DeleteLigneDocument (1);
```

Fonction :
```
bool DeleteLigneDocumentList(List<int> idLigneList, Guid? guidUtilisateur = null);
```

Description :
Supprime une liste de ligne de document.
Retourne true si la liste de ligne de document a été supprimée.
Si le paramètre `idLigneList` est nul, une ArgumentException est levée.

Exemple :
```
IDocumentService service = new
DocumentService("http://localhost:12345/Webservices100/BIJOU/");
List< int > ligneIdList = new List< int >();
Guid? guidUtilisateur = null;//Ou obtenu via ConnectUser
bool deleted = service. DeleteLigneDocumentList (ligneIdList, guidUtilisateur);
```

Fonction :
```
bool DeleteLignesDocument(TypeDocument typeDocument,string numeroDocument, Guid?
guidUtilisateur = null);
```

Description :
Supprime une liste de ligne de document dont le type et le numéro de document est passé en paramètre.
Retourne true si la liste de ligne de document a été supprimée.
Si le paramètre numeroDocument est nul, une ArgumentException est levée.

Exemple :
```
IDocumentService service = new
DocumentService("http://localhost:12345/Webservices100/BIJOU/");
Guid? guidUtilisateur = null;//Ou obtenu via ConnectUser
bool deleted = service. DeleteLignesDocument (TypeDocument.FactureAchat, "FFA00001",
guidUtilisateur);
```

Fonction :

```
List<LigneDocument> ReplaceDocumentLines(TypeDocument typeDocument, string
numeroDocument, List<LigneDocument> ligneDocumentList, bool useSageProcess = true, Guid?
guidCreateur = null, List<ParametreNomenclature> parametreNomenclatureList = null);
```

Description :

Remplace une liste de ligne de document dont le type et le numéro de document sont passés en paramètre par une nouvelle ligne de document.
Le paramètre parametreNomenclatureList fonctionne comme un dictionnaire dont la clé est {ReferenceArticle, IdGamme1, IdGamme2} et la valeur associée est une liste de DetailParametreNomenclature.
Lorsque les webServices100 traitent l'injection d'une ligne d'article dont l'article est un article à nomenclature. Ils regardent dans le paramètre parametreNomenclatureList, s'il existe une clé identique aux informations de la ligne d'article traitée. S'ils trouvent cette correspondance, il utilise la liste de DetailParametreNomenclature associée à cette clé pour composer la nomenclature (liste de lignes d'article) à insérer.
Si aucune clé correspondante n'est trouvée, la composition de la nomenclature sera faite en fonction de useSageProcess.
Si useSageProcess = true, la nomenclature sera faite à partir de la nomenclature de l'article.
Si useSageProcess = false, il n'y aura aucune nomenclature de générer.

Retourne la nouvelle liste de ligne de document.
Si le paramètre numeroDocument ou `ligneDocumentList` est nul, une ArgumentException est levée.

Exemple :

```
IDocumentService service = new
DocumentService("http://localhost:12345/Webservices100/BIJOU/");
List< LigneDocument> list = new List< LigneDocument>() ;
List< LigneDocument> listInserted = service. ReplaceDocumentLines
(TypeDocument.FactureAchat, "FFA00001",list, guidCreateur:UtilisateurConnecte.Guid);
```

Fonction :

```
IList<Reglement> InsertReglements(String numDocument, TypeDocument typeDocument,
IList<Reglement> reglementList, Guid? guidCreateur = null);
```

Description :

Ajoute la liste des réglements passés en paramètre au document correspondant aux paramètres numDocument et typeDocument.
Retourne la liste de règlement ajoutée.
Si le paramètre numDocument ou `reglementList` est nul, une ArgumentException est levée.

Exemple :

```
IDocumentService service = new
DocumentService("http://localhost:12345/Webservices100/BIJOU/");
List<Reglement> reglementList = new List<Reglement>();
//Ajout de règlement à liste reglementList
service.InsertReglements("FA00015", TypeDocument.FactureVente, reglementList,
utilisateurConnecte.Guid);
```

Fonction :

```
IList<Reglement> GetReglementsDocument(string numeroDocument, TypeDocument
typeDocument);
```

Description :

Renvoie la liste des réglements associés au document correspondant aux paramètres.
Si le paramètre numeroDocument est nul, une ArgumentException est levée.

Exemple :

```
IDocumentService service = new
DocumentService("http://localhost:12345/Webservices100/BIJOU/");
IList<Reglement> list = service.GetReglementsDocument("FBC00002",
TypeDocument.BonDeCommandeAchat);
```

Fonction :

```
bool UpdateInfosLibres(TypeDocument typeDocument, string numeroDocument,
List<InfoLibre> infosLibresList);
```

Description :

Met à jour les informations libres du document défini par les paramètres typeDocument et numeroDocument.
Renvoie vrai si la mise à jour est effectuée.
La méthode lève une exception si :

- Le document ciblé n'existe pas.
- Le nom d'une information libre est inconnu.
- La valeur d'une information libre n'est pas convertissable dans le type attendu.

Exemple :

```
IDocumentService service = new
DocumentService("http://localhost:12345/Webservices100/BIJOU/");
Document document = service.GetDocument(TypeDocument.BonDeCommandeAchat, "FBC00002");
ListInfoLibre infosLibres = document.InfoLibres;
//Modification des informations libres
bool success= service.UpdateInfosLibres(document.TypeDocument, document.NumeroDocument,
infosLibres);
```

Fonction :

```
bool UpdateChampsDocument(TypeDocument typeDocument, string numeroDocument,
ChampsDocumentUpdate champs, object value, Guid? guidUtilisateur = null);
```

Description :

Met à jour l'information ciblée par le paramètre champs (énumération ChampsDocumentUpdate) du document typeDocument numeroDocument.
Renvoie vrai si la mise à jour est effectuée.
La méthode lève une exception si :

- Le document ciblé n'existe pas.
- Le nom d'une information libre est inconnu.
- La valeur d'une information libre n'est pas convertissable dans le type attendu.

Exemple :

```
IDocumentService service = new
DocumentService("http://localhost:12345/Webservices100/BIJOU/");
Guid? guidUtilisateur = null;//Ou obtenu via ConnectUser
bool success= service. UpdateChampsDocument (TypeDocument.BonDeCommandeAchat, "FBC00002"
, ChampsDocumentUpdate.Affaire, "WEB", guidUtilisateur);
```

Fonction :

```
bool UpdateInfosLibresLigne(int idLigne, List<InfoLibre> infoLibresList);
```

Description :

Met à jour les informations libres de la ligne idLigne.
Renvoie vrai si la mise à jour est effectuée.
La méthode lève une exception si :

- La ligne n'existe pas.
- Le nom d'une information libre est inconnu.
- La valeur d'une information libre n'est pas convertissable dans le type attendu.

Exemple :

```
IDocumentService service = new
DocumentService("http://localhost:12345/Webservices100/BIJOU/");
LigneDocument ligne = service.GetLigne(352);
ListInfoLibre infosLibres = ligne.InfoLibres;
//Modification des informations libres
bool success= service.UpdateInfosLibresLigne (ligne.Id, infoLibres);
```

Fonction :
```
List<DocumentInterneInfo> GetListDocumentInterneInfo();
```
Description :
Renvoie la liste des informations de document interne.

Exemple :
```
IDocumentService service = new
DocumentService("http://localhost:12345/Webservices100/BIJOU/");
IList<DocumentInterneInfo> list = service.GetListDocumentInterneInfo();
```

Fonction :
- `DocumentInterneInfo GetDocumentInterneInfoByType(TypeDocument typeDocument);`

Description :
Renvoie les informations de document interne pour le type de document interne passé en paramètre.
La méthode lève une exception si le type de document passé en paramètre n'est pas un des 6 type de document interne.

Exemple :
```
IDocumentService service = new
DocumentService("http://localhost:12345/Webservices100/BIJOU/");
DocumentInterneInfo info =
service.GetDocumentInterneInfo(TypeDocument.DocumentInterne1);
```

Fonction :
```
DocumentInterneInfo GetDocumentInterneInfo(string intitule);
```
Description :
Renvoie les informations de document interne par l'intitulé de document interne.
La méthode lève une exception si l'intitulé de document passé en paramètre n'est pas un des intitulés de document interne.

Exemple :
```
IDocumentService service = new
DocumentService("http://localhost:12345/Webservices100/BIJOU/");
DocumentInterneInfo info = service. GetDocumentInterneInfo ("Bon intervention");
```

Fonction :

```
void AppliquerModeleReglement(TypeDocument typeDocument, string numeroDocument, int idModeleReglement, Guid? guidUtilisateur);
```

Description :

Applique le modèle de réglement {idModeleReglement} sur le document { typeDocument, numeroDocument }.

Lève une exception si :

- Le numéro de document est null ou vide,
- Le document n'existe pas,
- Le document est en cours d'utilisation,
- Le document est validé,
- Le document est une facture comptabilisée,
- Le domaine du document n'est pas un des domaines suivants [Achat, Vente],
- Le modèle de règlement idModeleReglement n'existe pas.

Exemple :

```
IDocumentService service = new DocumentService("http://localhost:12345/Webservices100/BIJOU/");
Document document = service.GetDocument(TypeDocument.BonDeCommandeAchat, "FBC00002");
service.AppliquerModeleReglement(document.TypeDocument, document.NumeroDocument, 1, guidUtilisateur);
```

Fonction :
```
IList<ConditionReglementDocument> DefinirConditionReglement(TypeDocument typeDocument,
string numeroDocument, List<ConditionReglementDocument> listConditionReglement);
```

Description :

Défini la liste des conditions de réglement sur le document { typeDocument, numeroDocument }.

Lève une exception si :

- Le numéro de document est null ou vide,
- Le document n'existe pas,
- Le document est en cours d'utilisation,
- Le document est validé,
- Le document est une facture comptabilisée,
- Le domaine du document n'est pas un des domaines suivants [Achat, Vente],
- Le modèle de règlement idModeleReglement n'existe pas,
- Plusieurs éléments de la liste des conditions de règlement possèdent la propriété Equilibre à vrai.

Nota : S'il n'existe pas de condition d'équilibre dans le paramètre listConditionReglement, les WebServices100 en créeront une automatiquement.

Exemple :
```
IDocumentService service = new
DocumentService("http://localhost:12345/Webservices100/BIJOU/");
Document document = service.GetDocument(TypeDocument.BonDeCommandeAchat, "FBC00002");
List<ConditionReglementDocument> newConditions = new List<ConditionReglementDocument>(){
new ConditionReglementDocument(){DatePaiement = DateTime.Now.Date.AddMonth(1),
IdReglement = 1, Pourcent = 30},
new ConditionReglementDocument(){DatePaiement = DateTime.Now.Date.AddMonth(3),
IdReglement = 1, Equilibre = true}
} ;
service. DefinirConditionReglement (document.TypeDocument, document.NumeroDocument,
newConditions);
```

Fonction :
```
IList<ConditionReglementDocument> GetConditionReglementDocuments(TypeDocument
typeDocument, string numeroDocument);
```

Description :

Renvoie la liste des conditions de règlement définis sur le document ciblé.
Lève une exception si le paramètre numeroDocument est vide ou null.

Exemple :
```
IDocumentService service = new
DocumentService("http://localhost:12345/Webservices100/BIJOU/");
IList<ConditionReglementDocument> newConditions =
service.GetConditionReglementDocuments(TypeDocument.BonDeCommandeAchat, "FBC00002");
```

Fonction :
```
IList<MotifPerteDevis> GetMotifPerteDevisList();
```
Description :
Renvoie la liste des motifs de perte de devis définis dans la gestion commerciale.

Exemple :
```
IDocumentService service = new
DocumentService("http://localhost:12345/Webservices100/BIJOU/");
IList< MotifPerteDevis > motifs = service.GetMotifPerteDevisList();
```

Fonction :
```
void AppliquerFraisPort(TypeDocument typeDocument, string numeroDocument, decimal
montantPort, TypePrix? typePrix);
```
Description :
Applique le montant des frais de port sur le document {typeDocument, numeroDocument} ciblé.

Lève une exception si :
- Le document n'existe pas,
- Le document est en cours d'utilisation.

Exemple :
```
IDocumentService service = new
DocumentService("http://localhost:12345/Webservices100/BIJOU/");
service.AppliquerFraisPort(TypeDocument.BonDeLivraisonVente,"BL000015",25m,
TypePrix.TTC);
```

Fonction :
```
void AppliquerExpeditionDefaut(TypeDocument typeDocument, string numeroDocument);
```
Description :
Applique les frais de port par défaut (associés au tiers du document) sur le document {typeDocument, numeroDocument} ciblé.

Lève une exception si :
- Le document n'existe pas,
- Le document est en cours d'utilisation.

Exemple :
```
IDocumentService service = new
DocumentService("http://localhost:12345/Webservices100/BIJOU/");
service.AppliquerExpeditionDefaut(TypeDocument.BonDeCommandeVente,"BC000012");
```

Fonction :

```
void AppliquerModeExpedition (TypeDocument typeDocument, string numeroDocument, short idModeExpedition);
```

Description :

Applique le mode d'expédition idModeExpedition sur le document {typeDocument, numeroDocument} ciblé.

Lève une exception si :

- Le document n'existe pas,
- Le document est en cours d'utilisation,
- Le document est une facture comptabilisée,
- Le domaine du document n'est pas Achat ou Vente,
- Le mode d'expédition n'existe pas.

Exemple :

```
IDocumentService service = new DocumentService("http://localhost:12345/Webservices100/BIJOU/");
service.AppliquerModeExpedition(TypeDocument.BonDeCommandeVente,"BC000012",2);
```

Fonction :

```
bool UpdateListChampsDocument(ListUpdateEnteteDoc updateInfoEnteteData);
```

Description :

Permet de mettre à jour une liste de documents, avec la possibilité de modifier simultanément plusieurs champs ainsi que plusieurs informations libres.
Renvoie vrai si la mise à jour est effectuée.
La méthode lève une exception si :

- Un des documents ciblés n'existe pas.
- Le nom d'une information libre est inconnu.
- La valeur d'une information libre n'est pas convertissable dans le type attendu.
- Le même champ est défini plusieurs fois pour le même document avec des valeurs différentes.
- Une même information libre est définie plusieurs fois pour le même document avec des valeurs différentes.
- Un numéro de document est manquant.

Exemple :

```
IDocumentService service = new
DocumentService("http://localhost:12345/Webservices100/BIJOU");
ListUpdateEnteteDoc listUpdateEnteteDoc = new ListUpdateEnteteDoc()
{
    Documents = new List<UpdateEnteteDocument>()
    {
        new UpdateEnteteDocument
        {
            NumeroDocument = "FA0007",
            TypeDocument = TypeDocument.FactureVente,
            ProprietesDocument = new List<UpdateEnteteData>
            {
                new UpdateEnteteData { Champ = ChampsDocumentUpdate.Escompte, Value =
5.00 },
                new UpdateEnteteData { Champ = ChampsDocumentUpdate.Reference, Value =
"Reference updated" },
            },
            InfosLibresList = new ListInfoLibre
            {
                new InfoLibre { Name = "Commentaires", Value = "Commentaires mis à jour"
},
                new InfoLibre { Name = "Date", Value = DateTime.Now.AddDays(5) }
            }
        },
        new UpdateEnteteDocument
        {
            NumeroDocument = "BC0005",
            TypeDocument = TypeDocument.BonDeCommandeVente,
            ProprietesDocument = new List<UpdateEnteteData>
            {
                new UpdateEnteteData { Champ = ChampsDocumentUpdate.Statut, Value =
StatutDocument.Saisie }
            }
        }
    },
};
service.UpdateListChampsDocument(listUpdateEnteteDoc);
```

g) FamilleService

Fonctions disponibles :

- `List<Famille> GetList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);`
- `List<Famille> GetByType(TypeFamille typeFamille, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);`
- `Famille GetFamille(string code);`

Fonction :

```
List<Famille> GetList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0)
```

Description :

Retourne la liste des familles correspondant aux critères passés en paramètre.

Exemple :

```
IFamilleService service = new FamilleService("http://localhost:12345/Webservices100/BIJOU/");
Criteria criteria = new CriteriaComparison("CodeFamille", ComparisonOperator.Like, "%" + codeFamilleLike + "%");
List<Famille> list = service.GetList(criteria, pageNumber : 2, rowsPerPage : 5) ;
```

Fonction :

```
List<Famille> GetByType(TypeFamille typeFamille, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
```

Description :

Retourne la liste des familles dont le type de famille correspond au paramètre typeFamille.

Exemple :

```
IFamilleService service = new FamilleService("http://localhost:12345/Webservices100/BIJOU/");
List<Famille> list = service.GetByType(TypeFamille.Detail);
```

Fonction :

```
Famille GetFamille(string code)
```

Description :

Retourne la famille par sa référence.
Renvoi null si le code de famille n'existe pas dans la base de données.
Lève une ArgumentException si le paramètre code est nul.

Exemple :

```
IFamilleService service = new FamilleService("http://localhost:12345/Webservices100/BIJOU/");
Famille famille = service.GetFamille("CBL");
```

h) SoucheService

Fonctions disponibles :

- `List<SoucheAchat> GetSoucheAchatList();`
- `SoucheAchat GetSoucheAchatByIndice(int id);`
- `SoucheAchat GetSoucheAchatByName(string nomSouche);`
- `List<SoucheVente> GetSoucheVenteList();`
- `SoucheVente GetSoucheVenteByIndice(int id);`
- `SoucheVente GetSoucheVenteByName(string nomSouche);`
- `List<SoucheInterne> GetSoucheInterneList();`
- `SoucheInterne GetSoucheInterneByIndice(int id);`
- `SoucheInterne GetSoucheInterneByName(string nomSouche);`

Fonction :
```
List<SoucheAchat> GetSoucheAchatList();
```

Description :
Retourne la liste des souches d'achat.

Exemple :
```
ISoucheService service = new
SoucheService("http://localhost:12345/Webservices100/BIJOU/");
List<SoucheAchat> souches = service.GetSoucheAchatList();
```

Fonction :
```
SoucheAchat GetSoucheAchatByIndice(int id);
```

Description :
Retourne la souche d'achat avec l'identifiant id passé en paramètre.

Exemple :
```
ISoucheService service = new
SoucheService("http://localhost:12345/Webservices100/BIJOU/");
SoucheAchat souche = service.GetSoucheAchatById(2);
```

Fonction :
```
SoucheAchat GetSoucheAchatByName(string nomSouche);
```

Description :
Retourne la souche d'achat ayant l'intitulé nomSouche passé en paramètre.

Exemple :
```
ISoucheService service = new
SoucheService("http://localhost:12345/Webservices100/BIJOU/");
SoucheAchat souche = service.GetSoucheAchatByName("N° Pièce");
```

Fonction :
```
List<SoucheVente> GetSoucheVenteList ();
```

Description :
Retourne la liste des souches de vente.

Exemple :
```
ISoucheService service = new
SoucheService("http://localhost:12345/Webservices100/BIJOU/");
List<SoucheVente> souches = service.GetSoucheVenteList();
```

Fonction :
```
SoucheVente GetSoucheVenteByIndice(int id);
```

Description :
Retourne la souche de vente avec l'identifiant id passé en paramètre.

Exemple :
```
ISoucheService service = new
SoucheService("http://localhost:12345/Webservices100/BIJOU/");
SoucheVente souche = service.GetSoucheVenteById(2);
```

Fonction :
```
SoucheVente GetSoucheVenteByName(string nomSouche);
```

Description :
Retourne la souche de vente ayant l'intitulé nomSouche passé en paramètre.

Exemple :
```
ISoucheService service = new
SoucheService("http://localhost:12345/Webservices100/BIJOU/");
SoucheVente souche = service.GetSoucheVenteByName("Refacturation");
```

Fonction :
```
List<SoucheInterne> GetSoucheInterneList();
```

Description :
Retourne la liste des souches internes.

Exemple :
```
ISoucheService service = new
SoucheService("http://localhost:12345/Webservices100/BIJOU/");
List<SoucheInterne> souches = service.GetSoucheInterneList();
```

Fonction :
```
SoucheInterne GetSoucheInterneByIndice(int id);
```

Description :
Retourne la souche interne avec l'identifiant id passé en paramètre.

Exemple :
```
ISoucheService service = new
SoucheService("http://localhost:12345/Webservices100/BIJOU/");
SoucheInterne souche = service.GetSoucheInterneByIndice(1);
```

Fonction :
```
SoucheInterne GetSoucheInterneByName(string nomSouche);
```

Description :
Retourne la souche interne ayant l'intitulé nomSouche passé en paramètre.

Exemple :
```
ISoucheService service = new
SoucheService("http://localhost:12345/Webservices100/BIJOU/");
SoucheInterne souche = service.GetSoucheInterneByName("N° Pièce");
```

i) StockService

Fonctions disponibles :

```
//Dépot
▪ Depot GetDepot(int idDepot);
▪ List<Depot> GetListDepot(Criteria criteria = null, List<Order> orders = null, int
  pageNumber = 0, int rowsPerPage = 0);


//Emplacement
▪ Emplacement GetEmplacement(int idDepot, int idEmplacement);
▪ Emplacement GetEmplacementParDefaut(string refArticle, int idDepot, int idGamme1 = 0,
  int idGamme2 = 0);
▪ List<Emplacement> GetListEmplacement(Criteria criteria = null, List<Order> orders =
  null, int pageNumber = 0, int rowsPerPage = 0);
▪ List<Emplacement> GetListEmplacementByDepot(int idDepot);


//StockDepot
▪ List<StockDepot> GetListStock(Criteria criteria = null, List<Order> orders = null, int
  pageNumber = 0, int rowsPerPage = 0);
▪ List<StockDepot> GetListStockGamme(Criteria criteria = null, List<Order> orders =
  null, int pageNumber = 0, int rowsPerPage = 0);
▪ StockDepot GetStockByRefArticle(string refArticle, int idDepot);
▪ StockDepot GetStockByRefArticleGamme(string refArticle, int idDepot, int idGamme1, int
  idGamme2);


//StockEmplacement
▪ List<StockEmplacement> GetListStockEmplacement(string refArticle, int idDepot, int
  idGamme1 = 0, int idGamme2 = 0);
▪ StockEmplacement GetStockEmplacement(string refArticle, int idDepot, int
  idEmplacement, int idGamme1 = 0, int idGamme2 = 0);


//LotSerie
▪ List<LotSerie> GetListLotSerie(Criteria criteria = null, List<Order> orders = null,
  int pageNumber = 0, int rowsPerPage = 0);
▪ List<LotSerie> GetListLotSerieByRefArticle(string refArticle);
▪ List<LotSerie> GetListLotSerieDisponible(string refArticle, int idDepot = 0);
▪ List<LotSerie> GetCombinaisonLotSerie(string refArticle,int idDepot, decimal
  quantite);
```

Fonction :
```
Depot GetDepot(int idDepot);
```

Description :
Renvoi un dépôt par son identifiant.

Exemple :
```
IStockService service = new
StockService("http://localhost:12345/Webservices100/BIJOU/");
Depot depot = service.GetDepot(1);
```

Fonction :
```
List<Depot> GetListDepot(Criteria criteria = null, List<Order> orders = null, int
pageNumber = 0, int rowsPerPage = 0);
```

Description :
Renvoi la liste des dépots correspondants au critère passé en paramètre.

Exemple :
```
IStockService service = new
StockService("http://localhost:12345/Webservices100/BIJOU/");
List<Depot> list = service.GetListDepot();
```

Fonction :
```
Emplacement GetEmplacement(int idDepot, int idEmplacement);
```

Description :
Renvoi un emplacement par son identifiant de dépôt et son identifiant.

Exemple :
```
IStockService service = new
StockService("http://localhost:12345/Webservices100/BIJOU/");
Emplacement emplacement = service.GetEmplacement(1, 69);
```

Fonction :
```
Emplacement GetEmplacementParDefaut(string refArticle, int idDepot, int idGamme1 = 0,
int idGamme2 = 0);
```

Description :
Renvoi l'emplacement par défaut de la référence d'article, ainsi que de ses gammes, dans le dépôt idDepot.
Si le paramètre refArticle est nul, une ArgumentException est levée.

Exemple :
```
IStockService service = new
StockService("http://localhost:12345/Webservices100/BIJOU/");
Emplacement emplacement = service.GetEmplacementParDefaut("CHAAR/VAR", 1, 6, 7);
```

Fonction :
```
List<Emplacement> GetListEmplacement(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
```

Description :
Renvoi la liste des emplacements correspondants au critère passé en paramètre.

Exemple :
```
IStockService service = new StockService("http://localhost:12345/Webservices100/BIJOU/");
List<Emplacement> list = service.GetListEmplacement();
```

Fonction :
```
List<Emplacement> GetListEmplacementByDepot(int idDepot);
```

Description :
Renvoi la liste des emplacements du dépôt idDepot.

Exemple :
```
IStockService service = new StockService("http://localhost:12345/Webservices100/BIJOU/");
List<Emplacement> list = service.GetListEmplacementByDepot(1);
```

Fonction :
```
List<StockDepot> GetListStock(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
```

Description :
Renvoi la liste des StockDepot correspondants au critère passé en paramètre.

Exemple :
```
IStockService service = new StockService("http://localhost:12345/Webservices100/BIJOU/");
List<StockDepot> list = service.GetListStock();
```

Fonction :
```
List<StockDepot> GetListStockGamme(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
```

Description :
Renvoi la liste des StockDepot d'article à gamme correspondants au critère passé en paramètre.

Exemple :
```
IStockService service = new StockService("http://localhost:12345/Webservices100/BIJOU/");
Criteria criteria = new CriteriaComparison("QteMini", ComparisonOperator.LessThan, qteMini);
List<StockDepot> list = service.GetListStockGamme(criteria);
```

Fonction :
```
StockDepot GetStockByRefArticle(string refArticle, int idDepot);
```

Description :
Renvoi le StockDepot de la référence article et du dépôt passés en paramètre.
Si le paramètre refArticle est nul, une ArgumentException est levée.

Exemple :
```
IStockService service = new
StockService("http://localhost:12345/Webservices100/BIJOU/");
StockDepot stockDepot = service.GetStockByRefArticle("BRAAR10", 2);
```

Fonction :
```
StockDepot GetStockByRefArticleGamme(string refArticle, int idDepot, int idGamme1, int
idGamme2);
```

Description :
Renvoi le StockDepot de la référence d'article à gamme et de ses gammes sur le dépôt idDepot.
Si le paramètre refArticle est nul, une ArgumentException est levée.

Exemple :
```
IStockService service = new
StockService("http://localhost:12345/Webservices100/BIJOU/");
StockDepot stockDepot = service.GetStockByRefArticleGamme("CHAAR/VAR", 2, 6, 7);
```

Fonction :
```
List<StockEmplacement> GetListStockEmplacement(string refArticle, int idDepot, int
idGamme1 = 0, int idGamme2 = 0);
```

Description :
Renvoi la liste des StockEmplacement de la référence article refArticle, ainsi que de ses gammes, sur le dépôt idDepot.
Si le paramètre refArticle est nul, une ArgumentException est levée.

Exemple :
```
IStockService service = new
StockService("http://localhost:12345/Webservices100/BIJOU/");
List<StockEmplacement> list = service.GetListStockEmplacement("CHAAR/VAR", 1, 6, 7);
```

Fonction :
```
StockEmplacement GetStockEmplacement(string refArticle, int idDepot, int idEmplacement,
int idGamme1 = 0, int idGamme2 = 0);
```

Description :
Renvoi le StockEmplacement de la référence d'article, ainsi que de ses gammes, sur l'emplacement idEmplacement du dépôt idDepot.
Si le paramètre refArticle est nul, une ArgumentException est levée.

Exemple :
```
IStockService service = new
StockService("http://localhost:12345/Webservices100/BIJOU/");
StockEmplacement stockEmplacement = service.GetStockEmplacement("BRAAR10", 1, 17);
```

Fonction :
```
List<LotSerie> GetListLotSerie(Criteria criteria = null, List<Order> orders = null, int
pageNumber = 0, int rowsPerPage = 0);
```

Description :
Renvoi la liste des LotSerie correspondants au critère passé en paramètre.

Exemple :
```
IStockService service = new
StockService("http://localhost:12345/Webservices100/BIJOU/");
List<LotSerie> list = service.GetListLotSerie();
```

Fonction :
```
List<LotSerie> GetListLotSerieByRefArticle(string refArticle);
```

Description :
Renvoi la liste des LotSerie de la référence d'article passée en paramètre.
Si le paramètre refArticle est nul, une ArgumentException est levée.

Exemple :
```
IStockService service = new
StockService("http://localhost:12345/Webservices100/BIJOU/");
List<LotSerie> list = service.GetListLotSerieByRefArticle("LINGOR18");
```

Fonction :

```
List<LotSerie> GetListLotSerieDisponible(string refArticle, int idDepot = 0);
```

Description :

Renvoi la liste des LotSerie disponible de la référence d'article sur le dépôt passé en paramètre.

Exemple :

```
IStockService service = new
StockService("http://localhost:12345/Webservices100/BIJOU/");
List<LotSerie> list = service.GetListLotSerieByRefArticle("LINGOR18",1);
```

Fonction :

- `List<LotSerie> GetCombinaisonLotSerie(string refArticle,int idDepot, decimal quantite);`

Description :

Renvoi une combinaison de LotSerie disponible pour la quantité de la référence d'article sur le dépôt passé en paramètre.

Exemple :

```
IStockService service = new
StockService("http://localhost:12345/Webservices100/BIJOU/");
List<LotSerie> list = service. GetCombinaisonLotSerie ("LINGOR18",1,7);
```

j) TarifService

Fonctions disponibles :

```
List<TarifArticle> GetListTarifArticle(string refArticle);
List<TarifArticle> GetListTarifArticles(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
List<TarifClient> GetListTarifClient(string numeroTiers);
List<TarifClient> GetListTarifClients(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
List<TarifFournisseur> GetListTarifFournisseurs(string numeroFournisseur, string refArticle);
List<TarifFournisseur> GetListTarifFournisseursCritere(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
TarifArticle GetTarifArticle(string refArticle, short idCategorieTarifaire);
TarifArticle GetTarifArticleGamme(string refArticle, short idCategorieTarifaire, int idGamme1, int idGamme2 = 0);
TarifArticle GetTarifArticleGammeIntitule(string refArticle, short idCategorieTarifaire, string intituleGamme1, string intituleGamme2 = "");
TarifClient GetTarifClient(string refArticle, string numeroTiers);
TarifClient GetTarifClientGamme(string refArticle, string numeroTiers, int idGamme1, int idGamme2 = 0);
TarifClient GetTarifClientGammeIntitule(string refArticle, string numeroTiers, string intituleGamme1, string intituleGamme2 = "");
TarifFournisseur GetTarifFournisseur(string numeroFournisseur, string refArticle, int idGamme1 = 0, int idGamme2 = 0);
bool InsertOrUpdateTarifArticle(TarifArticle tarif, Guid? guidCreateur = null);
bool InsertOrUpdateTarifClient(TarifClient tarif, Guid? guidCreateur = null);
bool DeleteTarifArticle(TarifArticle tarif);
bool DeleteTarifClient(TarifClient tarif);


//TarifRemise
TarifRemise GetTarifRemiseSurQuantiteByClientAndRefArticle(string numeroTiers, string refArticle, string quantite, string idGamme1 = "0", string idGamme2 = "0");
TarifRemise GetTarifRemiseSurMontantByClientAndRefArticle(string numeroTiers, string refArticle, string montant, string idGamme1 = "0", string idGamme2 = "0");
TarifRemise GetTarifRemiseSurQuantiteByCategorieTarifaireAndRefArticle(string idCategorieTarifaire, string refArticle, string quantite, string idGamme1 = "0", string idGamme2 = "0");
TarifRemise GetTarifRemiseSurMontantByCategorieTarifaireAndRefArticle(string idCategorieTarifaire, string refArticle, string montant, string idGamme1 = "0", string idGamme2 = "0");
TarifRemise GetTarifRemiseSurQuantiteByClientAndRefArticleWithIntitulesGammes(string numeroTiers, string refArticle, string quantite, string intituleGamme1, string intituleGamme2 = "");
TarifRemise GetTarifRemiseSurMontantByClientAndRefArticleWithIntitulesGammes(string numeroTiers, string refArticle, string montant, string intituleGamme1, string intituleGamme2 = "");
TarifRemise GetTarifRemiseSurQuantiteByCategorieTarifaireAndRefArticleWithIntitulesGammes(string idCategorieTarifaire, string refArticle, string quantite, string intituleGamme1, string intituleGamme2 = "");
```

```
- TarifRemise GetTarifRemiseSurMontantByCategorieTarifaireAndRefArticleWithIntituleGammes(string idCategorieTarifaire, string refArticle, string montant, string intituleGamme1, string intituleGamme2 = "");
- IList<TarifRemise> GetListTarifRemiseClient(string numeroTiers, string refArticle, int idGamme1 = 0, int idGamme2 = 0);
- IList<TarifRemise> GetListTarifRemiseCategorieTarifaire(short idCategorieTarifaire, string refArticle, int idGamme1 = 0, int idGamme2 = 0);
- IList<TarifRemise> GetListTarifRemiseClientWithIntitulesGammes(string numeroTiers, string refArticle, string intituleGamme1, string intituleGamme2 = "");
- IList<TarifRemise> GetListTarifRemiseCategorieTarifaireWithIntitulesGammes(short idCategorieTarifaire, string refArticle, string intituleGamme1, string intituleGamme2 = "");


//RemiseFamilleClient
- List<RemiseFamilleClient> GetListRemiseFamilleClientByNumeroClient(string numeroClient);
- List<RemiseFamilleClient> GetListRemiseFamilleClientByCodeFamille(string codeFamille);
- RemiseFamilleClient GetRemiseFamilleClient(string codeFamille, string numeroClient);
- List<RemiseFamilleClient> GetListRemiseFamilleClient(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
- List<RemiseFamilleClient> RemisesFamilleClient(int pageNumber = 0, int rowsPerPage = 0);
- RemiseFamilleClient InsertRemiseFamilleClient(RemiseFamilleClient remiseFamilleClient, Guid? guidCreateur = null);
- List<RemiseFamilleClient> InsertListRemiseFamilleClient(List<RemiseFamilleClient> remiseFamilleClientList, Guid? guidCreateur = null);
- RemiseFamilleClient UpdateRemiseFamilleClient(RemiseFamilleClient remiseFamilleClient);
- List<RemiseFamilleClient> UpdateListRemiseFamilleClient(List<RemiseFamilleClient> remiseFamilleClientList);
- bool DeletRemiseFamilleClient(string codeFamille, string numeroTiers);


- PrixUnitaireLigneArticle GetPrixUnitaireLigneArticle(PrixUnitaireLigneArticleParams parameters);
```

Fonction :
```
List<TarifArticle> GetListTarifArticle(string refArticle);
```
Si le paramètre refArticle est nul, une ArgumentException est levée.

Description :
Retourne la liste des tarifs articles de la référence d'article passée en paramètre.

Exemple :
```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
List<TarifArticle> list = service.GetListTarifArticle("BRAAR10");
```

Fonction :
```
List<TarifArticle> GetListTarifArticles(Criteria criteria = null, List<Order> orders =
null, int pageNumber = 0, int rowsPerPage = 0);
```

Description :
Retourne la liste des tarifs articles correspondants au critère passé en paramètre.

Exemple :
```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
Criteria criteria = new CriteriaComparison("RefArticle", ComparisonOperator.Like,
"%10%");
List<TarifArticle> list = service.GetListTarifArticle(criteria);
```

Fonction :
```
List<TarifClient> GetListTarifClient(string numeroTiers);
```

Description :
Retourne la liste des tarifs client associés au numéro du tiers passé en paramètre.
Si le paramètre numeroTiers est nul, une ArgumentException est levée.

Exemple :
```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
List<TarifClient> list = service.GetListTarifClient("CARAT");
```

Fonction :
```
List<TarifClient> GetListTarifClients(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
```

Description :
Retourne la liste des tarifs client correspondants au critère passé en paramètre.

Exemple :
```
ITarifService service = new TarifService("http://localhost:12345/Webservices100/BIJOU/");
Criteria criteria = new CriteriaComparison("Remise", ComparisonOperator.GreaterThanOrEqualTo, 10);
List<TarifClient> lList = service.GetListTarifClients(criteria);
```

Fonction :
```
List<TarifFournisseur> GetListTarifFournisseurs(string numeroFournisseur, string refArticle);
```

Description :
Retourne la liste des tarifs fournisseur du fournisseur et de l'article passés en paramètre.
Si le paramètre refArticle ou numeroFournisseur est nul, une ArgumentException est levée.

Exemple :
```
ITarifService service = new TarifService("http://localhost:12345/Webservices100/BIJOU/");
List<TarifFournisseur> list =service.GetListTarifFournisseurs("BILLO","BAOR01");
```

Fonction :
```
List<TarifFournisseur> GetListTarifFournisseursCritere(Criteria criteria = null,
List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
```

Description :
Retourne la liste des tarifs fournisseur correspondant au critère passé en paramètre.

Exemple :
```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
Criteria criteria = new CriteriaComparison("PrixAchat",
ComparisonOperator.LessThanOrEqualTo, 50);
List<TarifFournisseur> list =service.GetListTarifFournisseursCritere(criteria);
```

Fonction :
```
TarifArticle GetTarifArticle(string refArticle, short idCategorieTarifaire);
```

Description :
Retourne le tarif d'article de la référence d'article et de la catégorie tarifaire passées en paramètre.
Si le paramètre refArticle est nul, une ArgumentException est levée.

Exemple :
```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
TarifArticle tarif = service.GetTarifArticle("BRAAR10", 1);
```

Fonction :
```
TarifArticle GetTarifArticleGamme(string refArticle, short idCategorieTarifaire, int
idGamme1, int idGamme2 = 0);
```

Description :
Retourne le tarif article de la référence d'article, des gammes et de la catégorie tarifaire passées en paramètre.
Si le paramètre refArticle est nul, une ArgumentException est levée.

Exemple :
```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
TarifArticle tarif = service.GetTarifArticleGamme("CHAAR/VAR", 1,6,7);
```

Fonction :
```
TarifArticle GetTarifArticleGammeIntitule(string refArticle, short idCategorieTarifaire,
string intituleGamme1, string intituleGamme2 = "");
```

Description :
Retourne le tarif article de la référence d'article, des intitulés de gamme et de la catégorie tarifaire passés en paramètre.
Si le paramètre refArticle est nul, une ArgumentException est levée.

Exemple :
```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
TarifArticle tarif = service.GetTarifArticleGammeIntitule("CHAAR/VAR", 1, "54
cm","Classique");
```

Fonction :
```
TarifClient GetTarifClient(string refArticle, string numeroTiers);
```

Description :
Retourne le tarif client pour la référence article et le numéro client passé en paramètre.
Si le paramètre refArticle ou numeroTiers est nul, une ArgumentException est levée.

Exemple :
```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
TarifClient tarif = service.GetTarifClient("BAAR01", "CARAT");
```

Fonction :
```
TarifClient GetTarifClientGamme(string refArticle, string numeroTiers, int idGamme1, int
idGamme2 = 0);
```

Description :
Retourne le tarif client pour la référence article à gamme, des gammes et le numéro client passés en paramètre.

Exemple :
```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
TarifClient tarif = service.GetTarifClientGamme("CHAAR/VAR", "CISEL", 6, 7);
```

Fonction :

```
TarifClient GetTarifClientGammeIntitule(string refArticle, string numeroTiers, string intituleGamme1, string intituleGamme2 = "");
```

Description :

Retourne le tarif client pour la référence article à gamme, des intitulés de gamme et le numéro client passés en paramètre.
Si le paramètre refArticle ou numeroTiers est nul, une ArgumentException est levée.

Exemple :

```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
TarifClient tarif = service. GetTarifClientGammeIntitule ("CHAAR/VAR", "CISEL", "54 cm", "Classique");
```

Fonction :

```
TarifFournisseur GetTarifFournisseur(string numeroFournisseur, string refArticle, int idGamme1 = 0, int idGamme2 = 0);
```

Description :

Retourne le tarif fournisseur pour le numéro de fournisseur, la référence d'article et les gammes passés en paramètre.
Si le paramètre refArticle ou numeroFournisseur est nul, une ArgumentException est levée.

Exemple :

```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
TarifFournisseur tarif = service.GetTarifFournisseur("ECLAT","BAOR01", 3);
```

Fonction :

```
bool InsertOrUpdateTarifArticle(TarifArticle tarif, Guid? guidCreateur = null);
```

Description :

Insère, ou met à jour, un tarif d'article.
Si le paramètre tarif est nul, une ArgumentException est levée.
Retourne true si le tarif article a été inseré ou mis à jour.

Exemple :

```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
TarifArticle tarifArticle = new TarifArticle()
            {
                Coefficient = 2,
                IdGamme1 = 6,
                IdGamme2 = 7,
                IdCategorieTarifaire = 3,
                PrixVente = 123.45M,
                RefArticle = "CHAAR/VAR",
                TypePrix = TypePrix.TTC
            };
service.InsertOrUpdateTarifArticle(tarifArticle, utilisateurConnecte.Guid);
```

Fonction :

```
bool InsertOrUpdateTarifClient(TarifClient tarif, Guid? guidCreateur = null);
```

Description :

Insère, ou met à jour, un tarif client.
Si le paramètre tarif est nul, une ArgumentException est levée.
Retourne true si le tarif client a été inseré ou mis à jour.

Exemple :

```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
TarifClient tarifClient = new TarifClient()
            {
                Coefficient = 2,
                IdGamme1 = 6,
                IdGamme2 = 7,
                PrixVente = 123.45M,
                RefArticle = "CHAAR/VAR",
                TypePrix = TypePrix.TTC,
                NumeroTiers = "CARAT",
                Remise = 3,
                RefArticleClient = "TEST"
            };
service.InsertOrUpdateTarifClient(tarifClient, utilisateurConnecte.Guid);
```

Fonction :
```
bool DeleteTarifArticle(TarifArticle tarif);
```

Description :
Supprime le tarif d'article passé en paramètre.
Retourne true si le tarif d'article a été supprimé.

Exemple :
```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
TarifArticle tarif = service.GetTarifArticleGamme("CHAAR/VAR", 3, 6, 7);
bool deleted = service.DeleteTarifArticle(tarif);
```

Fonction :
```
bool DeleteTarifClient(TarifClient tarif);
```

Description :
Supprime le tarif client passé en paramètre.
Retourne true si le tarif client a été supprimé.

Exemple :
```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
TarifClient tarifClient = service.GetTarifClient("BRAAR10","CARAT");
bool deleted = service.DeleteTarifClient(tarifClient);
```

Fonction :
```
TarifRemise GetTarifRemiseSurQuantiteByClientAndRefArticle(string numeroTiers, string
refArticle, string quantite, string idGamme1 = "0", string idGamme2 = "0");
```

Description :
Renvoie le tarifRemise de l'article pour le tiers pour la quantité d'article passée en paramètre.

Exemple :
```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
TarifRemise tarifRemise = service. GetTarifRemiseSurQuantiteByClientAndRefArticle
("CARAT","BRAAR10",5);
```

Fonction :
```
TarifRemise GetTarifRemiseSurMontantByClientAndRefArticle(string numeroTiers, string refArticle, string montant, string idGamme1 = "0", string idGamme2 = "0");
```

Description :
Renvoie le tarifRemise de l'article pour le tiers pour le montant d'article passée en paramètre.

Exemple :
```
ITarifService service = new TarifService("http://localhost:12345/Webservices100/BIJOU/");
TarifRemise tarifRemise = service. GetTarifRemiseSurMontantByClientAndRefArticle ("CARAT","BRAAR10",500);
```

Fonction :
```
TarifRemise GetTarifRemiseSurQuantiteByCategorieTarifaireAndRefArticle(string idCategorieTarifaire, string refArticle, string quantite, string idGamme1 = "0", string idGamme2 = "0");
```

Description :
Renvoie le tarifRemise de l'article pour la catégorie tarifaire pour la quantité d'article passée en paramètre.

Exemple :
```
ITarifService service = new TarifService("http://localhost:12345/Webservices100/BIJOU/");
TarifRemise tarifRemise = service. GetTarifRemiseSurQuantiteByCategorieTarifaireAndRefArticle ("1","BRAAR10",20);
```

Fonction :
```
TarifRemise GetTarifRemiseSurMontantByCategorieTarifaireAndRefArticle(string idCategorieTarifaire, string refArticle, string montant, string idGamme1 = "0", string idGamme2 = "0");
```

Description :
Renvoie le tarifRemise de l'article pour la catégorie tarifaire pour le montant d'article passée en paramètre.

Exemple :
```
ITarifService service = new TarifService("http://localhost:12345/Webservices100/BIJOU/");
TarifRemise tarifRemise = service. GetTarifRemiseSurMontantByCategorieTarifaireAndRefArticle ("1","BRAAR10",600);
```

Fonction :
```
TarifRemise GetTarifRemiseSurQuantiteByClientAndRefArticleWithIntitulesGammes(string
numeroTiers, string refArticle, string quantite, string intituleGamme1, string
intituleGamme2 = "");
```

Description :
Renvoie le tarifRemise de l'article pour le client pour la quantité d'article passée en paramètre.

Exemple :
```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
TarifRemise tarifRemise = service.
GetTarifRemiseSurQuantiteByClientAndRefArticleWithIntitulesGammes ("CARAT","BAOR01",25,
"Rubis");
```

Fonction :
```
TarifRemise GetTarifRemiseSurMontantByClientAndRefArticleWithIntitulesGammes(string
numeroTiers, string refArticle, string montant, string intituleGamme1, string
intituleGamme2 = "");
```

Description :
Renvoie le tarifRemise de l'article pour le client pour le montant d'article passée en paramètre.

Exemple :
```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
TarifRemise tarifRemise = service.
GetTarifRemiseSurMontantByClientAndRefArticleWithIntitulesGammes ("CARAT","BAOR01",700,
"Rubis");
```

Fonction :
```
TarifRemise
GetTarifRemiseSurQuantiteByCategorieTarifaireAndRefArticleWithIntitulesGammes(string
idCategorieTarifaire, string refArticle, string quantite, string intituleGamme1, string
intituleGamme2 = "");
```

Description :
Renvoie le tarifRemise de l'article pour la catégorie tarifaire pour la quantité d'article passée en paramètre.

Exemple :
```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
TarifRemise tarifRemise = service.
GetTarifRemiseSurQuantiteByCategorieTarifaireAndRefArticleWithIntitulesGammes
("1","BAOR01",40, "Rubis");
```

Fonction :

```
TarifRemise
GetTarifRemiseSurMontantByCategorieTarifaireAndRefArticleWithIntituleGammes(string
idCategorieTarifaire, string refArticle, string montant, string intituleGamme1, string
intituleGamme2 = "");
```

Description :

Renvoie le tarifRemise de l'article pour la catégorie tarifaire pour le montant d'article passée en paramètre.

Exemple :

```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
TarifRemise tarifRemise = service.
GetTarifRemiseSurMontantByCategorieTarifaireAndRefArticleWithIntituleGammes
("1","BAOR01",400, "Rubis");
```

Fonction :

```
IList<TarifRemise> GetListTarifRemiseClient(string numeroTiers, string refArticle, int
idGamme1 = 0, int idGamme2 = 0);
```

Description :

Renvoie la liste des tarifRemise de l'article pour le client passée en paramètre.

Exemple :

```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
IList<TarifRemise> tarifRemiseList = service. GetListTarifRemiseClient ("CARAT","BAOR01"
4);
```

Fonction :

```
IList<TarifRemise> GetListTarifRemiseCategorieTarifaire(short idCategorieTarifaire,
string refArticle, int idGamme1 = 0, int idGamme2 = 0);
```

Description :

Renvoie la liste des tarifRemise de l'article pour la catégorie tarifaire passée en paramètre.

Exemple :

```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
IList<TarifRemise> tarifRemiseList = service. GetListTarifRemiseCategorieTarifaire
("1","BAOR01", 4);
```

Fonction :
```
IList<TarifRemise> GetListTarifRemiseClientWithIntitulesGammes(string numeroTiers,
string refArticle, string intituleGamme1, string intituleGamme2 = "");
```

Description :
Renvoie la liste des tarifRemise de l'article pour le client passée en paramètre.

Exemple :
```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
IList<TarifRemise> tarifRemiseList = service.
GetListTarifRemiseClientWithIntitulesGammes ("CARAT","BAOR01", "Rubis");
```

Fonction :
```
IList<TarifRemise> GetListTarifRemiseCategorieTarifaireWithIntitulesGammes(short
idCategorieTarifaire, string refArticle, string intituleGamme1, string intituleGamme2 =
"");
```

Description :
Renvoie la liste des tarifRemise de l'article pour la catégorie tarifaire passée en paramètre.

Exemple :
```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
IList<TarifRemise> tarifRemiseList = service.
GetListTarifRemiseCategorieTarifaireWithIntitulesGammes (1,"BAOR01", "Rubis");
```

Fonction :
```
List<RemiseFamilleClient> GetListRemiseFamilleClientByNumeroClient(string numeroClient);
```

Description :
Renvoie la liste des remiseFamilleClient pour le numéro client passé en paramètre.

Exemple :
```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
List<RemiseFamilleClient> datas =
service.GetListRemiseFamilleClientByNumeroClient("CARAT");
```

Fonction :
```
List<RemiseFamilleClient> GetListRemiseFamilleClientByCodeFamille(string codeFamille);
```

Description :
Renvoie la liste des remiseFamilleClient pour la famille passé en paramètre.

Exemple :
```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
List<RemiseFamilleClient> datas = service. GetListRemiseFamilleClientByCodeFamille
("BIJOUXOR");
```

Fonction :
```
RemiseFamilleClient GetRemiseFamilleClient(string codeFamille, string numeroClient);
```

Description :
Renvoie la remiseFamilleClient pour le client sur la famille passé en paramètre.

Exemple :
```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
RemiseFamilleClient remiseFamilleClient = service. GetRemiseFamilleClient ("BIJOUXOR",
"CARAT");
```

Fonction :
```
List<RemiseFamilleClient> GetListRemiseFamilleClient(Criteria criteria = null,
List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
```

Description :
Renvoie une liste de remiseFamilleClient correspondant aux critères.

Exemple :
```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
List<RemiseFamilleClient> remiseFamilleClientList =
service.GetListRemiseFamilleClient(new CriteriaComparison("CodeFamille",
ComparisonOperator. Like, "BIJOU%");
```

Fonction :

```
RemiseFamilleClient InsertRemiseFamilleClient(RemiseFamilleClient remiseFamilleClient,
Guid? guidCreateur = null);
```

Description :

Méthode d'insertion d'une remiseFamilleClient.

La méthode lève une exception si :

- La combinaison codeFamille – numéroTiers existe déjà.
- Le code famille est vide.
- La famille n'existe pas.
- La famille est en cours d'utilisation.
- Le numero client est vide.
- Le tiers correspondant au numéro tiers n'est pas un client.
- Le client n'existe pas.

Exemple :

```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
RemiseFamilleClient remiseFamilleClient = new RemiseFamilleClient();
remiseFamilleClient.CodeFamille = "BIJOUARG";
remiseFamilleClient.NumeroTiers = "CARAT";
remiseFamilleClient.Remise= 2.5M;
remiseFamilleClient = service. InsertRemiseFamilleClient(remiseFamilleClient);
```

Fonction :

```
List<RemiseFamilleClient> InsertListRemiseFamilleClient(List<RemiseFamilleClient>
remiseFamilleClientList, Guid? guidCreateur = null);
```

Description :

Méthode d'insertion d'une liste de remiseFamilleClient.
La méthode lève une exception si :

- Une combinaison codeFamille – numéroTiers existe déjà.
- Un des code famille est vide.
- Un des code famille n'existe pas.
- Un des code famille est en cours d'utilisation.
- Un des numero client est vide.
- Un des tiers associé au numéro tiers n'est pas un client.
- Un des client n'existe pas.

Exemple :

```
List<RemiseFamilleClient> list = new List<RemiseFamilleClient>();
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
RemiseFamilleClient remiseFamilleClient = new RemiseFamilleClient();
remiseFamilleClient.CodeFamille = "BIJOUARG";
remiseFamilleClient.NumeroTiers = "CARAT";
remiseFamilleClient.Remise= 2.5M;
list.Add(remiseFamilleClient);
remiseFamilleClient.CodeFamille = "BIJOUXOR";
remiseFamilleClient.NumeroTiers = "CARAT";
remiseFamilleClient.Remise= 3.2M;
list.Add(remiseFamilleClient);
list = service.InsertListRemiseFamilleClient(list, utilisateurConnecte.Guid);
```

Fonction :

```
RemiseFamilleClient UpdateRemiseFamilleClient(RemiseFamilleClient remiseFamilleClient);
```

Description :

Méthode de mise à jour de remiseFamilleClient.
La méthode lève une exception si :

- Le code famille est vide.
- La famille n'existe pas.
- La famille est en cours d'utilisation.
- Le numero client est vide.
- Le tiers associé au numéro client n'existe pas.
- Le tiers associé au numéro client n'est pas un client.
- autre chose que la remise est modifiée.

Exemple :

```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
RemiseFamilleClient remiseFamilleClient =
service.GetRemiseFamilleClient("BIJOUARG","CARAT");
remiseFamilleClient.Remise= 5.1M;
remiseFamilleClient = service.UpdateRemiseFamilleClient(remiseFamilleClient);
```

Fonction :

```
List<RemiseFamilleClient> UpdateListRemiseFamilleClient(List<RemiseFamilleClient>
remiseFamilleClientList);
```

Description :

Méthode de mise à jour d'une liste de remiseFamilleClient.
La méthode lève une exception si autre chose que la remise est modifiée.

Exemple :

```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
List<RemiseFamilleClient> list = service.
GetListRemiseFamilleClientByCodeFamille("BIJOUARG");
//Modification des remises
list = service. UpdateListRemiseFamilleClient (list);
```

Fonction :

```
bool DeletRemiseFamilleClient(string codeFamille, string numeroTiers);
```

Description :

Méthode de suppression d'une remiseFamilleClient.
La méthode lève une exception si

- La combinaison codeFamille numeroTiers n'existe pas.

Exemple :

```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
List<RemiseFamilleClient> list = service.
GetListRemiseFamilleClientByCodeFamille("BIJOUARG");
//Modification des remises
list = service. UpdateListRemiseFamilleClient (list);
```

Fonction :

```
PrixUnitaireLigneArticle GetPrixUnitaireLigneArticle(PrixUnitaireLigneArticleParams
parameters);
```

Description :

Renvoie le prix unitaire par défaut d'un article en fonction :

- De la référence de l'article,
- Des gammes (1 et 2) de l'article,
- Du conditionnement,
- Du type de document,
- De la quantité,
- De la devise,
- Du cours de la devise,
- Du Numéro tiers,
- De la catégorie tarifaire,
- Du dépôt.

Exemple :

```
ITarifService service = new
TarifService("http://localhost:12345/Webservices100/BIJOU/");
PrixUnitaireLigneArticleParams paramForPrixUnitaire = new
PrixUnitaireLigneArticleParams()
{
RerenceArticle = "BRAAR10",
TypeDocument = TypeDocument.BonDeCommandeVente,
Quantite = 3,
NumeroTiers = "CISEL",
IdDepot = 2
};
PrixUnitaireLigneArticle prixArticle =
service.GetPrixUnitaireLigneArticle(paramForPrixUnitaire) ;
```

k) TiersService

Fonctions disponibles :

- `Tiers GetTiers(string numeroTiers);`
- `List<Tiers> GetList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);`
- `List<Client> GetListClient(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);`
- `List<Fournisseur> GetListFournisseur(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);`
- `IList<Salarie> GetListSalarie(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);`
- `List<AutresTiers> GetListAutresTiers(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);`
- `int GetCount(Criteria criteria = null);`
- `Tiers Insert(Tiers tiers, Guid? guidCreateur = null);`
- `Tiers Update(Tiers tiers);`
- `bool Delete(string numeroTiers);`
- `void DefinirConditionReglement(string numeroTiers, List<ConditionReglement> conditionReglementList, Guid? guidCreateur);`
- `void AppliquerModeleReglement(string numeroTiers, int idModeleReglement, Guid? guidCreateur);`

```
//RoutageReceptionFactureElectronique
```

- `List<RoutageReceptionFactureElectronique> GetRoutageReceptionFactureElectroniqueList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);`
- `List<RoutageReceptionFactureElectronique> RoutageReceptionFactureElectroniqueListDuNumeroTiers(string numeroTiers);`
- `List<RoutageReceptionFactureElectronique> InsertRoutageReceptionFactureElectroniqueCollectionAuTiersNumero (RoutageReceptionFactureElectroniqueInsertToNumTiersCollectionData data);`
- `List<RoutageReceptionFactureElectronique> ReplaceRoutageReceptionFactureElectroniqueCollectionAuTiersNumero (RoutageReceptionFactureElectroniqueInsertToNumTiersCollectionData data);`
- `List<RoutageReceptionFactureElectronique> InsertRoutageReceptionFactureElectroniqueCollection (RoutageReceptionFactureElectroniqueInsertLightWithNumeroTiersCollectionData data);`
- `List<RoutageReceptionFactureElectronique> UpdateRoutageReceptionFactureElectroniqueCollection (RoutageReceptionFactureElectroniqueUpdateCollectionData data);`
- `TypeTiers GetTypeTiersByNumCompteGeneral(string numCompteGeneral);`
- `List<ParametresTiers> GetListParametresTiers();`

Fonction :
```
Tiers GetTiers(string numeroTiers);
```

Description :
Retourne un tiers par son numéro.
Lève une ArgumentException si le paramètre numeroTiers est nul

Exemple :
```
ITiersService service = new
TiersService("http://localhost:12345/Webservices100/BIJOU/");
Tiers tiers = service.GetTiers("BAGUES");
```

Fonction :
```
List<Tiers> GetList(Criteria criteria = null, List<Order> orders = null, int pageNumber
= 0, int rowsPerPage = 0);
```

Description :
Retourne la liste des tiers répondant aux critères passés en paramètre.

Exemple :
```
ITiersService service = new
TiersService("http://localhost:12345/Webservices100/BIJOU/");
List<Tiers> list = service.GetList();
```

Fonction :
```
List<Tiers> GetListClient(Criteria criteria = null, List<Order> orders = null, int
pageNumber = 0, int rowsPerPage = 0);
```

Description :
Retourne la liste des clients répondant aux critères passés en paramètre.

Exemple :
```
ITiersService service = new
TiersService("http://localhost:12345/Webservices100/BIJOU/");
List<Tiers> list = service.GetListClient();
```

Fonction :
```
List<Tiers> GetListFournisseur(Criteria criteria = null, List<Order> orders = null, int
pageNumber = 0, int rowsPerPage = 0);
```

Description :
Retourne la liste des fournisseurs répondant aux critères passés en paramètre.

Exemple :
```
ITiersService service = new
TiersService("http://localhost:12345/Webservices100/BIJOU/");
List<Tiers> list = service.GetListFournisseur();
```

Fonction :
```
List<Tiers> GetListSalarie(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
```

Description :
Retourne la liste des salariés répondant aux critères passés en paramètre.

Exemple :
```
ITiersService service = new TiersService("http://localhost:12345/Webservices100/BIJOU/");
List<Tiers> list = service.GetListSalarie();
```

Fonction :
```
List<AutresTiers> GetListAutresTiers(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
```

Description :
Retourne la liste des tiers de type Autres répondant aux critères passés en paramètre.

Exemple :
```
ITiersService service = new TiersService("http://localhost:12345/Webservices100/BIJOU/");
List<AutresTiers> list = service.GetListAutresTiers();
```

Fonction :
```
int GetCount(Criteria criteria = null);
```

Description :
Retourne le nombre de tiers répondant aux critères passés en paramètre.

Exemple :
```
ITiersService service = new TiersService("http://localhost:12345/Webservices100/BIJOU/");
Criteria criteria = new CriteriaComparison("TypeTiers", ComparisonOperator.Equals, TypeTiers.Client);

int nb = service.GetCount(criteria);
```

Fonction :

```
Tiers Insert(Tiers tiers, Guid? guidCreateur = null);
```

Description :

Ajoute un nouveau contact en respectant les règles suivantes :

- Le champ 'Intitule' doit être renseigné.
- Si le tiers est un salarié, le champ 'CompteCollectif' doit être renseigné et exister.
- Le tiers payeur renseigné doit exister.
- La catégorie tarifaire renseignée doit exister.
- La catégorie comptable renseignée doit exister.
- Le mode d'expédition renseigné doit exister.
- La condition de livraison renseignée doit exister.
- La modèle de règlement renseigné doit exister.
- Si TypeNumerotationTiers = Manuelle
  - Le champ 'NumeroTiers' doit être renseigné.

  Si TypeNumerotationTiers = Racine
  - La racine du champ 'NumeroTiers' doit être respectée.
  - La longueur du champ 'NumeroTiers' doit être respectée.
- Le tiers ne doit pas déjà exister.

Champs initialisés si non renseignés :

- Si TypeNumerotationTiers = Automatique
  - NumeroTiers
- CompteCollectif
- Abrege
- NumeroTiersPayeur

Retourne le tiers inseré
Lève une ArgumentException si le tiers est nul

Exemple :

```
Client tiers = new Client();
tiers.NumeroTiers = "PROTEST";
tiers.Intitule = "TestSocieté";
tiers.Adresse = "71 rue de la bourse";
tiers.CodePostal = "L-1010";
tiers.Commentaire = "";
tiers.Complement = "4eme étage";
tiers.Contact = "Dal Cengio Eric";
tiers.Email = "info@test.lu";
tiers.Fax = "+352 17 29 21 40";
tiers.NumTva = "LU123456789";
tiers.Pays = "Luxembourg";
tiers.Qualite = "Société";
tiers.Region = "Luxembourg";
tiers.SiteWeb = "http://www.test.lu/";
tiers.Telephone = "+352 17 29 21 69";
tiers.Ville = "Lange";

ITiersService service = new
TiersService("http://localhost:12345/Webservices100/BIJOU/");
Tiers tiersInserted = service.Insert(tiers, utilisateurConnecte.Guid);
```

Fonction :

```
Tiers Update(Tiers tiers);
```

Description :

Met à jour un tiers existant en respectant les règles suivantes :

- Le champ 'Intitule' doit être renseigné.
- Si le tiers est un salarié, le champ 'CompteCollectif' doit être renseigné et exister.
- Le tiers payeur renseigné doit exister.
- La catégorie tarifaire renseignée doit exister.
- La catégorie comptable renseignée doit exister.
- Le mode d'expédition renseigné doit exister.
- La condition de livraison renseignée doit exister.
- La modèle de règlement renseigné doit exister.
- Si TypeNumerotationTiers = Manuelle
    - Le champ 'NumeroTiers' doit être renseigné.

  Si TypeNumerotationTiers = Racine
    - La racine du champ 'NumeroTiers' doit être respectée.
    - La longueur du champ 'NumeroTiers' doit être respectée.
- Le tiers doit exister.

Champs initialisés si non renseignés :

- Si TypeNumerotationTiers = Automatique
    - NumeroTiers
- CompteCollectif
- Abrege
- NumeroTiersPayeur

Retourne le tiers mise à jour
Lève une ArgumentException si le tiers est nul

Exemple :

```
ITiersService service = new
TiersService("http://localhost:12345/Webservices100/BIJOU/");
Tiers tiers = service.GetTiers("BAGUES");
tiers.Intitule = "New Name";
tiers.Adresse = "1, rue de la nouvelle adresse";
Tiers tiersUpdated = service.Update(tiers);
```

Fonction :
```
bool Delete(string id);
```

Description :
Supprime un tiers existant en respectant les règles suivantes :

- Aucuns abonnements ne doivent être liés au tiers.
- Aucuns documents ne doivent être liés au tiers.
- Aucunes banques ne doivent être liées au tiers.
- Aucuns tarifs ne doivent être liés au tiers.
- Aucuns règlements ne doivent être liés au tiers.
- Aucunes écritures ne doivent être liées au tiers.
- Aucuns registres de taxe ne doivent être liés au tiers.
- Aucuns dossiers de recouvrement ne doivent être liés au tiers.

Données liées supprimées :

- Comptes généraux du tiers
- Adresses de livraison du tiers

Reourne true si le tiers a été supprimé

Exemple :
```
ITiersService service = new
TiersService("http://localhost:12345/Webservices100/BIJOU/");
bool deleted = service.Delete("BAGUES");
```

Fonction :
```
void DefinirConditionReglement(string numeroTiers, List<ConditionReglement>
conditionReglementList, Guid? guidCreateur);
```

Description :
Définit les conditions de règlement du tiers.

Exemple :
```
ITiersService service = new
TiersService("http://localhost:12345/Webservices100/BIJOU/");
List<ConditionReglement> conditionsReglement = new List<ConditionReglement>();
//Définir les conditions de règlement à ajouter dans la liste {conditionReglement}.
service.DefinirConditionReglement("BAGUES", conditionsReglement,
utilisateurConnecte.Guid);
```

Fonction :
```
void AppliquerModeleReglement(string numeroTiers, int idModeleReglement, Guid?
guidCreateur);
```

Description :
Applique un modèle de règlement existant sur un tiers.

Exemple :
```
ITiersService service = new
TiersService("http://localhost:12345/Webservices100/BIJOU/");
service.AppliquerModeleReglement("BAGUES", 1, utilisateurConnecte.Guid);
```

Fonction :
```
List<RoutageReceptionFactureElectronique>
GetRoutageReceptionFactureElectroniqueList(Criteria criteria = null, List<Order> orders
= null, int pageNumber = 0, int rowsPerPage = 0);
```

Description :
Renvoie la liste des RoutageReceptionFactureElectronique répondant au critère passé en paramètre.

Exemple :
```
ITiersService service = new
TiersService("http://localhost:12345/Webservices100/BIJOU/");
Criteria criteria = new ComparisonCriteria("CodeRoutage", ComparisonOperator.Like,
"Rout%");
List<RoutageReceptionFactureElectronique> routages =
service. GetRoutageReceptionFactureElectroniqueList (criteria);
```

Fonction :
```
List<RoutageReceptionFactureElectronique>
RoutageReceptionFactureElectroniqueListDuNumeroTiers(string numeroTiers);
```

Description :
Renvoie la liste des RoutageReceptionFactureElectronique associés au tiers passé en paramètre.

Exemple :
```
ITiersService service = new
TiersService("http://localhost:12345/Webservices100/BIJOU/");
List<RoutageReceptionFactureElectronique> routages =
service.RoutageReceptionFactureElectroniqueListDuNumeroTiers("BAGUES");
```

Fonction :

```
List<RoutageReceptionFactureElectronique>
InsertRoutageReceptionFactureElectroniqueCollectionAuTiersNumero
(RoutageReceptionFactureElectroniqueInsertToNumTiersCollectionData data);
```

Description :

Méthode d'insertion d'une collection de RoutageReceptionFactureElectronique à un tiers.
Une exception est levée si :

- Le paramètre data est null,
- Le tiers ciblé par le NumeroTiers n'existe pas,
- Le tiers ciblé par le NumeroTiers n'est pas un Client.

Une exception est levée si un ou plusieurs éléments de la collection RoutageReceptionFactureElectroniqueCollection :

- A sa propriété CodeRoutage à null ou vide,
- A sa propriété Type qui ne pointe pas vers une valeur de TypeCodeRoutage autorisées dans la version de Sage ciblée
- A la longueur de sa propriété CodeRoutage supérieure à 100 caractères.

Exemple :

```
ITiersService service = new
TiersService("http://localhost:12345/Webservices100/BIJOU/");
List<RoutageReceptionFactureElectroniqueInsertDataLight> list = new
List<RoutageReceptionFactureElectroniqueInsertDataLight>()
{
    new RoutageReceptionFactureElectroniqueInsertDataLight()
    {
       Type = TypeCodeRoutage.CodeService,
       CodeRoutage = "CR1",
       Intitule = "Code route 1",
       EstActif = true
    },
    new RoutageReceptionFactureElectroniqueInsertDataLight()
    {
       Type = TypeCodeRoutage.SWIFT,
       CodeRoutage = "CR4",
       EstActif = true
    }
};
RoutageReceptionFactureElectroniqueInsertToNumTiersCollectionData data = new
RoutageReceptionFactureElectroniqueInsertToNumTiersCollectionData ()
{
    NumeroTiers = "BAGUES",
    GuidCreateur = utilisateurConnecte.Guid,
    RoutageReceptionFactureElectroniqueCollection = list,
};
service.InsertRoutageReceptionFactureElectroniqueCollectionAuTiersNumero(data);
```

Fonction :

```
List<RoutageReceptionFactureElectronique>
ReplaceRoutageReceptionFactureElectroniqueCollectionAuTiersNumero
(RoutageReceptionFactureElectroniqueInsertToNumTiersCollectionData data);
```

Description :

Méthode de remplacement des routages de réception de facture électronique définis sur un tiers par une nouvelle collection de RoutageReceptionFactureElectronique.
Une exception est levée si :

- Le paramètre data est null,
- Le tiers ciblé par le NumeroTiers n'existe pas,
- Le tiers ciblé par le NumeroTiers n'est pas un Client,

Une exception est levée si un ou plusieurs éléments de la collection RoutageReceptionFactureElectroniqueCollection :

- A sa propriété CodeRoutage à null ou vide,
- A sa propriété Type qui ne pointe pas vers une valeur de TypeCodeRoutage autorisées dans la version de Sage ciblée
- A la longueur de sa propriété CodeRoutage supérieure à 100 caractères.

Exemple :

```
ITiersService service = new
TiersService("http://localhost:12345/Webservices100/BIJOU/");
List<RoutageReceptionFactureElectroniqueInsertDataLight> list = new
List<RoutageReceptionFactureElectroniqueInsertDataLight>()
{
    new RoutageReceptionFactureElectroniqueInsertDataLight()
    {
       Type = TypeCodeRoutage.CodeService,
       CodeRoutage = "CR1",
       Intitule = "Code route 1",
       EstActif = true
    },
    new RoutageReceptionFactureElectroniqueInsertDataLight()
    {
       Type = TypeCodeRoutage.SWIFT,
       CodeRoutage = "CR4",
       EstActif = true
    }
};
RoutageReceptionFactureElectroniqueInsertToNumTiersCollectionData data = new
RoutageReceptionFactureElectroniqueInsertToNumTiersCollectionData ()
{
    NumeroTiers = "BAGUES",
    GuidCreateur = utilisateurConnecte.Guid,
    RoutageReceptionFactureElectroniqueCollection = list,
};
service.ReplaceRoutageReceptionFactureElectroniqueCollectionAuTiersNumero(data);
```

Fonction :

```
List<RoutageReceptionFactureElectronique>
InsertRoutageReceptionFactureElectroniqueCollection
(RoutageReceptionFactureElectroniqueInsertLightWithNumeroTiersCollectionData data);
```

Description :

Méthode d'insertion d'une collection de RoutageReceptionFactureElectronique.

Une exception est levée si le paramètre data est null.
Une exception est levée si un ou plusieurs éléments de la collection RoutageReceptionFactureElectroniqueCollection :

- A sa propriété CodeRoutage à null ou vide,
- A la longueur de sa propriété CodeRoutage supérieure à 100 caractères.
- Pointe vers un tiers inexistant,
- A sa propriété Type qui ne pointe pas vers une valeur de TypeCodeRoutage autorisées dans la version de Sage ciblée
- Pointe vers un tiers qui n'est pas de type Client.

Exemple :

```
ITiersService service = new
TiersService("http://localhost:12345/Webservices100/BIJOU/");
List<RoutageReceptionFactureElectroniqueInsertDataWithNumTiers> list = new
List<RoutageReceptionFactureElectroniqueInsertDataWithNumTiers>()
{
    new RoutageReceptionFactureElectroniqueInsertDataWithNumTiers()
    {
       NumeroTiers = "BAGUES",
       Type = TypeCodeRoutage.CodeService,
       CodeRoutage = "CR1",
       Intitule = "Code route 1",
       EstActif = true
    },
    new RoutageReceptionFactureElectroniqueInsertDataWithNumTiers()
    {
       NumeroTiers = "CERAM",
       Type = TypeCodeRoutage.SWIFT,
       CodeRoutage = "CR4",
       EstActif = true
    }
};
RoutageReceptionFactureElectroniqueInsertLightWithNumeroTiersCollectionData data = new
RoutageReceptionFactureElectroniqueInsertLightWithNumeroTiersCollectionData()
{
    GuidCreateur = utilisateurConnecte.Guid,
    RoutageReceptionFactureElectroniqueCollection = list,
};
service.InsertRoutageReceptionFactureElectroniqueCollection(data);
```

Fonction :

```
List<RoutageReceptionFactureElectronique>
UpdateRoutageReceptionFactureElectroniqueCollection
(RoutageReceptionFactureElectroniqueUpdateCollectionData data);
```

Description :

Méthode de mise à jour d'une collection de RoutageReceptionFactureElectronique.

Une exception est levée si le paramètre data est null.
Une exception est levée si un ou plusieurs éléments de la collection RoutageReceptionFactureElectroniqueCollection :

- A sa propriété CodeRoutage à null ou vide,
- A la longueur de sa propriété CodeRoutage supérieure à 100 caractères.
- A sa propriété IdentifiantUnique qui ne pointe pas sur un RoutageReceptionFactureElectronique existant,
- A sa propriété Type qui ne pointe pas vers une valeur de TypeCodeRoutage autorisées dans la version de Sage ciblée,
- Le numéro Tiers est modifié.

Exemple :

```
ITiersService service = new
TiersService("http://localhost:12345/Webservices100/BIJOU/");
List<RoutageReceptionFactureElectronique> routages = GetRoutagesPerso(……);
Foreach(RoutageReceptionFactureElectronique item in routages)
{
//Modification des valeurs de l'item.
}
RoutageReceptionFactureElectroniqueUpdateCollectionData data = new
RoutageReceptionFactureElectroniqueUpdateCollectionData()
{
     RoutageReceptionFactureElectroniqueCollection = routages,
};
service.UpdateRoutageReceptionFactureElectroniqueCollection(data);
```

Fonction :

- `TypeTiers GetTypeTiersByNumCompteGeneral(string numCompteGeneral);`

Description :

Retourne le type de tiers correspondant au numéro de compte général passé en paramètre.

Une exception est levée si :

- Le paramètre numCompteGeneral est nul.
- Il n'existe pas de nature de compte général correspondant au numéro de compte passé en paramètre.
- La nature du compte général n'est pas de type tiers.

(Fichier → Paramètres société → Comptes généraux → Nature de compte).

Exemple :

```
ITiersService service = new
TiersService("http://localhost:12345/Webservices100/BIJOU/");
TypeTiers typeTiers = service.GetTypeTiersByNumCompteGeneral("4010000");
```

Fonction :
- `List<ParametresTiers> GetListParametresTiers();`

Description :
Retourne la liste des configurations des codifications des comptes tiers (Paramètres société → Tiers → Codification).

Exemple :
```
ITiersService service = new
TiersService("http://localhost:12345/Webservices100/BIJOU/");
List<ParametresTiers> result = service.GetListParametresTiers();
```

5. Les services de données de comptabilité

a) BanqueService

Fonctions disponibles :

- `List<Banque> GetList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);`
- `Banque GetByAbrege(string abrege);`
- `Banque GetById(int banqueId);`
- `Banque Insert(Banque banque, bool useSageProcess = true, Guid? guidCreateur = null);`
- `Banque Update(Banque banque);`
- `bool DeleteById(int idBanque);`
- `bool DeleteByAbrege(string abregeBanque);`

Fonction :
```
Ecriture GetList (Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0) ;
```

Description :
Retourne une liste de banques.

Exemple :
```
IBanqueService service = new
EBanqueService("http://localhost:12345/Webservices100/BIJOU/") ;
List<Banque> banques = service. GetList ();
```

Fonction :
```
Banque GetByAbrege (string abrege);
```

Description :
Retourne la banque correspondant à l'abregé passer en paramètre.

Exemple :
```
IBanqueService service = new
EBanqueService("http://localhost:12345/Webservices100/BIJOU/") ;
List<Banque> banques =service.GetByAbrege("BEU");
```

Fonction :

- `Banque GetById(int banqueId);`

Description :

Retourne la banque correspondant à l'identifiant passé en paramètre.

Exemple :

```
IBanqueService service = new IBanqueService
("http://localhost:12345/Webservices100/BIJOU/") ;
Banque data = service.GetById(1);
```

Fonction :
- `Banque Insert(Banque banque, bool useSageProcess = true, Guid? guidCreateur = null);`

Description :
Ajoute une banque
Retourne true si la banque a été ajoutée

Exemple :
```
IBanqueService service = new BanqueService
("http://localhost:12345/Webservices100/BIJOU/");
Banque banque = new Banque() ;
//Initialisation de la banque avec toutes les données nécessaires à la conception d'une
banque
service.Insert(banque, guidCreateur : utilisateurConnecte.Guid);
```

Fonction :
- `Banque Update(Banque banque);`

Description :
Met à jour une banque.
Retourne true si la banque a été mise à jour

Exemple :
```
IBanqueService service = new BanqueService
("http://localhost:12345/Webservices100/BIJOU/");
Banque banque = service.GetById(1) ;
//Modification des propriétés de la banque
service.Update(banque);
```

Fonction :
```
bool DeleteById(int idBanque);
```

Description :
Supprime une banque dont l'identifiant est passé en paramètre
Retourne true si la la banque a été supprimée
Si le paramètre est égal à 0, une ArgumentException est levée.

Exemple :
```
IBanqueService service = new BanqueService
("http://localhost:12345/Webservices100/BIJOU/");
Banque banque = service.DeleteById(1) ;
```

Fonction :

```
bool DeleteByAbrege(string abregeBanque);
```

Description :

Supprime une banque dont l'abregé est passé en paramètre.
Retourne true si la banque a été supprimée.

Exemple :

```
IBanqueService service = new BanqueService
("http://localhost:12345/Webservices100/BIJOU/");
Banque banque = service.DeleteByAbrege("BEU") ;
```

b) BanqueTiersService

Fonctions disponibles :

- `List<BanqueTiers> GetList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);`
- `List<BanqueTiers> GetListBanqueTiers(string numeroTiers);`
- `BanqueTiers GetById(int id);`
- `BanqueTiers Insert(BanqueTiers banqueTiers, Guid? guidCreateur = null);`
- `BanqueTiers Update(BanqueTiers banqueTiers);`
- `bool Delete(int id, bool setFirstBanqueAsPrincipaleSiPrincipaleSupprimee = false, Guid? guidUtilisateur = null)`

Fonction :

```
List<BanqueTiers> GetList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
```

Description :

Retourne une liste des informations bancaires des tiers.

Exemple :

```
IBanqueTiersService service = new BanqueTiersService("http://localhost:12345/Webservices100/BIJOU/") ;
List<BanqueTiers> banquesTiers = service.GetList();
```

Fonction :

```
List<BanqueTiers> GetListBanqueTiers (Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
```

Description :

Retourne une liste des informations bancaires d'un tiers dont le numéro est passé en paramètre.

Exemple :

```
IBanqueTiersService service = new BanqueTiersService("http://localhost:12345/Webservices100/BIJOU/") ;
List<BanqueTiers> banquesTiers = service.GetListBanqueTiers("BAGUES");
```

Fonction :

```
BanqueTiers GetById(int id);
```

Description :

Retourne la BanqueTiers ayant comme identifiant le paramètre id.

Exemple :

```
IBanqueTiersService service = new BanqueTiersService("http://localhost:12345/Webservices100/BIJOU/") ;
BanqueTiers banqueTiers = service.GetById(16);
```

Fonction :
```
BanqueTiers Insert(BanqueTiers banqueTiers, Guid? guidCreateur = null);
```

Description :
Ajoute une banque pour un tiers.
Retourne une nouvelle instance de BanqueTiers correspondant au paramètre banqueTiers passé en paramètre après insertion.

Exemple :
```
IBanqueTiersService service = new
BanqueTiersService("http://localhost:12345/Webservices100/BIJOU/") ;
BanqueTiers banqueTiers = new BanqueTiers() ;
//Initialisation de la banque avec toutes les données nécessaires à la conception d'une
banque pour un tiers
BanqueTiers inserted = service.Insert(banqueTiers, utilisateurConnecte.Guid);
```

Fonction :
```
BanqueTiers Update(BanqueTiers banqueTiers, Guid? guidUtilisateur = null);
```

Description :
Met à jour une banque pour un tiers.
Retourne une nouvelle instance de BanqueTiers correspondant à la banque tiers passée en paramètre après mise à jour.

Exemple :
```
IBanqueTiersService service = new
BanqueTiersService("http://localhost:12345/Webservices100/BIJOU/") ;
BanqueTiers banqueTiers = service. GetListBanqueTiers("BAGUES")[0];
//Modification des propriétés de la banque tiers
Guid? guidUtilisateur = null;//Ou obtenu via ConnectUser
BanqueTiers updated = service.Update(banqueTiers, guidUtilisateur);
```

Fonction :
```
bool Delete(int id, bool setFirstBanqueAsPrincipaleSiPrincipaleSupprimee = false, Guid?
guidUtilisateur = null)
```

Description :
Supprime la banque tiers ayant l'identifiant passé en paramètre.

Si le booléen setFirstBanqueAsPrincipalSiPrincipaleSupprimee est défini à true

- et que le tiers possède plusieurs banques
- et que la banque supprimée est la banque principale

alors, la première banque non principale deviendra la banque principale du tiers.

Exemple :
```
IBanqueTiersService service = new
BanqueTiersService("http://localhost:12345/Webservices100/BIJOU/") ;
Guid? guidUtilisateur = null;//Ou obtenu via ConnectUser
bool isDeleted = service.Delete(1, guidUtilisateur);
```

c) CompteAnalytiqueService

Fonctions disponibles :

```
▪ CompteAnalytique GetCompteAnalytique(short IdPlanAnalytique, string compte);
▪ List<CompteAnalytique> GetList(Criteria criteria = null, List<Order> orders = null,
  int pageNumber = 0, int rowsPerPage = 0);
▪ CompteAnalytique Insert(CompteAnalytique compteAnalytique, Guid? guidCreateur = null);
▪ CompteAnalytique Update(CompteAnalytique compteAnalytique);
```

Fonction :

```
CompteAnalytique GetCompteAnalytique(short IdPlanAnalytique, string compte);
```

Description :

Retourne un compte analytique par son numéro et son plan.
Si le paramètre compte est nul, une ArgumentException est levée.

Exemple :

```
ICompteAnalytiqueService service = new
CompteAnalytiqueService("http://localhost:12345/Webservices100/BIJOU/");
CompteAnalytique compte = service.GetCompteAnalytique(1, "921SI1");
```

Fonction :

```
List<CompteAnalytique> GetList(Criteria criteria = null, List<Order> orders = null, int
pageNumber = 0, int rowsPerPage = 0);
```

Description :

Retourne la liste des comptes analytiques répondant aux critères passés en paramètre.

Exemple :

```
ICompteAnalytiqueService service = new
CompteAnalytiqueService("http://localhost:12345/Webservices100/BIJOU/");
List<CompteAnalytique> list = service.GetList();
```

Fonction :

```
CompteAnalytique Insert(CompteAnalytique compteAnalytique, Guid? guidCreateur = null);
```

Description :

Insère un compte analytique.

Exemple :

```
ICompteAnalytiqueService service = new
CompteAnalytiqueService("http://localhost:12345/Webservices100/BIJOU/");
CompteAnalytique compteA = new CompteAnalytique();
compteA.IdPlanAnalytique = 1;
compteA.Section = "Test";
compteA.Intitule ="Compte analytique Test";
service.Insert(compteA, utilisateurConnecte.Guid);
```

Fonction :

```
CompteAnalytique Update(CompteAnalytique compteAnalytique);
```

Description :

Met à jour un compte analytique.

Exemple :

```
ICompteAnalytiqueService service = new
CompteAnalytiqueService("http://localhost:12345/Webservices100/BIJOU/");
CompteAnalytique compte = service.GetCompteAnalytique(1, "921SI1");
compteA.Intitule ="Compte analytique Test";
service.Update(compteA);
```

d) CompteBancaireService

Fonctions disponibles :

- `List<CompteBancaire> GetList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);`
- `List<CompteBancaire> GetListByBanque(int idBanque);`
- `CompteBancaire GetById(int idCompteBancaire);`
- `CompteBancaire GetByAbrege(string abrege);`
- `CompteBancaire Insert(CompteBancaire compteBancaire, Guid? guidCreateur = null);`
- `CompteBancaire Update(CompteBancaire compteBancaire, Guid? guidUtilisateur = null);`
- `bool Delete(int idCompteBancaire, Guid? guidUtilisateur = null);`

Fonction :

```
List<CompteBancaire> GetList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
```

Description :

Retourne une liste des comptes bancaires.

Exemple :

```
ICompteBancaireService service = new CompteBancaireService
("http://localhost:12345/Webservices100/BIJOU/") ;
List<CompteBancaire> compteBancaires = service. GetList();
```

Fonction :

```
List<CompteBancaire> GetListByBanque(int idBanque);
```

Description :

Retourne une liste des comptes bancaires d'une banque dont l'identifiant est passé en paramètre.

Exemple :

```
ICompteBancaireService service = new CompteBancaireService
("http://localhost:12345/Webservices100/BIJOU/") ;
List<CompteBancaire> compteBancaires = service.GetListByBanque(1);
```

Fonction :

```
CompteBancaire GetById(int idCompteBancaire);
```

Description :

Retourne comptebancaire dont l'identifiant est passé en paramètre.

Exemple :

```
ICompteBancaireService service = new CompteBancaireService
("http://localhost:12345/Webservices100/BIJOU/") ;
CompteBancaire compteBancaire = service.GetById(1);
```

Fonction :
```
CompteBancaire GetByAbrege(string abrege);
```

Description :
Retourne comptebancaire dont l'abregé est passé en paramètre.

Exemple :
```
ICompteBancaireService service = new CompteBancaireService
("http://localhost:12345/Webservices100/BIJOU/") ;
CompteBancaire compteBancaire = service.GetByAbrege("BEU1");
```

Fonction :
```
CompteBancaire Insert(CompteBancaire compteBancaire, Guid? guidCreateur = null);
```

Description :
Ajoute un compte bancaire.
Retourne true si le compte bancaire a été ajouté

Exemple :
```
ICompteBancaireService service = new CompteBancaireService
("http://localhost:12345/Webservices100/BIJOU/") ;
CompteBancaire compteBancaire = new CompteBancaire() ;
//Initialisation du compte bancaire avec toutes les données nécessaires à la conception
d'un compte bancaire
service.Insert(compteBancaire, utilisateurConnecte.Guid);
```

Fonction :
- `CompteBancaire Update(CompteBancaire compteBancaire, Guid? guidUtilisateur = null);`

Description :
Met à jour un compte bancaire.
Retourne true si le compte bancaire a été mis à jour

Exemple :
```
ICompteBancaireService service = new
CompteBancaireService("http://localhost:12345/Webservices100/BIJOU/") ;
CompteBancaire compteBancaire = service. GetById(1);
Guid? guidUtilisateur = null ; //Ou obtenu via ConnectUser
//Modification des propriétés du compte bancaire
service.Update(compteBancaire, guidUtilisateur);
```

Fonction :

```
bool Delete(int idCompteBancaire, Guid? guidUtilisateur = null);
```

Description :

Supprime un compte bancaire dont l'identifiant est passé en paramètre.

Exemple :

```
ICompteBancaireService service = new
CompteBancaireService("http://localhost:12345/Webservices100/BIJOU/") ;
Guid? guidUtilisateur = null; //Ou obtenu via ConnectUser
service.Delete(1, guidUtilisateur);
```

e) CompteGeneralService

Fonctions disponibles :

- `CompteGeneral GetCompteGeneral(string compte);`
- `List<CompteGeneral> GetList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);`
- `CompteGeneral Insert(CompteGeneral compte, bool useSageProcess = true, Guid? guidCreateur = null);`
- `bool UpdateInfosLibres(string numeroCompte, List<InfoLibre> infoLibreList);`

Fonction :

```
CompteGeneral GetCompteGeneral(string compte);
```

Description :

Retourne un compte général par son numéro.
Si le paramètre compte est nul, une ArgumentException est levée.

Exemple :

```
ICompteGeneralService service = new
CompteGeneralService("http://localhost:12345/Webservices100/BIJOU/");
CompteGeneral compte = service.GetCompteGeneral("608855");
```

Fonction :

```
List<CompteGeneral> GetList(Criteria criteria = null, List<Order> orders = null, int
pageNumber = 0, int rowsPerPage = 0);
```

Description :

Retourne la liste des comptes généraux répondant aux critères passés en paramètre.

Exemple :

```
ICompteGeneralService service = new
CompteGeneralService("http://localhost:12345/Webservices100/BIJOU/");
Criteria criteria = new CriteriaComparison("NatureCompte", ComparisonOperator.Equals,
TypeNatureCompte.Charge);
List<CompteGeneral> list = service.GetList(criteria);
```

Fonction :

```
CompteGeneral Insert(CompteGeneral compte, bool useSageProcess = true, Guid?
guidCreateur = null);
```

Description :

Ajoute un nouveau compte général en respectant les règles suivantes :

- Le champ 'NumeroCompte' doit être renseigné.
- Le champ 'Intitule' doit être renseigné.

Champs initialisés si useSageProcess = true :

- NatureCompte
- ReportCompte
- EstAnalytique

Si le paramètre compte est nul, une ArgumentException est levée.

Exemple :

```
CompteGeneral compte = new CompteGeneral();
compte.NumeroCompte = "4011TEST";
compte.Intitule = "Compte à créer";
ICompteGeneralService service = new
CompteGeneralService("http://localhost:12345/Webservices100/BIJOU/");
CompteGeneral compteGeneralInserted = service.Insert(compte, utilisateurConnecte.Guid);
```

Fonction :

```
bool UpdateInfosLibres(string numeroCompte, List<InfoLibre> infoLibreList );
```

Description :

Mets à jour les informations libres du compte général.

Exemple :

```
ICompteGeneralService service = new
CompteGeneralService("http://localhost:12345/Webservices100/BIJOU/");
CompteGeneral compteGeneral = service.GetCompteGeneral("40111000");
ListInfoLibre infoLibreList = compteGeneral.InfosLibres ;
//Modification des informations libres
CompteGeneral compteGeneralInserted = service.UpdateInfosLibres(compte.NumeroCompte,
infoLibreList);
```

f) ContactBanqueService

Fonctions disponibles :

- `ContactBanque GetContactBanque(int idContact);`
- `List<ContactBanque> GetList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);`
- `List<ContactBanque> GetListByBanque(int idBanque, List<Order> orders = null);`
- `List<ContactBanque> ContactsBanqueByIdBanque(string idBanque);`
- `int GetCount(Criteria criteria = null);`
- `ContactBanque Insert(ContactBanque contact, Guid? guidCreateur = null);`
- `ContactBanque Update(ContactBanque contact);`
- `bool Delete(int idContact);`

Fonction :

```
ContactBanque GetContactBanque(int idContact);
```

Description :

Retourne un contact d'une banque dont l'identifiant est passé en paramètre.

Exemple :

```
IContactBanqueService service = new
ContactBanqueService("http://localhost:12345/Webservices100/BIJOU/");
ContactBanque ContactBanque = service. GetContactBanque(1);
```

Fonction :

```
List<ContactBanque> GetList(Criteria criteria = null, List<Order> orders = null, int
pageNumber = 0, int rowsPerPage = 0);
```

Description :

Retourne une liste des contacts des banques.

Exemple :

```
IContactBanqueService service = new
ContactBanqueService("http://localhost:12345/Webservices100/BIJOU/");
List<ContactBanque> contactBanques = service.GetList();
```

Fonction :

```
List<ContactBanque> GetListByBanque(int idBanque, List<Order> orders = null);
```

Description :

Retourne une liste des contacts d'une banque dont l'identifiant est passé en paramètre.

Exemple :

```
IContactBanqueService service = new
ContactBanqueService("http://localhost:12345/Webservices100/BIJOU/");
List<ContactBanque> contactBanques = service.GetListByBanque(1);
```

Fonction :
```
int GetCount(Criteria criteria = null);
```

Description :
Retourne le nombre des contacts des banques.

Exemple :
```
IContactBanqueService service = new
ContactBanqueService("http://localhost:12345/Webservices100/BIJOU/");
Int count = service. GetCount ();
```

Fonction :
```
ContactBanque Insert(ContactBanque contact, Guid? guidCreateur = null);
```

Description :
Ajoute un contact d'une banque.
Retourne une nouvelle instance de ContactBanque correspondante au paramètre contact après insertion.

Exemple :
```
IContactBanqueService service = new ContactBanqueService
("http://localhost:12345/Webservices100/BIJOU/") ;
ContactBanque contactBanque = new ContactBanque() ;
//Initialisation d'un contact d'une banque avec toutes les données nécessaires à la
conception d'un contact d'une banque
ContactBanque inserted = service.Insert(contactBanque, utilisateurConnecte.Guid);
```

Fonction :
```
ContactBanque Update(ContactBanque contactBanque);
```

Description :
Met à jour un contact d'une banque.
Retourne une nouvelle instance de contactBanque correspondante au paramètre contactBanque après mise à jour.

Exemple :
```
IContactBanqueService service = new ContactBanqueService
("http://localhost:12345/Webservices100/BIJOU/") ;
ContactBanque contactBanque = service.GetContactBanque(1)
//Modification des propriétés du contact
ContactBanque updated = service.Update(contactBanque);
```

Fonction :

```
bool Delete(int idContact);
```

Description :

Supprime un contact d'une banque dont l'identifiant est passé en paramètre.
Retourne true si le contact a été supprimé

Exemple :

```
IContactBanqueService service = new ContactBanqueService
("http://localhost:12345/Webservices100/BIJOU/");
service.Delete(1);
```

g) EcheanceService

Fonctions disponibles :

- `List<Echeance> GetList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);`
- `List<Echeance> Echeances(int pageNumber = 0, int rowsPerPage = 0);`
- `List<Echeance> GetEcheancesByTiers(string numeroTiers);`
- `List<RejetBancaire> GetRejetsBancaires(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);`
- `List<RejetBancaire> RejetsBancaires(int pageNumber = 0, int rowsPerPage = 0);`
-

Fonction :
```
List<Echeance> GetList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
```

Description :
Retourne la liste des échéances répondant aux critères passés en paramètre.

Exemple :
```
IEcheanceService service = new EcheanceService("http://localhost:12345/Webservices100/BIJOU/");
List<Echeance> echeances = service.GetList();
```

Fonction :
```
List<Echeance> Echeances(int pageNumber = 0, int rowsPerPage = 0);
```

Description :
Retourne la liste de toutes les échéances (raccourci de GetList sans critère ni tri).

Exemple :
```
IEcheanceService service = new EcheanceService("http://localhost:12345/Webservices100/BIJOU/");
List<Echeance> echeances = service.Echeances();
```

Fonction :
```
List<Echeance> GetEcheancesByTiers(string numeroTiers);
```

Description :
Retourne la liste des échéances du tiers dont le numéro est passé en paramètre.

Exemple :
```
IEcheanceService service = new EcheanceService("http://localhost:12345/Webservices100/BIJOU/");
List<Echeance> echeances = service.GetEcheancesByTiers("CLI001");
```

Fonction :
```
List<RejetBancaire> GetRejetsBancaires(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
```

Description :
Retourne la liste des rejets bancaires répondant aux critères passés en paramètre.

Exemple :
```
IEcheanceService service = new EcheanceService("http://localhost:12345/Webservices100/BIJOU/");
List<RejetBancaire> rejets = service.GetRejetsBancaires();
```

Fonction :
```
List<RejetBancaire> RejetsBancaires(int pageNumber = 0, int rowsPerPage = 0);
```

Description :
Retourne la liste de tous les rejets bancaires (raccourci de GetRejetsBancaires sans critère ni tri).

Exemple :
```
IEcheanceService service = new
EcheanceService("http://localhost:12345/Webservices100/BIJOU/");
List<RejetBancaire> rejets = service.RejetsBancaires();
```

h) EcritureService

Fonctions disponibles :

- `Ecriture GetEcriture(int id, bool includeEcrituresAnalytiques = false);`
- `IList<Ecriture> GetListEcritures(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0, bool includeEcrituresAnalytiques = false);`
- `List<Ecriture> GetEcrituresByJournal(string codeJournal, Periode periode = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0, bool includeEcrituresAnalytiques = false);`
- `List<Ecriture> GetEcrituresByCompteGeneral(string compteGeneral, Periode periode = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0, bool includeEcrituresAnalytiques = false);`
- `List<Ecriture> GetEcrituresByTiers(string numeroTiers, Periode periode = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);`
- `string GetLastPiece(string codeJournal, DateTime date);`
- `List<Ecriture> InsertEcritures(List<Ecriture> list, Guid? guidCreateur = null);`
- `List<Ecriture> InsertPiece(Ecriture ecriture, bool useSageProcess = true, Guid? guidCreateur = null); //Obsolete`
- `List<List<Ecriture>> InsertPieces(List<List<Ecriture>> pieceList, Guid? guidCreateur = null);`
- `bool LettrerEcritures(List<Ecriture> list);`
- `List<EcritureAnalytique> GetListAnalytique(int idEcriture);`
- `List<EcritureAnalytique> GetListAnalytiqueByPlan(int idEcriture, int idPlanAnalytique);`
- `List<EcritureAnalytique> GetListAnalytiqueBySection(int idPlanAnalytique, string section);`
- `bool InsertAnalytiques(List<EcritureAnalytique> list, Guid? guidCreateur = null);`
- `bool DeleteAnalytique(int idEcriture);`
- `bool DeleteAnalytiqueByPlan(int idEcriture, int idPlanAnalytique);`
- `bool UpdateInfosLibres(int idEcriture, List<InfoLibre> infoLibreList);`
- `UpdateModeReglementResult UpdateModeReglement(DataMajModeReglement dataMajModeReglement);`
- `List<EcritureAnalytique> GetListAnalytiqueByCriteria(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);`
- `ElementsLettrageResult GetElementsLettrageParEcriture(ElementsLettrageRequest getElementsLettrageRequest);`

Fonction :
```
Ecriture GetEcriture(int id, bool includeEcrituresAnalytiques = false);
```

Description :
Retourne une écriture par son Id.
Si includeEcrituresAnalytiques est à true, la fonction lit aussi les écritures analytiques associées.

Exemple :

```
IEcritureService service = new
EcritureService("http://localhost:12345/Webservices100/BIJOU/");
Ecriture ecriture = service.GetEcriture(167);
```

Fonction :
```
List<Ecriture> GetListEcritures(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0, bool includeEcrituresAnalytiques = false);
```

Description :
Retourne la liste des écritures répondant aux critères passés en paramètre.
Si includeEcrituresAnalytiques est à true, la fonction lit aussi les écritures analytiques associées.

Exemple :
```
IEcritureService service = new EcritureService("http://localhost:12345/Webservices100/BIJOU/");
List<Ecriture> list = service.GetListEcritures();
```

Fonction :
```
List<Ecriture> GetEcrituresByJournal(string codeJournal, Periode periode = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0, bool includeEcrituresAnalytiques = false);
```

Description :
Retourne la liste des écritures par journal répondant aux critères passés en paramètre.
Si le paramètre codeJournal est nul, une ArgumentException est levée.
Si includeEcrituresAnalytiques est à true, la fonction lit aussi les écritures analytiques associées.

Exemple :
```
IEcritureService service = new EcritureService("http://localhost:12345/Webservices100/BIJOU/");
List<Ecriture> ecritures = service.GetEcrituresByJournal("VTE");
```

Fonction :
```
List<Ecriture> GetEcrituresByCompteGeneral(string compteGeneral, Periode periode = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0, bool includeEcrituresAnalytiques = false);
```

Description :
Retourne la liste des écritures par compte général répondant aux critères passés en paramètre.
Si le paramètre compteGeneral est nul, une ArgumentException est levée.
Si includeEcrituresAnalytiques est à true, la fonction lit aussi les écritures analytiques associées.

Exemple :
```
IEcritureService service = new EcritureService("http://localhost:12345/Webservices100/BIJOU/");
List<Ecriture> ecritures = service.GetEcrituresByCompteGeneral("601019");
```

Fonction :
```
List<Ecriture> GetEcrituresByTiers(string numeroTiers, Periode periode = null,
List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
```

Description :
Renvoyer la liste des écritures par tiers répondant aux critères passés en paramètre.
Si le paramètre numeroTiers est nul, une ArgumentException est levée.

Exemple :
```
IEcritureService service = new
EcritureService("http://localhost:12345/Webservices100/BIJOU/");
List<Ecriture> ecritures = service.GetEcrituresByTiers("BILLO");
```

Fonction :
```
string GetLastPiece(string codeJournal, DateTime date);
```

Description :
Retourne le dernier numéro de pièce pour un journal et une période.
Si le paramètre codeJournal est nul, une ArgumentException est levée.

Exemple :
```
IEcritureService service = new
EcritureService("http://localhost:12345/Webservices100/BIJOU/");
string piece = service.GetLastPiece("ACH", DateTime.Today);
```

Fonction :
```
List<Ecriture> InsertEcritures(List<Ecriture> list, Guid? guidCreateur = null);
```

Description :
Ajoute une liste d'écriture.
Retourne la liste d'écriture ajoutée
Si le paramètre list est nul, une ArgumentException est levée.

Exemple :
```
List<Ecriture> data = new List<Ecriture>();
//Ajout des écritures à insérer dans la liste data
IEcritureService service = new
EcritureService("http://localhost:12345/Webservices100/BIJOU/");
service.InsertEcritures(data, utilisateurConnecte.Guid);
```

Fonction :

```
List<Ecriture> InsertPiece(Ecriture ecriture, bool useSageProcess = true, Guid?
guidCreateur = null);
```

Fonction Obsolète

Description :

Ajoute une nouvelle pièce dont les données sont contenues dans l'écriture passée paramètre en respectant les règles suivantes :

- Le numéro du tiers doit être défini
- Le code journal doit être défini
- Le compte général de contrepartie doit être défini
- Le montant (hors taxe) doit être défini

Renvoie la liste des écritures générées.
Si le paramètre ecriture est nul, une ArgumentException est levée.

Exemple :

```
IEcritureService service = new
EcritureService("http://localhost:12345/Webservices100/BIJOU/");
Ecriture ecriture = new Ecriture();
//Initialisation de l'ecriture contenant toutes les données nécessaires à la conception
d'une pièce comptable.
List<Ecriture> ecritureInjectedList = service.InsertPiece(ecriture,
utilisateurConnecte.Guid);
```

Fonction :

```
List<List<Ecriture>> InsertPieces(List<List<Ecriture>> pieceList, Guid? guidCreateur = null);
```

Description :

Méthode d'insertion d'une liste de liste d'Ecriture.
Puisqu'une pièce comptable est une collection d'écritures comptables. Nous pouvons considérer que cette méthode fait l'insertion d'une liste de pièces comptables.

Cette méthode permet d'assembler des écritures qui vont ensemble, une pièce comptable, et d'affecter un numéro de pièce différent à chaque pièce comptable.
Toutes les écritures doivent avoir :

- Une date d'écriture,
- Un code journal,
- Un montant,
- Un compte général si le compte tiers n'est pas fourni.

Pour chaque pièce comptable les montants doivent être équilibrés (somme des débits = somme des crédits).

Toutes les écritures d'une même pièce comptable doivent :

- Avoir le même code journal.
- Être sur la même période (mois/année).
- Avoir le même numéro de pièce (null ou vide).

Si le paramètre pieceList est nul, une ArgumentException est levée.
Une exception sera levée si :

- Le code journal n'existe pas,
- L'exercice ciblé n'existe pas dans la comptabilité,
- L'exercice ciblé est clôturé,
- La période du journal ciblée est complètement clôturée,
- Le compte général n'existe pas,
- Le compte tiers n'existe pas,
- Le compte tiers est un Client prospect,
- Le code taxe n'existe pas,
- La devise n'existe pas,
- Le mode de règlement n'existe pas.

Renvoie la liste des écritures générées.

Exemple :

```
IEcritureService service = new EcritureService("http://localhost:12345/Webservices100/BIJOU/");
List<List<Ecriture>> pieceToInsert = CreatePiecesCustom();I//Méthode customisée de création de List<List<Ecriture>>
List<List<Ecriture>> piecesInjected = service.InsertPieces(ecriture, utilisateurConnecte.Guid);
```

Fonction :
```
bool LettrerEcritures(List<Ecriture> list);
```

Description :
Lettre une liste d'écritures.
Retourne true si ma liste a été lettrée.
Si le paramètre list est nul, une ArgumentException est levée.

Exemple :
```
IEcritureService service = new
EcritureService("http://localhost:12345/Webservices100/BIJOU/");
List<Ecriture> data = new List<Ecriture>();
//Ajout des écritures à lettrer ensemble dans la liste des écritures data
bool lettree = service.LettrerEcritures(data);
```

Fonction :
```
List<EcritureAnalytique> GetListAnalytique(int idEcriture);
```

Description :
Retourne la liste des écritures analytique d'une écriture.

Exemple :
```
IEcritureService service = new
EcritureService("http://localhost:12345/Webservices100/BIJOU/");
List<EcritureAnalytique> data = service.GetListAnalytique(478);
```

Fonction :
```
List<EcritureAnalytique> GetListAnalytiqueByPlan(int idEcriture, int idPlanAnalytique);
```

Description :
Retourne la liste des écritures analytiques d'un plan analytique d'une écriture comptable.

Exemple :
```
IEcritureService service = new
EcritureService("http://localhost:12345/Webservices100/BIJOU/");
List<EcritureAnalytique> data = service.GetListAnalytique(478, 1);
```

Fonction :
```
List<EcritureAnalytique> GetListAnalytiqueBySection(int idPlanAnalytique, string
section);
```

Description :
Retourne la liste des écritures analytiques d'une section et d'un plan analytique.
Si le paramètre section est nul, une ArgumentException est levée.

Exemple :
```
IEcritureService service = new
EcritureService("http://localhost:12345/Webservices100/BIJOU/");
List<EcritureAnalytique> data = service.GetListAnalytique(1, "929ZZ9");
```

Fonction :
```
bool InsertAnalytique(List<EcritureAnalytique> list, Guid? guidCreateur = null);
```

Description :
Ajoute une liste d'écriture analytique.
Retourne true si la liste a été ajoutée
Si le paramètre list est nul, une ArgumentException est levée.

Exemple :
```
IEcritureService service = new
EcritureService("http://localhost:12345/Webservices100/BIJOU/");
List<EcritureAnalytique> data = new List<EcritureAnalytique>();
//Ajout d'écritures analytiques à la liste data
service.InsertAnalytique(data, utilisateurConnecte.Guid);
```

Fonction :
```
bool DeleteAnalytique(int idEcriture);
```

Description :
Supprime l'analytique d'une écriture.
Retourne true si l'analytique a été supprimée.

Exemple :
```
IEcritureService service = new
EcritureService("http://localhost:12345/Webservices100/BIJOU/");
bool deleted = service.DeleteAnalytique(478);
```

Fonction :
```
bool DeleteAnalytiqueByPlan(int idEcriture, int idPlan);
```

Description :
Supprime l'analytique d'un plan d'une écriture.
Retourne true si l'analytique a été supprimée.

Exemple :
```
IEcritureService service = new
EcritureService("http://localhost:12345/Webservices100/BIJOU/");
bool deleted = service.DeleteAnalytiqueByPlan(478, 1);
```

Fonction :
```
bool UpdateInfosLibres(int idEcriture, List<InfoLibre> infoLibreList);
```

Description :
Met à jour les informations libres d'une écriture.

Exemple :
```
IEcritureService service = new
EcritureService("http://localhost:12345/Webservices100/BIJOU/");
Ecriture ecriture = service.GetEcriture(326);
ListInfoLibre infosLibres = ecriture.InfosLibres;
//Mise à jour des informations libres
bool success = service.UpdateInfosLibres(478, infosLibres);
```

Fonction :
- `UpdateModeReglementResult UpdateModeReglement(DataMajModeReglement dataMajModeReglement);`

Description :
Met à jour le mode de règlement des écritures répondant au critère passé en paramètre.
Une exception est levée si :
- Le DTO DataMajModeReglement est nul,
- Dans le DTO DataMajModeReglement :
  - Aucun IdReglement n'est fourni,
  - Aucun Criteria n'est fourni.

Exemple :
```
IEcritureService service = new
EcritureService("http://localhost:12345/Webservices100/BIJOU");
DataMajModeReglement dataMajModeReglement = new DataMajModeReglement()
{
    Criteria = new CriteriaComparison("Id", ComparisonOperator.Equals, 500),
    IdReglement = 4
};
service.UpdateModeReglement(dataMajModeReglement);
```

Fonction :
```
List<EcritureAnalytique> GetListAnalytiqueByCriteria(Criteria criteria = null,
List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
```

Description :
Retourne la liste des écritures répondant aux critères passés en paramètre.

Exemple :
```
IEcritureService service = new
EcritureService("http://localhost:12345/Webservices100/BIJOU");
Criteria criteria = new CriteriaComparison("DateCreation", ComparisonOperator.LessThan,
DateTime.Now.AddDays(-5));
List<EcritureAnalytique> ecrituresAnalytiques =
service.GetListAnalytiqueByCriteria(criteria);
```

Fonction :

```
ElementsLettrageResult GetElementsLettrageParEcriture(ElementsLettrageRequest
getElementsLettrageRequest);
```

Description :

Retourne la liste des éléments de lettrage des écritures comptables répondant aux critères spécifiés.

Le DTO d'entrée est de type ElementsLettrageRequest et contient un Criteria.

Une exception est levée si :

- getElementsLettrageRequest est null ;
- Le criteria de getElementsLettrageRequest est null.

Exemple :

```
IEcritureService ecritureService = new
EcritureService("http://localhost:12345/Webservices100/BIJOU/");
ElementsLettrageRequest request = new ElementsLettrageRequest
{
    Criteria = new CriteriaComparison(nameof(Ecriture.NumeroPiece),
ComparisonOperator.Equals, "PIECE0001")
};
ecritureService.GetElementsLettrageParEcriture(request);
```

i) ExtraitBancaireService

Fonctions disponibles :

- `List<ReleveBancaire> GetRelevesBancaires(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);`
- `List<ReleveBancaire> RelevesBancaires(int pageNumber = 0, int rowsPerPage = 0);`
- `List<LigneReleveBancaire> GetLignesByReleve(int idReleve);`
- `List<LigneReleveBancaire> LignesReleve(string idRereleve);`
- `List<LigneReleveBancaire> GetLignesRelevesBancaires(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);`
-

Fonction :

```
List<ReleveBancaire> GetRelevesBancaires(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
```

Description :

Retourne la liste des relevés bancaires répondant aux critères passés en paramètre.

Exemple :

```
IExtraitBancaireService service = new ExtraitBancaireService("http://localhost:12345/Webservices100/BIJOU/");
List<ReleveBancaire> releves = service.GetRelevesBancaires();
```

Fonction :

```
List<ReleveBancaire> RelevesBancaires(int pageNumber = 0, int rowsPerPage = 0);
```

Description :

Retourne la liste de tous les relevés bancaires (raccourci de GetRelevesBancaires sans critère ni tri).

Exemple :

```
IExtraitBancaireService service = new ExtraitBancaireService("http://localhost:12345/Webservices100/BIJOU/");
List<ReleveBancaire> releves = service.RelevesBancaires();
```

Fonction :

```
List<LigneReleveBancaire> GetLignesByReleve(int idReleve);
```

Description :

Retourne la liste des lignes du relevé bancaire dont l'identifiant est passé en paramètre.

Exemple :

```
IExtraitBancaireService service = new ExtraitBancaireService("http://localhost:12345/Webservices100/BIJOU/");
List<LigneReleveBancaire> lignes = service.GetLignesByReleve(1);
```

Fonction :
```
List<LigneReleveBancaire> LignesReleve(string idReleve);
```

Description :
Retourne la liste des lignes du relevé bancaire dont l'identifiant, passé sous forme de chaîne, est indiqué en paramètre.
Si le paramètre ne peut pas être converti en entier, une ArgumentException est levée.

Exemple :
```
IExtraitBancaireService service = new
ExtraitBancaireService("http://localhost:12345/Webservices100/BIJOU/");
List<LigneReleveBancaire> lignes = service.LignesReleve("1");
```

Fonction :
```
List<LigneReleveBancaire> GetLignesRelevesBancaires(Criteria criteria = null,
List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
```

Description :
Retourne la liste des lignes de relevés bancaires répondant aux critères passés en paramètre.

Exemple :
```
IExtraitBancaireService service = new
ExtraitBancaireService("http://localhost:12345/Webservices100/BIJOU/");
List<LigneReleveBancaire> lignes = service.GetLignesRelevesBancaires();
```

j) ModeReglementService

Fonctions disponibles :

- `ModeReglement GetByIndice(short indice);`
- `ModeReglement GetByName(string name);`
- `List<ModeReglement> GetList();`

Fonction :
```
ModeReglement GetByIndice(short indice);
```

Description :
Retourne un mode de règlement par son indice.

Exemple :
```
IModeReglementService service = new
ModeReglementService("http://localhost:12345/Webservices100/BIJOU/");
ModeReglement mode = service.GetByIndice(1);
```

Fonction :
```
ModeReglement GetByName(string name);
```

Description :
Retourne un mode de règlement par son nom.
Si le paramètre name est nul, une ArgumentException est levée.

Exemple :
```
IModeReglementService service = new
ModeReglementService(«http://localhost:12345/Webservices100/BIJOU/»);
ModeReglement mode = service.GetByName("Virement");
```

Fonction :
```
List<ModeReglement> GetList();
```

Description :
Retourne la liste des modes de règlement.

Exemple :
```
IModeReglementService service = new
ModeReglementService(«http://localhost:12345/Webservices100/BIJOU/»);
List<ModeReglement> list = service.GetList();
```

k) JournalService

Fonctions disponibles :

- `Journal GetJournal(string codeJournal);`
- `Journal JournauxCode(string code);`
- `List<Journal> GetList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);`
- `List<Journal> Journaux(int pageNumber = 0, int rowsPerPage = 0);`
- `List<PeriodeJournal> GetPeriodesJournal(string code, int pageNumber = 0, int rowsPerPage = 0);`
- `List<PeriodeJournal> GetPeriodesJournalList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);`

Fonction :

```
Journal GetJournal(string codeJournal);
```

Description :

Retourne un journal par son code.
Si le paramètre codeJournal est nul, une ArgumentException est levée.

Exemple :

```
IJournalService service = new
JournalService("http://localhost:12345/Webservices100/BIJOU/");
Journal journal = service.GetJournal("VTE");
```

Fonction :

```
Journal JournauxCode(string code);
```

Description :

Retourne un journal par son code (raccourci de GetJournal).
Si le paramètre code est nul, une ArgumentException est levée.

Exemple :

```
IJournalService service = new
JournalService("http://localhost:12345/Webservices100/BIJOU/");
Journal journal = service.JournauxCode("VTE");
```

Fonction :

```
List<Journal> GetList(Criteria criteria = null, List<Order> orders = null, int
pageNumber = 0, int rowsPerPage = 0);
```

Description :

Renvoyer la liste des journaux répondant aux critères passés en paramètre.

Exemple :

```
IJournalService service = new
JournalService("http://localhost:12345/Webservices100/BIJOU/");
List<Journal> list = service.GetList();
```

Fonction :
```
List<Journal> Journaux(int pageNumber = 0, int rowsPerPage = 0);
```

Description :
Retourne la liste de tous les journaux (raccourci de GetList sans critère ni tri).

Exemple :
```
IJournalService service = new
JournalService("http://localhost:12345/Webservices100/BIJOU/");
List<Journal> journaux = service.Journaux();
```

Fonction :
```
List<PeriodeJournal> GetPeriodesJournal(string code, int pageNumber = 0, int rowsPerPage
= 0);
```

Description :
Retourne la liste des périodes du journal dont le code est passé en paramètre.
Si le paramètre code est nul, une ArgumentException est levée.

Exemple :
```
IJournalService service = new
JournalService("http://localhost:12345/Webservices100/BIJOU/");
List<PeriodeJournal> periodes = service.GetPeriodesJournal("VTE");
```

Fonction :
```
List<PeriodeJournal> GetPeriodesJournalList(Criteria criteria = null, List<Order> orders
= null, int pageNumber = 0, int rowsPerPage = 0);
```

Description :
Retourne la liste des périodes de journaux répondant aux critères passés en paramètre.

Exemple :
```
IJournalService service = new
JournalService("http://localhost:12345/Webservices100/BIJOU/");
List<PeriodeJournal> periodes = service.GetPeriodesJournalList();
```

I) TaxeService

Fonctions disponibles :

- `Taxe GetById(int id);`
- `Taxe GetByCode(string code);`
- `List<Taxe> GetList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);`
- `List<Taxe> Taxes(int pageNumber = 0, int rowsPerPage = 0);`
- `List<MentionExonerationTaxe> GetMentionExonerationTaxeList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);`
- `MentionExonerationTaxe GetMentionExonerationTaxeById(int id);`
- `List<MentionExonerationTaxe> MentionExonerationTaxeList(int pageNumber = 0, int rowsPerPage = 0);`
- `MentionExonerationTaxe MentionExonerationTaxeById(string id);`

Fonction :
`Taxe GetTaxeById(int id);`

Description :
Retourne une taxe par son Id.

Exemple :
```
ITaxeService service = new TaxeService("http://localhost:12345/Webservices100/BIJOU/");
Taxe taxe = service.GetTaxeById(1);
```

Fonction :
`Taxe GetTaxeByCode(string code);`
Si le paramètre code est nul, une ArgumentException est levée.

Description :
Retourne une taxe par son code.

Exemple :
```
ITaxeService service = new TaxeService("http://localhost:12345/Webservices100/BIJOU/");
Taxe taxe = service.GetTaxeByCode("VL4");
```

Fonction :
`List<Taxe> GetList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);`

Description :
Renvoyer la liste des taxes répondant aux critères passés en paramètre.

Exemple :
```
ITaxeService service = new TaxeService("http://localhost:12345/Webservices100/BIJOU/");
List<Taxe> list = service.GetList();
```

Fonction :
```
List<Taxe> Taxes(int pageNumber = 0, int rowsPerPage = 0);
```

Description :
Retourne la page de la liste des taxes.

Exemple :
```
ITaxeService service = new TaxeService("http://localhost:12345/Webservices100/BIJOU/");
List<Taxe> taxes = service.Taxe(2,5);
```

Fonction :
```
List<MentionExonerationTaxe> GetMentionExonerationTaxeList(Criteria criteria = null,
List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
```

Description :
Renvoie la liste des mentions d'exoneration de taxe répondant aux critères passés en paramètre.

Exemple :
```
ITaxeService service = new TaxeService("http://localhost:12345/Webservices100/BIJOU/");
List<MentionExonerationTaxe> mentions = service.GetMentionExonerationTaxeList();
```

Fonction :
```
MentionExonerationTaxe GetMentionExonerationTaxeById(int id);
```

Description :
Renvoie une mention d'exoneration de taxe par son identifiant.

Exemple :
```
ITaxeService service = new TaxeService("http://localhost:12345/Webservices100/BIJOU/");
MentionExonerationTaxe mention = service. GetMentionExonerationTaxeById(2);
```

Fonction :
```
List<MentionExonerationTaxe> MentionExonerationTaxeList(int pageNumber = 0, int
rowsPerPage = 0);
```

Description :
Renvoie la page des mentions d'exonération de taxe.

Exemple :
```
ITaxeService service = new TaxeService("http://localhost:12345/Webservices100/BIJOU/");
List<MentionExonerationTaxe> mentions = service.MentionExonerationTaxeList(2,4);
```

Fonction :

```
MentionExonerationTaxe MentionExonerationTaxeById(string id);
```

Description :

Renvoie une mention d'exoneration de taxe par son identifiant.

Exemple :

```
ITaxeService service = new TaxeService("http://localhost:12345/Webservices100/BIJOU/");
MentionExonerationTaxe mention = service.MentionExonerationTaxeById ("4");
```

m) RegistreRevisionService

Fonctions disponibles :

- `RegistreRevision RegistreRevision(string idEcriture)`
- `RegistreRevision Insert(RegistreRevision registreRevision, Guid? guidCreateur = null);`
- `bool Delete(int idEcriture);`
- `List<RegistreRevisionService> GetList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);`

Fonction :
`RegistreRevision RegistreRevision(string idEcriture);`

Description :
Retourne un registre de révision par l'Id d'écriture correspondante.

Exemple :
```
IRegistreRevisionService service = new RegistreRevision
("http://localhost:12345/Webservices100/BIJOU/");
RegistreRevision registreRevision = service. RegistreRevision("1");
```

Fonction :
`RegistreRevision Insert(RegistreRevision registreRevision, Guid? guidCreateur = null);`Si le paramètre `registreRevision` est nul, une ArgumentException est levée.

Description :
Insert un nouveau registre de révision, puis retourne l'objet créé.

Exemple :
```
RegistreRevision registreToCreate = new RegistreRevision();
//Initialisation du registre avec toutes les données nécessaires à la conception de
l'objet (IdEcriture, etc…)

IRegistreRevisionService service = new RegistreRevision
("http://localhost:12345/Webservices100/BIJOU/");
RegistreRevision registreRevision = service.Insert(registreToCreate,
utilisateurConnecte.Guid);
```

Fonction :
`bool Delete(int idEcriture);`

Description :
Supprime le registre de révision de l'écriture passée en paramètre.

Exemple :
```
IRegistreRevisionService service = new RegistreRevision
("http://localhost:12345/Webservices100/BIJOU/");
bool isDeleted = service.Delete(1);
```

Fonction :

```
List<RegistreRevision> GetList(Criteria criteria = null, List<Order> orders = null, int pageNumber = 0, int rowsPerPage = 0);
```

Description :

Renvoyer la liste des registres de révision répondant aux critères passés en paramètre.

Exemple :

```
IRegistreRevisionService service = new IRegistreRevisionService("http://localhost:12345/Webservices100/BIJOU/");
List<RegistreRevision> list = service.GetList();
```

C. Dictionnaire de données

1. Section commun

a) *Calendrier*

Calendrier	
Id	Int
Intitule	String
PremierJourSemaine	PremierJourSemaine
PremiereSemaineAnnee	PremiereSemaineAnnee
JoursOuvres	JoursCochable
Createur	String
DateModification	DateTime
DateCreation	DateTime
UtilisateurCreateur	String
JoursCommande	JoursCochable
JoursLivraison	JoursCochable

b) *ContactTiers*

ContactTiers	
Id	Int
NumeroTiers	String
IdTypeContact	Short
IdServiceContact	Short
Civilite	Civilite
Nom	String
Prenom	String
Fonction	String
Telephone	String
Gsm	String
Fax	String
Email	String
Skype	String
Facebook	String
LinkedIn	String
Createur	String
DateModification	DateTime
DateCreation	DateTime
UtilisateurCreateur	String

c) Criteria

CriteriaComparison

CriteriaComparison	
FieldName	String
Operator	ComparisonOperator
Value	Object

CriteriaLogical

CriteriaLogical	
Operator	LogicalOperator
Operand1	Criteria
Operand2	Criteria
OperandCollection	IEnumerable<Criteria>

CriteriaIn

CriteriaIn	
FieldName	String
IsIn	Bool
Values	IEnumerable<object>

d) DocumentFileStream

DocumentFileStream	
Id	Guid
Donnees	Byte[]
Nom	String
TypeMime	String
DateCreation	DateTime
DateModification	DateTime
EstRepertoire	Bool
EstArchive	Bool

e) DocumentAvecFileStream

DocumentAvecFileStream	
TypeDocument	TypeDocument
NumeroDocument	String
FileStreamData	DocumentFileStream

f) DocumentFileStreamResult

DocumentFileStreamResult	
NombreElementsSansPagination	Int
NombreElementsAvecPagination	Int
PaginationUtilisee	Pagination
ElementsAvecFileStream	List<DocumentAvecFileStream>
ElementsSansFileStream	List<DocumentSansFileStream>

g) DocumentSansFileStream

DocumentSansFileStream	
TypeDocument	TypeDocument
NumeroDocuments	List<string>

h) GetFilesStreamsRequest

GetFilesStreamsRequest	
Criteria	Criteria
PageNumber	Int
RowsPerPage	Int
Orders	List<Order>
InclureElementsSansFileStreamInResponse	bool

i) EcritureAvecFileStream

EcrituresAvecFileStream	
IdEcriture	Int
FileStreamData	DocumentFileStream

j) EcritureFileStreamResult

EcritureFileStreamResult	
NombreElementsSansPagination	Int
NombreElementsAvecPagination	Int
PaginationUtilisee	Pagination
ElementsAvecFileStream	List<EcritureAvecFileStream>
ElementsSansFileStream	List<EcritureSansFileStream>

k) EcrituresSansFileStream

EcrituresSansFileStream	
Mois	Short
Annee	Int
CodeJournal	String
NumeroPiece	String
IdEcritures	List<int>

l) InfoLibre

InfoLibre	
Name	String
Type	TypeInfoLibre
Size	Int
EstCalculee	Bool
Value	Object

m) JoursCochable

JoursCochable	
Lundi	Bool

Mardi	Bool
Mercredi	Bool
Jeudi	Bool
Vendredi	Bool
Samedi	Bool
Dimanche	Bool

n) ModeleReglement

ModeleReglement	
Id	Int
Intitule	String
Createur	String
DateModification	DateTime
DateCreation	DateTime
UtilisateurCreateur	String

o) NiveauAnalyse

NiveauAnalyse	
Id	Short
Intitule	String

p) Order

Order	
FieldName	String
OrderType	OrderType

q) Pagination

Pagination	
RowsPerPage	Int
PageNumber	Int
Orders	List<Order>

r) Pays

Pays	
Intitule	String
Code	String
CodeEdi	String
CodeIso	String
Createur	String
DateModification	DateTime
DateCreation	DateTime
UtilisateurCreateur	String
Localisation	LocalisationPays

s) ParametrageCompteGeneralTiers

ParametrageCompteGeneralTiers	
Compte	String
TypeParametrage	TypeParametrageComptePrincipal

t) ParametresTiers

ParametresTiers	
TypeTiers	TypeTiers
TypeNumerotationTiers	TypeNumerotationTiers
Longueur	Int
Racine	String
ParametrageCompteGeneral	ParametrageCompteGeneralTiers

u) Periodicite

Periodicite	
Id	Short
Intitule	String

v) Preferences

Preferences	
UnitePoids	UnitePoids

w) Tiers

Tiers	
TypeTiers	TypeTiers
NumeroTiers	String
NumeroTiersPayeur	String
IdCategorieComptable	Short
IdCategorieTarifaire	Short
IdModeleReglement	Int
Intitule	String
Qualite	String
Abrege	String
Contact	String
CompteCollectif	String
Adresse	String
Complement	String
CodePostal	String
Ville	String
Region	String
Pays	String
Telephone	String
Fax	String
Email	String
SiteWeb	String
NumeroSiret	String
NumeroTva	String
CodeEdi	String
CodeNAF	String
Commentaire	String
Langue	LangueSage
EstEnSommeil	Bool
idDepot	Int
NumeroCentraleAchat	String
EnCoursAutorise	Decimal
IdRisque	Short
ModeControleEnCours	ModeControleEnCours
AutorisationLettrageAutomatique	Bool
AutomatiserEcritureMultiEcheances	Bool
ExclureRappelsReleve	Bool
CessionCreance	Bool
FaceBook	String
LinkedIn	String
ExclureTraitementMarketing	Bool
GDPRActif	Bool
InfosLibres	ListInfoLibre
Createur	String

DateModification	DateTime
IdDevise	Short
DateCreation	DateTime
UtilisateurCreateur	String

Client

Hérite de Tiers avec les propriétés supplémentaires suivantes.

Client	
AssuranceCredit	Decimal
PrioriteLivraison	Short
DelaiTransport	Short
FactureParBonLivraison	Bool
IdModeExpedition	Short
NePasAppliquerPenaliteRetard	Bool
IdRepresentant	Int
IdPeriodicite	Short
AutorisationLivraisonPartielle	Bool
TauxRemise	Decimal
TauxEscompte	Decimal
EstProspect	Bool
TauxReleve	Decimal
TauxRFA	Decimal
TypeCodeEDI	TypeCodeEDI
FactureElectroniqueAssujetissementTVA	FactureElectroniqueAssujettissement
FactureElectroniqueAutreIdentifiantType	FactureElectroniqueAutreIdentifiantType
FactureElectroniqueAutreIdentifiantValeur	String
FactureElectroniqueTypeEntite	FactureElectroniqueTypeEntite
FactureElectroniqueControleEmission	FactureElectroniqueControleEmission

Fournisseur

Hérite de Tiers avec les propriétés supplémentaires suivantes.

Fournisseur	
DelaiApprovisionnement	Short
IdConditionLivraison	Short
IdRepresentant	Int
TauxRemise	Decimal
TauxEscompte	Decimal
JoursCommande	JoursCochable
JoursLivraison	JoursCochable
IdCalendrier	Int
TauxReleve	Decimal
TauxRFA	Decimal
TypeCodeEDI	TypeCodeEDI
FactureElectroniqueAssujetissementTVA	FactureElectroniqueAssujettissement
FactureElectroniqueAutreIdentifiantType	FactureElectroniqueAutreIdentifiantType
FactureElectroniqueAutreIdentifiantValeur	String
FactureElectroniqueTypeEntite	FactureElectroniqueTypeEntite

Salarie

Hérite de Tiers avec les propriétés supplémentaires suivantes.

Salarie	
AssuranceCredit	Decimal

AutreTiers

Hérite de Tiers avec les propriétés supplémentaires suivantes.

AutreTiers	
AssuranceCredit	Decimal
NePasAppliquerPenaliteRetard	Bool
IdRepresentant	Int
TypeCodeEDI	TypeCodeEDI
FactureElectroniqueAssujetissementTVA	FactureElectroniqueAssujettissement
FactureElectroniqueAutreIdentifiantType	FactureElectroniqueAutreIdentifiantType
FactureElectroniqueAutreIdentifiantValeur	String
FactureElectroniqueTypeEntite	FactureElectroniqueTypeEntite

x) TypeContact

TypeContact	
Id	Short
Intitule	String

y) UtilisateurSage

UtilisateurSage	
Nom	String
Guid	Guid nullable

z) WebSqlParameter

WebSqlParameter	
ParameterName	String
SqlDbType	DbType
Direction	ParameterDirection
IsNullable	Bool
Value	Objet
Precision	Byte
Scale	Byte
Size	int

2. Section Gestion Commerciale

a) Abonnement

Abonnement	
Id	Int
AbonnementTypeTiers	AbonnementTypeTiers
AbonnementType	AbonnementType
NumeroTiers	String
Modele	String
Intitule	String
Contrat	String
Duree	Short
TypeDureeAbonnement	AbonnementTypeDuree
DateDebut	DateTime
DateFin	DateTime
TypeReconduction	AbonnementReconduction
DateResiliation	DateTime
DateFinAbonnement	DateTime
TypePieceGeneree	Short
IdSouche	Short
Periodicite	Short
TypePeriodicite	AbonnementTypePeriodicite
DelaiPreavis	Short
TypeDelaiPreavis	AbonnementDureeDelaisPreavis
TypeGeneration	AbonnementDateReference
NbJoursGeneration	Short
JourTombeeGeneration1	Short
JourTombeeGeneration2	Short
JourTombeeGeneration3	Short
JourTombeeGeneration4	Short
JourTombeeGeneration5	Short
JourTombeeGeneration6	Short
TypeLivraison	AbonnementDateReference
NbJoursLivraison	Short
JourTombeeLivraison1	Short
JourTombeeLivraison2	Short
JourTombeeLivraison3	Short
JourTombeeLivraison4	Short
JourTombeeLivraison5	Short
JourTombeeLivraison6	Short
SourceDepot	AbonnementSourceGeneration
SourceEntete	AbonnementSourceGeneration
SourceTarif	AbonnementSourceGeneration
SourceRemise	AbonnementSourceGeneration
SourceCategorieComptable	AbonnementSourceGeneration

SourceRepresentant	AbonnementSourceGeneration
SourceEscompte	AbonnementSourceGeneration
SourceEcheance	AbonnementSourceGeneration
SourceContact	AbonnementSourceContact
Createur	String
DateModification	DateTime
IdModifResiliation	Short
DateCreation	DateTime
UtilisateurCreateur	String

b) AbonnementEnTete

AbonnementEntete	
IdAbonnement	Int
Reference	String
IdCollaborateur	Int
Periodicite	Short
IdDevise	Short
CoursDevise	Decimal
IdAdresseLivraison	Int
NumeroTiersPayeur	String
IdModeExpedition	Short
NbFactures	Short
TauxExcompte	Decimal
SectionAnalytique	String
Donnee1	String
Donnee2	String
Donnee3	String
Donnee4	String
IdConditionLivraison	Short
IdCategorieTarifaire	Short
IdCategorieComptable	Short
Langue	LangueSage
EstGenere	Bool
Contact	String
Createur	String
DateModification	DateTime
IdCentraleAchat	String
CompteGeneral	String
InfosLibres	ListInfoLibre
DateCreation	DateTime
UtilisateurCreateur	String
BaseCalculEcheances	AbonnementCalculEcheance
BaseCalcul	Short
BLParFacture	Bool
Colisage	Short
Regime	Short
Transaction	Short
TypeColisage	Short
NumeroIFRS	String

c) AbonnementLigne

AbonnementLigne	
Id	Int
IdAbonnement	Int
NumeroLigne	Int
Reference	String
Designation	String
Reconduction	ReconductionLigne
Createur	String
DateModification	DateTime
InfosLibres	ListInfoLibre
TexteSupplementaire	String
DateCreation	DateTime
UtilisateurCreateur	String

AbonnementLigneArticle

Hérite de AbonnementLigne avec les propriétés supplémentaires suivantes.

AbonnementLigneArticle	
RefArticle	String
IdGamme1	Int
IdGamme2	Int
Quantite	Decimal
PoidsNet	Decimal
PoidsBrut	Decimal
RemiseType1	RemiseTypeLigne
RemiseValeur1	Decimal
RemiseType2	RemiseTypeLigne
RemiseValeur2	Decimal
RemiseType3	RemiseTypeLigne
RemiseValeur3	Decimal
PrixUnitaire	Decimal
PrixUnitaireTTC	Decimal
SectionAnalytique	String
RefFournisseur	String
Conditionnement	String
QuantiteConditionnement	Decimal
LigneTTC	TypePrix
IdDepot	Int
IdCollaborateur	Int
PrixRevientUnitaire	Decimal
PrixUnitaireDevise	Decimal
MontantHT	Decimal
MontantTTC	Decimal
CMUP	Decimal
RefArticleCompose	String
EstValorise	Bool
EstNomenclature	Bool
CodeTaxe1	String
CodeTaxe2	String
CodeTaxe3	String
Periodicite	PeriodiciteLigne
DateDebutFacturation	DateTime
DateFinFacturation	DateTime
Prorata	AbonnementProrataLigne
GestionAnnee	Bool
NbPeriodes	Short
ReferenceExterne	String

AbonnementLigneTotal

Hérite de AbonnementLigne avec les propriétés supplémentaires suivantes.

AbonnementLigneTotal	
Quantite	Decimal
PoidsBrut	Decimal
PoidsNet	Decimal
MontantHT	Decimal
MontantTTC	Decimal

AbonnementLigneTexte

Hérite de AbonnementLigne avec les propriétés supplémentaires suivantes.

AbonnementLigneTexte	
ReferenceExterne	String

d) AbonnementPeriode

AbonnementPeriode	
IdAbonnement	Int
DateFin	DateTime
DateGeneration	DateTime
DateLivraison	DateTime
AbonnementEtatPeriodicite	AbonnementEtatPeriodicite
Createur	String
DateModification	DateTime
DateCreation	DateTime
UtilisateurCreateur	String

e) AdresseLivraison

AdresseLivraison	
Id	Int
NumeroTiers	String
IdModeExpedition	Short
IdConditionLivraison	Short
Intitule	String
Contact	String
Adresse	String
Complement	String
CodePostal	String
Ville	String
Region	String
Pays	String
Telephone	String
Fax	String
Email	String
EstPrincipale	Bool
Createur	String
DateModification	DateTime
DateCreation	DateTime
UtilisateurCreateur	String
Commentaire	String
DelaiTransport	Short

f) Article

Article	
Reference	String
Intitule	String
CodeFamille	String
TypeArticle	TypeArticle
TypeNomenclature	TypeNomenclature
TypeSuiviStock	TypeSuiviStock
TypeEscompte	TypeEscompte
UnitePoids	UnitePoids
PoidsNet	Decimal
PoidsBrut	Decimal
IdUniteVente	Short
DelaiLivraison	Int
Garantie	Int
PrixAchat	Decimal
PrixUnitaireNet	Decimal
PrixVente	Decimal
Coefficient	Decimal
Statistique1	String
Statistique2	String
Statistique3	String
Statistique4	String
Statistique5	String
Langue1	String
Langue2	String
CodeEDI	String
CodeBarres	String
Photo	String
IdCatalogue1	Short
IdCatalogue2	Short
IdCatalogue3	Short
IdCatalogue4	Short
HorsStatistique	Bool
VenteAuDebit	Bool
NonImpression	Bool
Contremarque	Bool
FacturationPoids	Bool
FacturationForfait	Bool
Transfere	Bool
Publie	Bool
EstEnSommeil	Bool
InfosLibres	ListInfoLibre
EstEnPrixTTC	Bool
Createur	String

DateModification	DateTime
FraisFixe1	FraisFixe
FraisFixe2	FraisFixe
FraisFixe3	FraisFixe
DateCreation	DateTime
UtilisateurCreateur	String
AutorisationModificationNomenclatureEnSaisie	Bool
InterdireCommande	Bool
ExclureReapprovisionnement	Bool

ArticleStandard

Hérite de Article.

ArticleGamme

Hérite de Article avec les propriétés supplémentaires suivantes.

ArticleGamme	
IdGamme1	Int
IdGamme2	Int

g) ArticleFournisseur

ArticleFournisseur	
IdArticle	String
IdFournisseur	String
ReferenceFournisseur	String
PrixAchat	Decimal
IdUnite	Int
Conversion	Decimal
DelaiApprovisionementEnJours	Int
GarantieEnMois	Int
Colisage	Decimal
QuantiteMini	Decimal
TypeTarification	TypeTarificationFournisseur
Gamme	Int
EstFournisseurPrincipal	Bool
PrixAchatDevise	Decimal
IdDevise	Int
Remise	Decimal
TypeRemise	TypeRemise
CodeBarres	String
Createur	String
DateModification	DateTime
DateCreation	DateTime
UtilisateurCreateur	String

h) Catalogue

Catalogue	
Id	Int
Nom	String
IdCatalogueParent	Int
SousCatalogues	List<Catalogue>
DateCreation	DateTime
UtilisateurCreateur	String

i) CategorieComptable

CategorieComptable	
Id	Short
Intitule	String
TaxationHorsFrance	Bool

j) CategorieTarifaire

CategorieTarifaire	
Id	Short
Intitule	String
EstPrixTTC	Bool

k) Collaborateur

Collaborateur	
Id	Int
Nom	String
Prenom	String
Service	String
Matricule	String
Fonction	String
Adresse	String
Complement	String
CodePostal	String
Ville	String
Region	String
Pays	String
Telephone	String
Fax	String
Gsm	String
Email	String
EstVendeur	Bool
EstCaissier	Bool
EstControleur	Bool
EstAcheteur	Bool
EstChageRecouvrement	Bool
EstResponsableFinancier	Bool
Facebook	String
LinkedIn	String
Skype	String
Createur	String
DateModification	DateTime
DateCreation	DateTime
UtilisateurCreateur	String
EstEnSommeil	Bool
EstChefVente	Bool
IdChefVente	Int

l) CombinaisonGammeArticle

CombinaisonGammeArticle	
RefArticle	String
IdGamme1	Int
IntituleGamme1	String
IdGamme2	Int
IntituleGamme2	String
Reference	String
CodeBarres	String
PrixAchat	Decimal
EnSommeil	Bool

m) ConditionLivraison

ConditionLivraison	
Id	Short
Intitule	String
Code	String

n) Conditionnement

Conditionnement	
Id	Short
Intitule	String

o) ConditionnementArticle

ConditionnementArticle	
ReferenceArticle	String
Id	Int
Intitule	String
Quantite	Decimal
Reference	String
CodeBarres	String
EstPrincipal	Bool
CodeEDI	String
Createur	String
DateModification	DateTime
DateCreation	DateTime
UtilisateurCreateur	String

p) ConditionReglementDocument

ConditionReglementDocument	
DomaineDocument	DomaineDocument
TypeDocument	TypeDocument
TypeReglement	TypeReglement
NumeroDocument	String
DatePaiement	DateTime
Libelle	String
Pourcent	Decimal
Montant	Decimal
MontantDevise	Decimal
Equilibre	Bool
NumeroEcriture	Int
Regle	Bool
IdReglement	Short
NumeroCaisse	Int
TypeDocumentOrigine	TypeDocument
Createur	String
DateModification	DateTime
DateCreation	DateTime
AdressePaiement	String
NumeroDocumentAcompte	String

q) Depot

Depot	
Id	Int
Intitule	String
Code	String
Adresse	String
Complement	String
CodePostal	String
Ville	String
Region	String
Pays	String
Contact	String
Email	String
Telephone	String
Fax	String
EstPrincipal	Bool
IdEmplacementDefaut	Int
IdCategorieComptable	Short
Createur	String
SoucheDocumentVente	Short
SoucheDocumentAchat	Short
SoucheDocumentInterne	Short
DateModification	DateTime
DateCreation	DateTime
UtilisateurCreateur	String

r) DetailParametreNomenclature

DetailParametreNomenclature	
RefArticle	String
IdGamme1	Short
IdGamme2	Short
Designation	String
Quantite	Decimal
PrixUnitaire	Decimal
PrixRevientUnitaire	Decimal
RemiseType1	RemiseTypeLigne
RemiseType2	RemiseTypeLigne
RemiseType3	RemiseTypeLigne
RemiseValeur1	Decimal
RemiseValeur2	Decimal
RemiseValeur3	Decimal
CodeTaxe1	String
CodeTaxe2	String
CodeTaxe3	String
NumLotSerie	String
ComplementLotSerie	String
DatePeremption	Date
DateFabrication	Date

s) Document

Document	
TypeDocument	TypeDocument
Domaine	DomaineDocument
NumeroDocument	String
Date	DateTime
Reference	String
NumeroTiers	String
NumeroTiersPayeur	String
Statut	StatutDocument
Provenance	ProvenanceDocument
DateLivraison	DateTime
DateLivraisonRealisee	DateTime
DateExpedition	DateTime
SectionAnalytique	String
Contact	String
NbFactures	Short
NbColis	Short
TauxEscompte	Decimal
Langue	LangueSage
Donnees1	String
Donnees2	String
Donnees3	String
Donnees4	String
EstImprime	Bool
EstTransfere	Bool
EstCloture	Bool
EstReliquat	Bool
IdSouche	Short
IdCollaborateur	Int
IdDepot	Int
IdAdresseLivraison	Int
IdModeExpedition	Short
IdConditionLivraison	Short
IdCategorieComptable	Short
IdCategorieTarifaire	Short
IdModeleReglement	Int
IdDevise	Short
CoursDevise	Decimal
FraisExpedition	Decimal
TypeFraisExpedition	TypeFraisModeExpedition
TypePrixFraisExpedition	TypePrix
FrancoDePort	Decimal
TypeFrancoPort	TypeFrancoModeExpedition
ExpeditionCodeTaxe1	String

ExpeditionCodeTaxe2	String
ExpeditionCodeTaxe3	String
ExpeditionTaux1	Decimal
ExpeditionTaux2	Decimal
ExpeditionTaux3	Decimal
NumeroCentraleAchat	String
MontantHT	Decimal
Createur	String
DateModification	DateTime
DateCreation	DateTime
UtilisateurCreateur	String
TotalHTNet	Decimal
TotalTTC	Decimal
NetAPayer	Decimal
MontantRegle	Decimal
NumeroCentraleAchat	String

DocumentAchat

Hérite de Document avec les propriétés supplémentaires suivantes.

DocumentAchat	
PaiementTVADebit	Bool
DateDebutAbonnement	DateTime
DateFinAbonnement	DateTime
DateDebutPeriodeAbonnement	DateTime
DateFinPeriodeAbonnement	DateTime
GuidFichierPdf	Guid ?

DocumentVente

Hérite de Document avec les propriétés supplémentaires suivantes.

DocumentVente	
DateDebutAbonnement	DateTime
DateFinAbonnement	DateTime
DateDebutPeriodeAbonnement	DateTime
DateFinPeriodeAbonnement	DateTime
IdMotifDevisPerdu	Short
PaiementTvaDebit	Bool
GuidFichierPdf	Guid ?

DocumentStock

Hérite de Document

DocumentDepotDepot

Hérite de DocumentStock avec les propriétés supplémentaires suivantes.

DocumentDepotDepot	
IdDepotOrigine	Int
IdDepotDestination	Int

DocumentInterne

Hérite de Document

t) DocumentInterneInfo

DocumentInterneInfo	
TypeMouvementStockDocInterne	TypeMouvementStockDocInterne
Libelle	String
TypeDocument	TypeDocument
IdDepot	Int

u) Emplacement

Emplacement	
Id	Int
Intitule	String
IdDepot	Int
Code	String
ZoneEmplacement	ZoneEmplacement
TypeEmplacement	TypeEmplacement
Createur	String
DateModification	DateTime
DateCreation	DateTime
UtilisateurCreateur	String

v) EnumereConditionnement

EnumereConditionnement	
IdConditionnement	Int
Intitule	String
Quantite	Decimal
Createur	String
DateModification	DateTime
DateCreation	DateTime
CodeEDI	String
Createur	String
UtilisateurCreateur	String

w) Famille

Famille	
CodeFamille	String
Intitule	String
TypeFamille	TypeFamille
Createur	String
DateModification	DateTime
DateCreation	DateTime
UtilisateurCreateur	String

x) FichierLieArticle

FichierLieArticle	
Reference	String
NomFichier	String
Commentaire	String
Id	Int
TypeMime	String
Createur	String
DateModification	DateTime

y) FichierLieDocument

FichierLieDocument	
NumeroDocument	String
TypeDocument	TypeDocument
Intitule	String
NomFichier	String
Id	Int
Transmettre	Short
Createur	String
DateModification	DateTime

z) FraisFixe

FraisFixe	
Intitule	String
Valeur1	Decimal
Type1	TypeFrais
Valeur2	Decimal
Type2	TypeFrais
Valeur3	Decimal
Type3	TypeFrais

aa) Gamme

Gamme	
Id	Short
Intitule	String

bb) GammeArticle

GammeArticle	
Id	Short
Intitule	String

cc) InformationsDocumentsDestinationTransformation

InformationsDocumentDestinationTransformation	
TypeDocument	TypeDocument
NumeroDocument	String
Date	DateTime
DateLivraison	DateTime
InfosLibres	ListInfoLibre
CompleterDocumentDestinationSiExistant	Bool

dd) InformationsDocumentsOrigines

InformationsDocumentsOrigines	
InformationsDocumentOrigineList	List<InfoDocumentOrigineTransformation>
TransformerLigneArticleNonSuiviEnStockDocument	Bool ?
TransformerLigneCommentaireDocument	Bool ?
ConserverDocumentSiReliquat	Bool

ee) InfoDocumentOrigineTransformation

InfoDocumentOrigineTransformation	
TypeDocument	TypeDocument
NumeroDocument	String
ConserverDocumentSiReliquat	Bool ?
Lignes	List<LigneDocumentOrigineTransformation>

ff) InfosLotSerieTransformation

InfosLotSerieTransformation	
NumeroLotSerie	String
ComplementLotSerie	String
Quantite	Decimal
DatePeremption	DateTime
DateFabrication	DateTime
InfosLibres	ListInfoLibre

gg) LigneDocument

LigneDocument	
Id	Int
TypeDocument	TypeDocument
Domaine	DomaineDocument
NumeroDocument	String
NumeroDocumentBC	String
NumeroDocumentBL	String
NumeroDocumentPL	String
NumeroDocumentDE	String
NumeroTiers	String
Date	DateTime
DateBC	DateTime
DateBL	DateTime
DatePL	DateTime
DateDE	DateTime
NumeroLigne	Int
Designation	String
Createur	String
DateModification	DateTime
InfosLibres	ListInfoLibre
DateCreation	DateTime
UtilisateurCreateur	String
TexteSupplementaire	String

LigneArticle

Hérite de LigneDocument avec les propriétés supplémentaires suivantes.

LigneArticle	
RefArticle	String
IdGamme1	Int
IdGamme2	Int
DateLivraison	DateTime
PrixUnitaire	Decimal
PrixUnitaireBC	Decimal
PrixRevientUnitaire	Decimal
PrixUnitaireTTC	Decimal
PrixUnitaireDevise	Decimal
MontantHT	Decimal
MontantTTC	Decimal
CMUP	Decimal
Quantite	Decimal
QuantiteBC	Decimal
QuantiteBL	Decimal
QuantitePL	Decimal
QuantiteDE	Decimal
PoidsNet	Decimal
PoidsBrut	Decimal
Conditionnement	String
QuantiteConditionnement	Decimal
RefFournisseur	String
RefArticleCompose	String
SectionAnalytique	String
EstValorise	Bool
EstNomenclature	Bool
LigneTTC	TypePrix
IdDepot	Int
IdCollaborateur	Int
RemiseType1	RemiseTypeLigne
RemiseValeur1	Decimal
RemiseType2	RemiseTypeLigne
RemiseValeur2	Decimal
RemiseType3	RemiseTypeLigne
RemiseValeur3	Decimal
CodeTaxe1	String
CodeTaxe2	String
CodeTaxe3	String
Taux1	Decimal
Taux2	Decimal
Taux3	Decimal
IdEmplacement	Int

NumLotSerie	String
ComplementLotSerie	String
DatePeremption	DateTime
DateFabrication	DateTime
EstTypeRemisePied	Bool
EstTypeRemiseExceptionnelle	Bool
ReferenceExterne	String

LigneDepotDepot

Hérite de LigneArticle avec les propriétés supplémentaires suivantes.

LigneDepotDepot	
IdEmplacementOrigine	Int
IdEmplacementDestination	Int

LigneTexte

Hérite de LigneDocument avec les propriétés supplémentaires suivantes.

LigneTexte	
ReferenceExterne	String

LigneTotal

Hérite de LigneDocument avec les propriétés supplémentaires suivantes.

LigneTotal	
Quantite	Decimal
PoidsBrut	Decimal
PoidsNet	Decimal
MontantHT	Decimal
MontantTTC	Decimal

LigneDocumentOrigineTransformation

LigneDocumentOrigineTransformation	
Id	Int
IdPosition	Int
RefArticle	String
Quantite	Decimal ?
PrixUnitaire	Decimal ?
InfosLibres	ListInfoLibre
SelectionLotSerieAutomatique	Bool
InformationsLotSerie	List<InformationLotSerieTransformation>

hh) LignesDocumentsData

LignesDocumentsData	
Lignes	List<LigneDocument>
GlobalParameter	AdditionalParams

ii) ListUpdateEnteteDoc

ListUpdateEnteteDoc	
Documents	List<UpdateEnteteDocument>
GuidUtilisateur	Guid?

jj) LotSerie

LotSerie	
RefArticle	String
NumLotSerie	String
Complement	String
DatePeremption	DateTime
DateFabrication	DateTime
Quantite	Decimal
QuantiteRestante	Decimal
QuantiteReserve	Decimal
EstEpuise	Bool
IdDepot	Int
TypeMouvement	TypeMouvement
Createur	String
DateModification	DateTime
DateCreation	DateTime
UtilisateurCreateur	String

kk) ModeExpedition

ModeExpedition	
Id	Short
Intitule	String
Code	String
ReferenceArticleFrais	String
TypeFrais	TypeFraisModeExpedition
ValeurFrais	Decimal
EstTypeLigneFrais	TypePrix
TypeFranco	TypeFrancoModeExpedition
ValeurFranco	Decimal
EstTypeLigneFranco	TypePrix

ll) MotifPerteDevis

MotifPerteDevis	
Id	Short
Intitule	String

mm) MotifResiliation

MotifResiliation	
Id	Short
Intitule	String

nn) NomenclatureArticle

NomenclatureArticle	
RefArticleCompose	String
RefArticleComposant	String
Quantite	Decimal
IdGamme1	Short
IdGamme2	Short
EstQuantiteVariable	Bool
Operation	String
IndexOrdre	Int
EstSousTraitance	Bool
IdDepot	Short
Createur	String
DateModification	DateTime
DateCreation	DateTime
UtilisateurCreateur	String

oo) ParametreEnumereGamme

ParametreEnumereGamme	
Inclure	Bool
Enumeres	IEnumerable<String>

pp) ParametreInsertArticle

ParametreInsertArticle	
Article	Article
UseSageProcess	Bool
GuidCreateur	Guid ?
ParametreEnumereGamme1	ParametreEnumereGamme
ParametreEnumereGamme2	ParametreEnumereGamme

qq) ParametreNomenclature

ParametreNomenclature	
ReferenceArticle	String
IdGamme1	Short
IdGamme2	Short
Details	List<DetailParametreNomenclature>

rr) PrixUnitaireLigneArticle

PrixUnitaireLigneArticle	
ReferenceArticle	String
IdGamme1	Int
IdGamme2	Int
Prix	Decimal
TypePrix	TypePrix
PrixDevise	Decimal

ss) PrixUnitaireLigneArticleParams

PrixUnitaireLigneArticleParams	
ReferenceArticle	String
IntituleConditionnement	String
TypeDocument	TypeDocument
Quantite	Decimal
IdDevise	Short
CoursDevise	Decimal ?
NumeroTiers	String
IdCategorieTarifaire	Int
IdGamme1	Int
IdGamme2	Int
IdDepot	Int

tt) Reglement

Reglement	
Id	Int
Valeur	Decimal
TypeValeur	TypeRepartitionReglement
MontantDevise	Decimal
Date	DateTime
Libelle	String
IdModeReglement	Short
Regle	Bool
Createur	String
DateModification	DateTime
DateCreation	DateTime
UtilisateurCreateur	String

uu) RemiseFamilleClient

RemiseFamilleClient	
NumeroTiers	String
CodeFamille	String
Remise	Decimal
Createur	String
DateModification	DateTime
DateCreation	DateTime
UtilisateurCreateur	String

vv) RoutageReceptionFactureElectronique

RoutageReceptionFactureElectronique	
Type	TypeCodeRoutage
CodeRoutage	String
Intitule	String
EstActif	Bool
NumeroTiers	String
Index	Int
IdentifiantUnique	Int
UtilisateurCreateur	String
Createur	String
DateCreation	DateTime
DateModification	DateTime

ww) RoutageReceptionFactureElectroniqueInsertDataLight

RoutageReceptionFactureElectroniqueInsertDataLight	
Type	TypeCodeRoutage
CodeRoutage	String
Intitule	String
EstActif	Bool

xx) RoutageReceptionFactureElectroniqueInsertDataWithNumeroTiers

RoutageReceptionFactureElectroniqueInsertDataWithNumeroTiers	
Type	TypeCodeRoutage
CodeRoutage	String
Intitule	String
EstActif	Bool
NumeroTiers	String

yy) RoutageReceptionFactureElectroniqueInsertLightWithNumeroTiersCollectionData

RoutageReceptionFactureElectroniqueInsertLightWithNumeroTiersCollectionData	
GuidCreateur	Guid ?
RoutageReceptionFactureElectroniqueCollection	IEnumerable<RoutageReceptionFactureElectroniqueInsertDataWithNumTiers>

zz) RoutageReceptionFactureElectroniqueInsertToNumTiersCollectionData

RoutageReceptionFactureElectroniqueInsertToNumTiersCollectionData	
NumeroTiers	String
GuidCreateur	Guid ?
RoutageReceptionFactureElectroniqueCollection	IEnumerable<RoutageReceptionFactureElectroniqueInsertDataLight>

aaa) RoutageReceptionFactureElectroniqueUpdateCollectionData

RoutageReceptionFactureElectroniqueInsertToNumTiersCollectionData	
RoutageReceptionFactureElectroniqueCollection	IEnumerable<RoutageReceptionFactureElectronique>

bbb) SoucheAchat

SoucheAchat	
Id	Short
Intitule	String
EstValide	Bool
CodeJournal	String
CodeJournalSituation	String

ccc) SoucheInterne

SoucheInterne	
Id	Int
Intitule	String
EstValide	Bool

ddd) SoucheVente

SoucheVente	
Id	Int
Intitule	String
EstValide	Bool
CodeJournal	String
CodeJournalSituation	String

eee) StatistiqueArticle

StatistiqueArticle	
Niveau	Short
Intitule	String
Valeurs	List<ValeurStatistiqueArticle>

fff) Stock

Stock	
IdDepot	Int
RefArticle	String
IdGamme1	Int
IdGamme2	Int
QteStock	Decimal
StockPrepare	Decimal
Createur	String
DateModification	DateTime

ggg) StockDepot

StockDepot	
IdDepot	Int
RefArticle	String
IdGamme1	Int
IdGamme2	Int
QteStock	Decimal
StockPrepare	Decimal
QteReserve	Decimal
QteCommande	Decimal
QteMini	Decimal
QteMaxi	Decimal
EstPrincipal	Bool
IdEmplacementPrincipal	Int
MontantStock	Decimal
StockMouvemente	Bool
Createur	String
DateModification	DateTime
DateCreation	DateTime
UtilisateurCreateur	String

hhh) StockEmplacement

StockEmplacement	
IdDepot	Int
RefArticle	String
IdGamme1	Int
IdGamme2	Int
QteStock	Decimal
StockPrepare	Decimal
IdEmplacement	Int
IntituleEmplacement	String
TypeEmplacement	TypeEmplacement
Createur	String
DateModification	DateTime
DateCreation	DateTime
UtilisateurCreateur	String

iii) TarifArticle

TarifArticle	
RefArticle	String
IdGamme1	Int
Gamme1	String
IdGamme2	Int
Gamme2	String
IdCategorieTarifaire	Short
IntituleCategorieTarifaire	String
Coefficient	Decimal
PrixVente	Decimal
Remise	Decimal
TypePrix	TypePrix
DateModification	DateTime

jjj) TarifClient

TarifClient	
RefArticle	String
IdGamme1	Short
IdGamme2	Short
Gamme1	String
Gamme2	String
NumeroTiers	String
Coefficient	Decimal
PrixVente	Decimal
Remise	Decimal
TypePrix	TypePrix
RefArticleClient	String
DateModification	DateTime
DateCreation	DateTime
UtilisateurCreateur	String

kkk) TarifFournisseur

TarifFournisseur	
RefArticle	String
IdGamme1	Short
IdGamme2	Short
Gamme1	String
Gamme2	String
PrixAchat	Decimal
IdFournisseur	String

lll) TarifRemise

TarifRemise	
RefArticle	String
IdGamme1	Short
IdGamme2	Short
Gamme1	String
Gamme2	String
NumeroTiers	String
IdCategorieTarifaire	Short
IntituleCategorieTarifaire	String
BorneMin	Decimal
BorneMax	Decimal
PrixVenteRemise	Decimal
TypeTarification	TypeTarificationQuantiteMontant

mmm) TransformationDocumentInfos

TransformationDocumentInfos	
InformationsDocumentDestination	InformationsDocumentDestinationTransformation
InformationsDocumentsOrigines	InformationsDocumentsOrigines
LignesDocumentAdditionelles	LignesDocumentsData
GuidCreateur	Guid ?

nnn) UniteVente

UniteVente	
Id	Short
Nom	String
CodeEdi	String

ooo) UpdateEnteteDocument

UpdateEnteteDocument	
NumeroDocument	String
TypeDocument	TypeDocument
ProprietesDocument	UpdateEnteteData
InfosLibresList	ListInfoLibre

ppp) UpdateEnteteData

UpdateEnteteData	
Champ	ChampsDocumentUpdate
Value	object

qqq) ValeurStatistiqueArticle

ValeurStatistiqueArticle	
IndexStatistiqueParent	Short
Intitule	String

3. Section Comptabilité

a) Banque

Banque	
Id	Int
Abrege	String
CodeBic	String
Intitule	String
interlocuteur	String
Adresse	String
ComplementAdresse	String
CodePostal	String
Ville	String
Region	String
NomPays	String
Telephone	String
Telecopie	String
Email	String
SiteInternet	String
SiteRecuperationExtrait	String
Createur	String
DateModification	DateTime
DateCreation	DateTime
UtilisateurCreateur	String

b) BanqueTiers

BanqueTiers	
NomTiers	String
Intitule	String
Commentaire	String
EstPrincipale	Bool
IdDevise	Short
NomPays	String
CalculIbanAuto	Bool
CodeBic	String
Banque	String
Guichet	String
NumeroCompte	String
Cle	String
Iban	String
Structure	StructureCompteBancaire
AgenceNom	String
AgenceAdresse	String
AgenceComplementAdresse	String
AgenceCodePostal	String
AgenceVille	String
AgenceRegion	String
AgenceNomPays	String
Index	Short
Createur	String
DateModification	DateTime
DateCreation	DateTime
UtilisateurCreateur	String

c) CompteAnalytique

CompteAnalytique	
IdPlanAnalytique	Short
Section	String
Intitule	String
TypeCompte	TypeCompte
Abrege	String
Raccourci	String
NbLignesSaut	Short
GenererReportANouveau	Bool
EstEnSommeil	Bool
NiveauAnalyse	Short
IdCollaborateur	Int
DateCreationAffaire	DateTime
DateAcceptationAffaire	DateTime
DateDebutAffaire	DateTime
DateFinAffaire	DateTime
ObjectifChiffreAffaireVente	Decimal
ObjectifChiffreAffaireAchat	Decimal
DomaineAffaire	DomaineAffaire
StatutAffaire	StatutAffaire
ModeFacturationAffaire	ModeFacturationAffaire
InfosLibres	ListInfoLibre
Createur	String
DateModification	DateTime
DateCreation	DateTime
UtilisateurCreateur	String

d) CompteBancaire

Banque	
Abrege	String
EstIntraGroupe	Bool
EstCompteDepotTresorPublic	Bool
EstCompteBanqueFrance	Bool
Devise	Short
NomPays	String
Structure	StructureCompteBancaire
CalculIbanAuto	Bool
idBanque	Int
Guichet	String
NumeroCompte	String
Banque	String
Cle	String
Iban	String
CodeBic	String
CodeJournalBanque	String
CodeJournalEscompte	String
CodeJournalEncaissement	String
AgenceNom	String
AgenceAdresse	String
AgenceComplementAdresse	String
AgenceCodePostal	String
AgenceVille	String
AgenceRegion	String
AgenceNomPays	String
Id	Int
Createur	String
DateModification	DateTime
DateCreation	DateTime
UtilisateurCreateur	String
EstEnSommeil	Bool

e) CompteGeneral

CompteGeneral	
NumeroCompte	String
Intitule	String
TypeCompte	TypeCompte
NatureCompte	TypeNatureCompte
ReportCompte	TypeReportCompte
EstAnalytique	Bool
EstEnSommeil	Bool
SaisirCompteTiers	Bool
SaisirDateEcheance	Bool
SaisirEnDevise	Bool
SaisirEnQuantite	Bool
InfosLibres	ListInfoLibre
Createur	String
DateModification	DateTime
LettrageEnSaisie	Bool
DateCreation	DateTime
UtilisateurCreateur	String

f) ContactBanque

ContactBanque	
Id	Int
IdBanque	Int
IdTypeContact	Short
IdServiceContact	Short
Civilite	Civilite
Nom	String
Prenom	String
Fonction	String
Telephone	String
Gsm	String
Fax	String
Email	String
Createur	String
DateModification	DateTime
DateCreation	DateTime
UtilisateurCreateur	String

g) DataMajModeReglement

DataMajModeReglement	
Criteria	Criteria
IdReglement	Short

h) Devise

Devise	
Id	Short
Intitule	String
Cours	Decimal
Monnaie	String
CodeIso	String
Sigle	String

i) Echeance

Echeance	
Id	Int
Numero	Int ?
IdDocument	Int
CompteGeneral	String
NumeroTiers	String
NumeroTiersPayeur	String
DateEcheance	DateTime
Montant	Decimal
Sens	Int
Etape	Int
EtatRemise	Int
DerniereEtapeComptable	Int
Reference	String
ReferenceBordereau	String
Libelle	String
NumeroCheque	String
VirementPrelevement	Int
FinDeMois	Int
IdDevise	Int
CodeModeleReglement	String
IdModeReglement	Int
PieceDocument	String
DateDocument	DateTime ?
DomaineDocument	Int ?
TypeDocument	Int ?
ReferenceDocument	String
Createur	String
DateCreation	DateTime
DateModification	DateTime
UtilisateurCreateur	String

j) Ecriture

Ecriture	
Id	Int
CodeJournal	String
NumeroPiece	String
DateEcriture	DateTime
DateSaisie	DateTime
DateEcheance	DateTime
NumeroFacture	String
Reference	String
Intitule	String
CompteGeneral	String

CompteGeneralContrepartie	String
NumeroTiers	String
NumeroTiersContrepartie	String
Montant	Decimal
Sens	SensEcriture
Parite	Decimal
MontantDevise	Decimal
IdDevise	Short
CodeTaxe	String
IdReglement	Short
EstLettree	Bool
Lettre	String
Createur	String
DateModification	DateTime
CessionCreance	CessionCreanceEcriture
InfosLibres	ListInfoLibre
EcrituresAnalytiques	List<EcritureAnalytique>
DateCreation	DateTime
UtilisateurCreateur	String
NumCloture	Int
ExtProvenance	ExternalProvenence
ExtSequence	Int
IdSessionSAC	String
Quantite	Decimal
CodeIsoDevise	String

k) EcritureAnalytique

EcritureAnalytique	
IdEcriture	Int
IdPlanAnalytique	Short
Section	String
Montant	Decimal
Createur	String
DateModification	DateTime
DateCreation	DateTime
UtilisateurCreateur	String

l) ElementsLettrageRequest

ElementsLettrageRequest	
Criteria	Criteria

m) ElementsLettrageResult

ElementsLettrageResult	
Results	List<LettrageElement>

n) InfoEcriture

InfoEcriture	
IdEcriture	Int
Lettrage	String
CodeJournal	String
JMDate	DateTime
NumeroPiece	String
NumeroTiers	String
NumeroCompte	String

o) Journal

Journal	
CodeJournal	String
Intitule	String
TypeJournal	TypeJournal
TypeNumerotationPiece	TypeNumerotationPieceJournal
CompteGeneral	String
EstAnalytique	Bool
EstEnSommeil	Bool
Createur	String
DateModification	DateTime
LettrageEnSaisie	Bool
DateCreation	DateTime
UtilisateurCreateur	String
EstProtege	Bool

p) LettrageElement

LettrageElement	
InfoEcritureInitial	InfoEcriture
EstPreLettree	Bool
EstLettree	Bool
InfoEcritureFinal	InfoEcriture

q) LienEcritureEcheance

LienEcritureEcheance	
Id	Int
IdEcheance	Int
IdEcriture	Int
MontantVentile	Decimal
EstAssociationOrigine	Bool
DateEcriture	DateTime
IntituleEcriture	String
CompteGeneral	String
NumeroTiers	String
CodeJournal	String
MontantEcriture	Decimal
SensEcriture	SensEcriture
Createur	String
DateCreation	DateTime
DateModification	DateTime
UtilisateurCreateur	String

r) MentionExonerationTaxe

MentionExonerationTaxe	
Id	Int
Intitule	String
CodeMotif	String
CodeCategorie	String
Createur	String
DateCreation	DateTime
UtilisateurCreateur	String

s) ModeReglement

ModeReglement	
Id	Short
Intitule	String
Code	String

t) Periode

Periode	
Debut	DateTime
Fin	DateTime
EstCloturee	Bool

u) PlanAnalytique

PlanAnalytique	
Id	Short
Intitule	String
EstObligatoire	Bool

v) StructureCompte

StructureCompte	
EstEdi	Bool
LongueurChampBanque	Ushort
ChampBanqueAlphaNumerique	Bool
LongueuChampGuichet	Ushort
ChampGuichetAlphaNumerique	Bool
LongueurChampNumeroCompte	Ushort
ChampNumeroCompteAlphaNumerique	Bool
LongueurchampCle	Ushort
ChampCleAlphanumerique	Bool
StructureCompteBancaire	StructureCompteBancaire

w) Taxe

Taxe	
Id	Int
CodeTaxe	String
Intitule	String
CodeRegroupement	String
TypeTaux	TypeTauxTaxe
TypeTaxe	TypeTaxe
Taux	Decimal
CompteGeneral	String
SensTaxe	SensTaxe
ProvenanceTaxe	TypeProvenanceTaxe
EstNonPercue	Bool
Createur	String
DateModification	DateTime
DateCreation	DateTime
UtilisateurCreateur	String
IdMotifExoneration	Int

x) UpdateModeReglementResult

UpdateModeReglementResult	
IsSuccess	Bool
NbEcrituresProcessed	Int
IdEcrituresUpdated	List<int>
IdEcrituresIgnored	List<int>
Messages	List<string>

4. Enumérations

a) AbonnementCalculEcheance

AbonnementCalculEcheance	
DateDocument	0
DebutPeriodicite	1
FinPeriodicite	2

b) AbonnementDateReference

AbonnementDateReference	
DateDebut	0
DateFin	1

c) AbonnementDureeDelaiPreavis

AbonnementDureeDelaiPreavis	
Jours	0
Semaine	1
Mois	2
Annee	3

d) AbonnementEtatPeriodicite

AbonnementEtatPeriodicite	
Generee	0
NonGeneree	1
Suspendue	2
Renouvellement	3

e) AbonnementProrataLigne

AbonnementProrataLigne	
Aucun	0
Quantite	1
Valeur	2
QuantiteTotale	3
ValeurTotale	4

f) AbonnementReconduction

AbonnementReconduction	
NonDefini	-1
Aucune	0
Tacite	1
AConfirmer	2

g) AbonnementSourceContact

AbonnementSourceContact	
Modele	0
Aucun	1

h) AbonnementSourceGeneration

AbonnementSourceGeneration	
Modele	0
FicheClient	1

i) AbonnementType

AbonnementType	
Proposition	0
Abonnement	1
Resiliation	2
Modele	3

j) AbonnementTypeDuree

AbonnementTypeDuree	
Jours	0
Semaine	1
Mois	2
Annee	3
MoisCivil	4
AnneeCivile	5

k) AbonnementTypePeriodicite

AbonnementTypePeriodicite	
Jours	0
Semaine	1
Mois	2
Annee	3
MoisCivil	4
AnneeCivile	5

l) AbonnementTypeTiers

AbonnementTypeTiers	
Client	0
Fournisseur	1
Interne	2

m) ActionRisque

ActionRisque	
Alivrer	0
ASurveiller	1
ABloquer	2

n) CessionCreanceEcriture

CessionCreanceEcriture	
NonCedee	0
Emise	1
Acceptee	2
Refusee	3
Comptabilisee	4
Cession	5

o) ChampsDocumentUpdate

ChampsDocumentUpdate	
Statut	0
DateLivraison	1
Reference	2
Affaire	3
Representant	4
Expedition	5
Entete1	6
Entete2	7
Entete3	8
Entete4	9
Depot	10
Escompte	11

p) Civilite

Civilite	
M	0
Mme	1
Mlle	2

q) ComparisonOperator

ComparisonOperator	
Equals	0
GreaterThan	1
LessThan	2
GreaterThanOrEqualsTo	3
LessThanOrEqualsTo	4
NotEqualTo	5
NotGreaterThan	6
NotLessThan	7
Like	8

NotLike	9

r) DomaineAffaire

DomaineAffaire	
LesDeux	0
Ventes	1
Achats	2

s) DomaineDocument

DomaineDocument	
Vente	0
Achat	1
Stock	2
DocumentInterne	3

t) DbType

Les données contenues dans Sage sont principalement des types suivants :

- Texte => String
- Montant, Quantité, Prix => Decimal
- Valeur booléenne => Int16 (avec 0 = faux et 1 = vrai)
- Valeur d'indice => Int32

DbType	
AnsiString	0
Binary	1
Byte	2
Boolean	3
Currency	4
Date	5
DateTime	6
Decimal	7
Double	8
Guid	9
Int16	10
Int32	11
Int64	12
Object	13
SByte	14
Single	15
String	16
Time	17
UInt16	18
UInt32	19
UInt64	20
VarNumeric	21
AnsiStringFixedLength	22
StringFixedLength	23
Xml	25
DateTime2	26
DateTimeOffset	27

u) FactureElectroniqueAssujettissement

FactureElectroniqueAssujettissement	
Assujetti	0
ParticulierOuNonAssujettiTVA	1

v) FactureElectroniqueAutreIdentifiantType

FactureElectroniqueAutreIdentifiantType	
Aucun	0
UnionEuropeenneHorsFrance	1
HorsUnionEuropeenne	2
RIDET	3
TAHITI	4
Particulier	5
Autre	6

w) FactureElectroniqueControleEmission

FactureElectroniqueControleEmission	
Aucun	0
CodeServiceObligatoire	1
ReferenceEngagementObligatoire	2
CodeServiceEtReferenceEngagementObligatoire	3
CodeServiceOuReferenceEngagementObligatoire	4

x) FactureElectroniqueTypeEntite

FactureElectroniqueTypeEntite	
NonReferenceAnnuaire	0
PriveeAssujettieTVAFrance	1
Publique	2

y) LangueSage

LangueSage	
Aucune	0
Langue1	1
Langue2	2

z) LocalisationPays

LocalisationPays	
UnionEuropeenne	0
France	1
HorsUnionEuropeenne	2
DROM	3
COM	4

aa) LogicalOperator

LogicalOperator	
And	0
Or	1

bb) ModeControleEnCours

ModeControleEnCours	
Automatique	0
SelonCodeRisque	1
CompteBloque	2

cc) ModeFacturationAffaire

ModeFacturationAffaire	
Forfaitaire	0
Avancement	1

dd) OrderType

OrderType	
Asc	0
Desc	1

ee) ParameterDirection

ParameterDirection	
Input	1
Output	2
InputOutput	3
ReturnValue	6

ff) PremierJourSemaine

PremierJourSemaine	
Lundi	0
Samedi	1
Dimanche	2

gg) PremiereSemaineAnnee

PremierJourSemaineAnnee	
CommencePremierJanvier	0
PRemiereSemaine4Jours	1
PremiereSemaineEntiere	2

hh) PeriodiciteLigne

PeriodiciteLigne		
Toutes	0	
Unique	1	
Date	2	
Jour	3	Valable à partir de la 100C V6
Semaine	4	Valable à partir de la 100C V6
Mois	5	Valable à partir de la 100C V6

ii) PrioriteDestockage

PrioriteDestockage	
EmplacementNonPrincipal	0
EmplacementPrincipal	1
ZoneEmplacement	2

jj) ProvenanceDocument

ProvenanceDocument	
Normale	0
FactureDeRetour	1
FactureAvoir	2
Ticket	3
Rectificatif	4

kk) ReconductionLigne

ReconductionLigne	
NonDefini	-1
Non	0
Oui	1

ll) RemiseTypeLigne

RemiseTypeLigne	
Montant	0
Pourcentage	1
Quantite	2

mm) SensEcriture

SensEcriture	
Debit	0
Credit	1

nn) SensTaxe

SensTaxe	
Deductible	0
Collecte	1

oo) StatutAffaire

StatutAffaire	
Proposition	0
Accepte	1
Perdu	2
Encours	3
EnAttente	4
Termine	5

pp) StatutDocument

StatutDocument	
Saisie	0
Confirme	1
Accepte	2

qq) StructureCompteBancaire

StructureCompteBancaire	
Locale	0
Autre	1
BBAN	2
IBAN	3

rr) TypeArticle

TypeArticle	
Standard	0
Gamme	1

ss) TypeCodeEDI

TypeCodeEDI	
GLN	0
DUNS	1
Autre	2

tt) TypeCodeRoutage

TypeCodeRoutage	
CodeService	0
GLN	1
ODETTE	2
SWIFT	3
Autre	4

uu) TypeCompte

TypeCompte	
Detail	0
Total	1

vv) TypeDocument

TypeDocument	
DevisVente	0
BonDeCommandeVente	1
PreparationDeLivraisonVente	2
BonDeLivraisonVente	3
BonDeRetourVente	4
BonAvoirVente	5
FactureVente	6
FactureComptabiliseeVente	7
DemandeAchat	10
PreparationDeCommandeAchat	11
BonDeCommandeAchat	12
BonDeLivraisonAchat	13
BonDeRetourAchat	14
BonAvoirAchat	15
FactureAchat	16
FactureComptabiliseeAchat	17
MouvementEntreeStock	20
MouvementSortieStock	21
DepreciationDeStock	22
VirementDeDepotDepotStock	23
PreparationDeFabricationStock	24
OrdreDeFabricationStock	25
BonDeFabricationStock	26
DocumentInterne1	40
DocumentInterne2	41
DocumentInterne3	42
DocumentInterne4	43
DocumentInterne5	44
DocumentInterne6	45
SaisieDuRealise	46

ww) TypeEmplacement

TypeEmplacement	
Standard	0
Transit	1
Controle	2

xx) TypeEscompte

TypeEscompte	
Soumis	0
NonSoumis	1

yy) TypeFamille

TypeFamille	
Detail	0
Total	1
Centralisateur	2

zz) TypeFrais

TypeFrais	
Montant	0
Pourcentage	1
Unite	2

aaa) TypeFraisModeExpedition

TypeFraisModeExpedition	
MontantForfaitaire	0
Quantite	1
PoidsNet	2
PoidsBrut	3

bbb) TypeFrancoModeExpedition

TypeFrancoModeExpedition	
MontantForfaitaire	0
Quantite	1

ccc) TypeInfoLibre

TypeInfoLibre	
String	0
SmallDate	1
LongDate	2
Decimal	3

ddd) TypeJournal

TypeJournal	
Achat	0
Vente	1
Tresorerie	2
General	3
Situation	4

eee) TypeMouvement

TypeMouvement	
Aucun	0
EntreeQuantite	1
EntreeFrancs	2
SortieQuantite	3
SortieFrancs	4

fff) TypeNatureCompte

TypeNatureCompte	
Aucun	0
Client	1
Fournisseur	2
Salarie	3
Banque	4
Caisse	5
AmortissementProvision	6
ResultatBilan	7
Charge	8
Produit	9
ResultatGestion	10
Immobilisation	11
Capitaux	12
Stock	13
Titre	14

ggg) TypeNomenclature

TypeNomenclature	
Aucune	0
Fabrication	1
CommercialeCompose	2
CommercialComposant	3
ArticleLie	4

hhh) TypeNumerotationPieceJournal

TypeNumerotationPieceJournal	
Manuelle	0
ContinueJournal	1
ContinueFichier	2
Mensuelle	3

iii) TypeNumerotationTiers

TypeNumerotationTiers	
Manuelle	0
Automatique	1
Racine	2

jjj) TypePrix

TypePrix	
HT	0
TTC	1

kkk) TypeParametrageComptePrincipal

TypeParametrageComptePrincipal	
Radical	0
Compte	1

lll) TypeProvenanceTaxe

TypeProvenanceTaxe	
Nationale	0
Intracommunautaire	1
Export	2
Divers1	3
Divers2	4
Divers3	5
Divers4	6
Divers5	7

mmm) TypeRemise

TypeRemise	
Remise	0
HorsRemise	1

nnn) TypeRepartitionReglement

TypeRepartitionReglement	
Pourcentage	0
Equilibre	1
Montant	2

ooo) TypeReportCompte

TypeReportCompte	
Aucun	0
Solde	1
Detail	2

ppp) TypeSuiviStock

TypeSuiviStock	
Aucun	0
Serialise	1
CMUP	2
FIFO	3
LIFO	4
ParLot	5

qqq) TypeTiers

TypeTiers	
Client	0
Fournisseur	1
Salarie	2
Autre	3

rrr) TypeTarificationFournisseur

TypeTarificationFournisseur	
Quantite	0
Montant	1
PrixNet	2

sss) TypeTarificationQuantiteMontant

TypeTarificationQuantiteMontant	
SimpleHorsRemise	0
Quantite	1
Montant	2
PrixNet	3

ttt) TypeTauxTaxe

TypeTauxTaxe	
Taux	0
Montant	1
Quantite	2

uuu) TypeTaxe

TypeTaxe	
TVADebits	0
TVAEncaissements	1
TaxeParafiscaleHT	2
TaxeParafiscaleTTC	3
TaxeParafiscalePoids	4
TVACEE	5
Surtaxe	6
IRPF	7
Agraire	8
IGIC	9

vvv) UnitePoids

UnitePoids	
Tonne	0
Quintal	1
Kilogramme	2
Gramme	3
Milligramme	4

www) ZoneEmplacement

ZoneEmplacement	
Aucun	0
A	1
B	2
C	3

xxx) TypeConditionReglement

TypeConditionReglement	
JourNet	0
FinMoisCivil	1
FinMois	2

yyy) TypeRepartitionReglement

TypeConditionReglement	
Pourcentage	0
Equilibre	1
Montant	2