Product Engineering
Communication Product Engineering

Le :	Vendredi 13 novembre 2020
De :	Sage Product Engineering
Objet :	**Sage 100c Objets métiers version 7.20**

Nouveautés

Les modifications et nouveautés implémentées pour Sage 100c Objets métiers version 7.20 portent sur les points suivants :

- Modifications des Traitements
- Modifications des interfaces
- Modifications processus

La mise à jour vers les versions 7.00 des applications Sage 100c ne requièrent pas de conversion de base de données.

L'ajout des nouvelles propriétés a nécessité la modification de certaines interfaces. Du fait de ces modifications, les développements spécifiques, compilés (exécutables .net par exemple), réalisés avec une version précédente des Objets Métiers 100c (<= v7.20), et qui ne référencent pas explicitement une version de l'activeX des Objets métiers 100c (n'utilisent pas de fichier manifest), devront être recompilés avec la version 7.20 des Objets métiers 100c pour être compatibles avec cette version.

Modifications Traitement

La sortie de stock en quantité, ainsi que la sortie des emplacements ne testent plus une sortie négative dans le cas où le stock est **déjà négatif**. Cela peut se produire lorsque l'on coche et décoche l'option dans la Gescom : « Gérer le stock négatif ».

Modifications interfaces

ILicence

Ajout de la Propriété

Accès	Syntaxe	Description
Lecture	Version ()	Donne le Numéro de version de la DLL : "7.20"

Exemple C#

```
BSCIALApplication100c BaseCial
MessageBox.Show( BaseCial.Licence.Version , "Version de la dll");
```

IBODocumentLigne3

Ajout des Propriétés

Accès	Syntaxe	Description
Ecriture	SetEmplacementEntree(As IBODepotEmplacement)	Permet de saisir un emplacement d'entrée différent de l'emplacement par défaut
Ecriture	WriteCumulLot ()	Ecriture de la ligne avec la particularité de cumuler la ligne de lot en entrée s'il existe déjà : même document, article, dépôt, N° lot, complément et emplacement.

Exemple C#

Cet exemple cumule une ligne de lots d'entrée si le processus trouve pour cet article : le même numéro de série, le même emplacement (optionnel), le même complement (optionnel) et que le lot n'est pas entamé.SInon une nouvelle ligne est créée par un WriteDefault.

```
void ProcessEntrerLots(ref BSCIALApplication100c BaseCial, string sNoSerie, string sAr_Ref, int Qte, DateTime datePeremption)
{
        try
        {
          IBODocumentStock3 pcBoc = BaseCial.FactoryDocumentStock.ReadPiece (DocumentType.DocumentTypeStockFabrication, "BF00002");

          IBODocumentStockLigne3 pLigne2 = (IBODocumentStockLigne3) pcBoc.FactoryDocumentLigne.Create();
          IBOArticle3 pcBOArticle = BaseCial.FactoryArticle.ReadReference(sAr_Ref);
           IBODepot3 pDepot = BaseCial.FactoryDepot.ReadIntitule("Bijou SA");
           IBODepotEmplacement pDepotEmpl = pDepot.FactoryDepotEmplacement.ReadCode("SECTIONBC5");
           IBOArticleDepot3 pArtDepot = pcBOArticle.FactoryArticleDepot.ReadDepot(pDepot);
           IBOArticleDepotLot pArtDepotLot = (IBOArticleDepotLot)pArtDepot.FactoryArticleDepotLot.Create();
           pArtDepotLot.DateFabrication = pLigne2.DO_Date;
           pArtDepotLot.DatePeremption = datePeremption;
           pArtDepotLot.NoSerie = sNoSerie;
           pArtDepotLot.Complement = "COMPLEMENT02";

           pLigne2.SetDefaultLot(pArtDepotLot, Qte);
           pLigne2.SetDefaultArticle(pcBOArticle, Qte);
// Le dépôt de l'article doit déjà être affecté parce que cette méthode teste si c'est le même que celui de l'emplacement
           pLigne2.SetEmplacementEntree(pDepotEmpl);
           pLigne2.WriteCumulLot();
        }
        catch (Exception ex)
        {
          MessageBox.Show(ex.Message, "Resultat Entrer Lot");
        }
}
```

Modifications interfaces

Il est maintenant possible de créer une structure d'infos libres pour tous les fichiers qui le gèrent. Le créateur de ces champs est 'COBJ'.

eTypeInfoLibre

Nouveau type d'énumérés : Liste de tous les types d'infos libres que l'on peut créer dans les structures d'infos libres.

```
enum eTypeInfoLibre
 {
    eTIL_DATE,
    eTIL_DOUBLE,
    eTIL_CSTR,
    eTIL_LDATE ,
    eTIL_MONTANT,
    eTIL_TABLE
 }
```

IBODocumentFactory
IBODocumentLigneAllFactory
IBOTiersFactory3
IBOCompteGFactory3
IBOCompteAFactory3
IBOEcritureFactory3
IBOArticleFactory3
IBOArticleDepotLotAllFactory
IBIRessourceFactory

Pour chacune de ces interfaces : Ajout de la Propriété CreateInfoLibre

Accès	Syntaxe	Description
Ecriture	CreateInfoLibre (String Nom, eTypeInfoLibre Type, short lenght)	Permet de créer une entrée dans la structure Infos Libre du fichier. Si le nom existe déjà : elle est modifiée si l'info libre avait été créée par les OM au préalable.

Exemple C#

Cet exemple crée une structure d'infos libres pour chaque fichier.

```
void CreateInfoLibre(ref BSCPTAApplication100c BaseCpta, ref BSCIALApplication100c BaseCial)
{
    try
    {
        BaseCpta.FactoryCompteG.CreateInfoLibre("Info Compte Général", eTypeInfoLibre.eTIL_CSTR, 15);
        BaseCpta.FactoryCompteA.CreateInfoLibre("Info Compte Analytique", eTypeInfoLibre.eTIL_CSTR, 25);
        BaseCpta.FactoryEcriture.CreateInfoLibre("Info Ecriture Comptable", eTypeInfoLibre.eTIL_CSTR, 35);
        BaseCpta.FactoryTiers.CreateInfoLibre("Info Tiers Comptable", eTypeInfoLibre.eTIL_CSTR, 45);

        BaseCial.FactoryDocument.CreateInfoLibre("Info Document", eTypeInfoLibre.eTIL_MONTANT, 0);
        BaseCial.FactoryDocumentLigne.CreateInfoLibre("Info Ligne", eTypeInfoLibre.eTIL_DOUBLE, 0);
        BaseCial.FactoryArticle.CreateInfoLibre("Info Article", eTypeInfoLibre.eTIL_DATE, 0);
        BaseCial.FactoryArticleDepotLot.CreateInfoLibre("Info Lots", eTypeInfoLibre.eTIL_CSTR, 15);
        BaseCial.FactoryRessourceBase.CreateInfoLibre("Info Ressource", eTypeInfoLibre.eTIL_LDATE, 0);
    }
    catch (Exception ex)
    {
          MessageBox.Show(ex.Message, "Resultat Errors");
    }
}
```

IBOCollaborateur Modification

Collaborateur. Ajout d'une propriété : Financier

Lecture / Ecriture	Financier () As Boolean	Vendeur (True) ou non (False).

BSCPTAApplication100c Modification

Ajout des propriétés :

Lecture	FactoryBonAPayer () As IBPBonAPayer	Donne le factory de l'objet IBPBonAPayer
Process	CreateProcess_BAPValider(as IBOCollaborateur) as IPMBonAPayer	Crée un processus permettant de Valider un Bon à Payer
Process	CreateProcess_BAPAttendre (as IBOCollaborateur) as IPMBonAPayer	Crée un processus permettant de mettre en attente un Bon à Payer
Process	CreateProcess_BAPPayer (as IBOCollaborateur) as IPMBonAPayer	Crée un processus permettant de Payer un Bon à Payer
Process	CreateProcess_BAPConformer (as IBOCollaborateur) as IPMBonAPayer	Crée un processus permettant de mettre Conforme un Bon à Payer
Process	CreateProcess_BAPRejeter (as IBOCollaborateur) as IPMBonAPayer	Crée un processus permettant de rejeter un Bon à Payer

IBPBonAPayer Nouveau

Paramètres de comptabilisation – Gestion du Bon à payer (Paramètres société / Comptabilisation)

Propriétés

Accès	Syntaxe	Description
Lecture / Ecriture	Autorisation () As eBAPAutorisationType	Niveau de validation.
Lecture / Ecriture	Responsable() As IBOCollaborateur	Responsable financier.
Lecture / Ecriture	Facture () As eBAPAValider	Facture à Valider
Lecture / Ecriture	SeuilValidation () As double	Seuil de validation.

IBonAPayer Nouveau

Historique du Bon à payer pour une pièce d'écriture

Propriétés

Accès	Syntaxe	Description
Lecture	Statut () As eTypeBAP	Statut du Bon à payer.
Lecture	Responsable() As IBOCollaborateur	Collaborateur du Bon à payer.
Lecture	DateAction () As Date	Date du Bon à payer.
Lecture	Commentaire () As String	Commentaire
Lecture	Application () As integer	Application à l'origine du Bon.
Lecture	Retour () As integer	Retour fournisseur.

Ajout des énumérateurs

eTypeBAP

Statut Bon à payer.

Syntaxe	Description
Aucun	Aucun
AValider	A Valider
NonConforme	Non Conforme
Conforme	Conforme
EnAttente	En Attente
BonAPayer	Bon A Payer

eBAPAutorisationType

Niveau de validation.

Syntaxe	Description
BAP_TypeAucun	Aucun
BAP_Financier	Responsable financier uniquement
BAP_LesDeux	Acheteur et Responsable financier

eBAPAValider

Factures à valider.

Syntaxe	Description
BAP_ToutesFactures	Toutes les Factures
BAP_SelonMontant	Selon Montant

Ajout des Processus

IPMBonAPayer

Description

Ce processus permet d'affecter un bon à payer à une écriture.
Il doit être decliné en CreateProcess_BAPValider, CreateProcess_BAPAttendre, CreateProcess_BAPPayer, CreateProcess_BAPConformer ou CreateProcess_BAPRejeter.

Interface héritée

Syntaxe	Description
IPMProcess	Cf. Interface IPMProcess pour les propriétés et méthodes héritées.

Ajout des Propriétés

Accès	Syntaxe	Description
Ecriture	Ecriture (As IBOEcriture3)	Permet d'affecter l'écriture sur laquelle le Bon à payer sera affecté
Lecture / Ecriture	Complement (As String)	Complément du Bon à payer.

Exemple

```
private void TestBonAPayer(ref BSCPTAApplication100c BaseCpta)
{
   try
   {
      IBOCollaborateur cCollabo = BaseCpta.FactoryCollaborateur.ReadNomPrenom("PLATY", "Nelly");
      cCollabo.Financier = true;
      cCollabo.Write();
      IBPBonAPayer bonAPayer = (IBPBonAPayer) BaseCpta.FactoryBonAPayer.List[1];
      bonAPayer.Responsable = cCollabo;
      bonAPayer.Autorisation = eBAPAutorisationType.BAP_Financier;
      bonAPayer.Facture = eBAPAValider.BAP_SelonMontant;
      bonAPayer.SeuilValidation = 999;
      bonAPayer.Write();

      short EC_No = 65;

      if (BaseCpta.FactoryEcriture.ExistNumero(EC_No))
      {
         IBOEcriture3 cEcrit = BaseCpta.FactoryEcriture.ReadNumero(EC_No);
         IPMBonAPayer pBonAPayer = BaseCpta.CreateProcess_BAPPayer(cCollabo);
         pBonAPayer.Commentaire = "Ok pour Payer";
         pBonAPayer.Ecriture = cEcrit;
         pBonAPayer.Process();

         IBOEcriture3 cEcrit2 = BaseCpta.FactoryEcriture.ReadNumero(EC_No);
         IBonAPayer BonAPayer = cEcrit2.HistoriqueBonAPayer;
         MessageBox.Show(BonAPayer.Collaborateur.Nom +" "+ BonAPayer.Commentaire, "Bon à Payer");
      }
   }
   catch (Exception ex)
   {
      MessageBox.Show(ex.Message, "Resultat Errors");
   }
}
```

Modifications des Processus

IPMDocTransferer

Ajout des Propriétés

Accès	Syntaxe	Description
Ecriture	ComplementSerie (sName As String)	Permet de retrouver en **priorité** les lots suivant un complément donné. Agit comme un filtre.
Lecture	ComplementSerie () As String	Lecture du complément utilisé comme filtre prioritaire.
Ecriture	Transactional (bLock As Boolean)	Option qui permet de passer le processus en mode transaction sécurisée
Lecture	Transactional () As Boolean	Option qui permet de passer le processus en mode transaction sécurisée

IPMSortirLots

Ajout des Propriétés

Accès	Syntaxe	Description
Ecriture	ComplementSerie (sName As String)	Permet de retrouver en **priorité** les lots suivant un complément donné. Agit comme un filtre.
Lecture	ComplementSerie () As String	Lecture du complément utilisé comme filtre prioritaire.
Ecriture	Transactional (bLock As Boolean)	Option qui permet de passer le processus en mode transaction sécurisée, Vraie par défaut.
Lecture	Transactional () As Boolean	Option qui permet de passer le processus en mode transaction sécurisée

IPMPreleverLots

Ajout des Propriétés

Accès	Syntaxe	Description
Ecriture	Transactional (bLock As Boolean)	Option qui permet de passer le processus en mode transaction sécurisée, Vraie par défaut.
Lecture	Transactional () As Boolean	Option qui permet de passer le processus en mode transaction sécurisée

Exemple C#

Cet exemple fait une sortie du lot “FRVA0001” en **sortant en priorité** les lots dont le complement est “COMPLEMENT2”.
A Noter que l’option Transactional est mise à True, c’est l’option **par défaut** : Ainsi, en cas d’erreur, le processus pourra assurer l’intégrité des données.

```
void ProcessSortirLots(ref BSCIALApplication100c BaseCial, string sPiece, string
sAr_Ref, int Qte)
{
   IPMSortirLots SortieLot = BaseCial.CreateProcess_SortirLots();
   try
   {
      SortieLot.Transactional = false;
      IBODocumentStock3 pcBODocument2 = BaseCial.FactoryDocumentStock.ReadPiece
   (DocumentType.DocumentTypeStockFabrication, "BF00003");
      IBODocumentStockLigne3 pLigne2 = (IBODocumentStockLigne3)
pcBODocument2.FactoryDocumentLigne.Create();
      IBOArticle3 pcBOArticle = BaseCial.FactoryArticle.ReadReference(sAr_Ref);
      IBODepot3 pDepot = BaseCial.FactoryDepot.ReadIntitule("Bijou SA");
      IBODepotEmplacement pDepotEmpl = null;
      if(sPiece.Length > 0)
        pDepotEmpl  = pDepot.FactoryDepotEmplacement.ReadCode(sPiece);
      pLigne2.SetDefaultArticle(pcBOArticle, Qte);
      SortieLot.ComplementSerie = "COMPLEMENT02";
      SortieLot.SetLigneDefaultLot(pLigne2, "FRVA0001", pDepotEmpl);
      SortieLot.Process();
   }
   catch (Exception ex)
   {
      string sResult = ex.Message;
      foreach (IFailInfo erreur in SortieLot.Errors)
        sResult = sResult + " " + erreur.Text;
      MessageBox.Show(sResult, "Resultat Sortie Lot");
   }
}
```