Dossier de développement

30/06/2015

Application réalisée par

tel :
fax :

Partie 1

En-tête

Code soource Scénario Conférence

Partie 2

Projet

Projet

Code

Initialisation de TDT

GLOBAL

```
cnxMySQL sont des Connexions

gbConnected est un boolean

gnNBM, gnNBB,gnNBN sont des entiers
gbModeEPCIS est un boolean = True
gbModeAutorisation est un boolean = True
 gnCpt est un entier
gsMsgTCP est une chaîne
gbCNxGGL est un boolean = False
```

Projet

Statistiques sur le code

Lignes	%	Lig./tr	
266	44	29	COL_ProcéduresGlobales
0	0	0	FEN_CodeBarre
26	4	8	FEN_TDT1
189	33	17	FEN_Test
634	31	31	SITUATION
26	0	26	TDT
17	0	17	ETAT_cb
16	0	0	ETAT_ean128
16	0	0	ETAT_ean128_hexa

Lignes	%	Lig./tr
1116	34	24

Lignes: Nombre total de lignes de code.
Lig./trait.: Nombre de lignes de code par traitement.
% comm.: Pourcentage de commentaires dans le code.

Partie 3

Fenêtre WinDev

FEN_Test

Code des champs

Clic sur BTN_SansNom1

```
i est un entier
CB est une chaîne


FOR i = 1 TO 4
    CB = String__Complete (NumériqueVersChaîne(i),"0",5,True)
    cb = "Test SW le 25/08/2014 15:02"
    Create_ZPLFile("Smartwear.txt",CB)
    Print_ZPLFile( "172.21.21.169","Smartwear.txt")
END
```

Clic sur BTN_SansNom2

```
HSupprimeTout(ean128)
nFic est un entier = fOuvre("C:\data\ean128.txt",foLectureEcriture)
sLigne est une chaîne


IF nfic > 0 THEN
    sligne = fLitLigne(nfic)
    TANTQUE sligne <> EOT
        sligne = Caract_Delete(sligne,"(")
        sligne = Caract_Delete(sligne,")")
        HRAZ(ean128)
        ean128.Code=sligne
        HAjoute(ean128)
        sLigne = fLitLigne(nFic)
    FIN
    fFerme(nfic)
END
Info ("fin")
```

Clic sur BTN_SansNom3

```
sCléMap est une chaîne = "clé google maps"
```

```
Localisation est un gglCoordonnée
adresse est une chaîne ="rue de péronnes 13 7640 Antoing Belgique"
___GglAdresseVersCoordonnées(adresse)
Info( Localisation..Latitude+CR+Localisation..Longitude)

//http://maps.googleapis.com/maps/api/geocode/xml?address="+rue+de+péronnes+13+7640+Antoing+Belgique+ "&sensor=false
```

Clic sur BTN_SansNom4

```
url est une string = "http://192.168.14.3:8080/epcis-repository-0.5.0/capture"
messages est une string
sLigne est une string

nFic est un entier = fOuvre("C:\data\XML_Capture.txt",foLectureEcriture)
IF nfic > 0 THEN
    sligne = fLitLigne(nfic)
    TANTQUE sligne <> EOT
        messages += sligne +CR
        sLigne = fLitLigne(nFic)
    FIN
    fFerme (nfic)

END
SAI_Http = messages
HTTPRequête(url,"","",messages)
sai_http= HTTPDonneRésultat()
```

Clic sur BTN_SansNom5

```
sChaineXml est une chaîne
sUser,sPassword sont chaîne
url est une string = "http://192.168.14.3:8080/epcis-repository-0.5.0/capture"
sUser = "Login"
sPassword = "MotDePasse"


//XMLDocument("XmlDoc")
//XMLEcrit("XmlDoc","/Donnee","1233")
sChaineXml = fChargeTexte("C:\data\XML_Capture.txt")//XMLConstruitChaîne("XmlDoc")
//XMLTermine("XmlDoc")
HTTPCréeFormulaire("POSTDATA")
HTTPAjouteParamètre( "POSTDATA", "", sChaineXml)
// en enfin on renvoie les données avec le shipping Nbr modifié
bRes est un boolean  = HTTPEnvoieFormulaire( "POSTDATA", url, httpPost, "", "", "text/xml")
SI bRes = Vrai ALORS
    Info(HTTPDonneRésultat(httpEntête))
FIN
```

Clic sur BTN_SansNom6

```
HLitPremier(ean128,IDean128)
TANTQUE PAS HEnDehors
    //svalret = Code128_To_SGTIN96__Convert(ean128.Code)
    ean128.compl="("+Milieu(ean128.Code,1,2)+")"+Milieu(ean128.Code,3,14)+"("+Milieu(ean128.Code,17,2)+")"+Milieu(ean128.Code,19,12)
    HModifie(ean128)
    HLitSuivant(ean128,IDean128)

FIN
```

Clic sur BTN_SansNom7

```
sValRet est une chaîne

HLitPremier(ean128,IDean128)
TANTQUE PAS HEnDehors
    //svalret = Code128_To_SGTIN96__Convert(ean128.Code)
    svalret = Code128_To_SGTIN96__Convert(ean128.compl)
    ean128.epc = ExtraitChaîne(svalret,1,";")
    ean128.HEXA=Binaire_to_Hexa_Convert(ExtraitChaîne(svalret,3,";"))
    HModifie(ean128)
    HLitSuivant(ean128,IDean128)

FIN
```

Clic sur BTN_SansNom8

```
Connexions_serveurs(1)
IF gbConnected THEN
HSupprimeTout (event_objectevent)
HSupprimeTout(event_objectevent_epcs)
END

Connexions_serveurs(2)
IF gbConnected THEN
    HSupprimeTout (event_objectevent)
    HSupprimeTout(event_objectevent_epcs)
END
Connexions_serveurs(3)
IF gbConnected THEN
    HSupprimeTout (event_objectevent)
    HSupprimeTout(event_objectevent_epcs)
END
```

FEN_Test

Procédures

Procédure locale ___gglAdresseVersCoordonnées

```
// Résumé : <indiquez ici ce que fait la procédure>
// Syntaxe :
//[ <Résultat> = ] ___gglAdresseVersCoordonnées (<sAdresse> est chaîne)
//
// Paramètres :
//  sAdresse (chaîne) : <indiquez ici le rôle de sAdresse>
// Valeur de retour :
//  gglCoordonnée : //  Aucune
//
// Exemple :
// Indiquez ici un exemple d'utilisation.
//

PROCEDURE ___gglAdresseVersCoordonnées(LOCAL sAdresse est une chaîne )

sUrlgglMaps est une chaîne
xmlGgl est un xmlDocument
localisation est un gglCoordonnée

// Adresse de la nouvelle API
sUrlgglMaps="http://maps.googleapis.com/maps/api/geocode/xml?address="+sAdresse+"&sensor=false"
sai_ggl = sUrlgglMaps
// Exécution de la requête
SI HTTPRequête(sUrlgglMaps) = Vrai ALORS

    // l'API retourne un document xml
    sret est une chaîne = HTTPDonneRésultat()
    xmlGgl = XMLOuvre(sret,depuisChaîne)
    localisation.Latitude = xmlGgl.GeocodeResponse.RESULT.geometry.location.lat..Texte
    localisation.Longitude = xmlGgl.GeocodeResponse.RESULT.geometry.location.lng..Texte

FIN

RENVOYER localisation
```

Procédure locale __gglAdresseVersCoordonnées

```
// Résumé : <indiquez ici ce que fait la procédure>
// Syntaxe :
//[ <Résultat> = ] __gglAdresseVersCoordonnées (<sAdresse> est chaîne)
//
// Paramètres :
//  sAdresse (chaîne) : <indiquez ici le rôle de sAdresse>
// Valeur de retour :
//  gglCoordonnée : //  Aucune
//
// Exemple :
// Indiquez ici un exemple d'utilisation.
//


PROCEDURE __gglAdresseVersCoordonnées(LOCAL sAdresse est une chaîne )

sUrlgglMaps est une chaîne
xmlGgl est un xmlDocument
localisation est un gglCoordonnée


//sadresse = oemversansi(sadresse)
// Adresse de la nouvelle API
sUrlgglMaps="http://maps.googleapis.com/maps/api/geocode/xml?address="+sAdresse+"&sensor=false"
sUrlgglMaps="http://maps.googleapis.com/maps/api/geocode/xml?address=rue de péronnes 13 7640 Antoing Belgique&sensor=false"
sai_ggl = surlgglmaps
// Exécution de la requête
SI HTTPRequête(sUrlgglMaps) = Vrai ALORS

    // l'API retourne un document xml
    sRet est une chaîne = HTTPDonneRésultat()
//  SAI_Http = sret
    xmlGgl = XMLOuvre(HTTPDonneRésultat(),depuisChaîne)

    localisation.Latitude = xmlGgl.GeocodeResponse.RESULT.geometry.location.lat..Texte
    localisation.Longitude = xmlGgl.GeocodeResponse.RESULT.geometry.location.lng..Texte

FIN

RENVOYER localisation
```

Procédure locale _GglAdresseVersCoordonnées

```
// Résumé : <indiquez ici ce que fait la procédure>
// Syntaxe :
//_GglAdresseVersCoordonnées (<P_adresse> est chaîne)
//
// Paramètres :
//  P_adresse (chaîne) : <indiquez ici le rôle de P_adresse>
```

```
// Valeur de retour :
//  Aucune
//
// Exemple :
// Indiquez ici un exemple d'utilisation.
//


PROCEDURE _GglAdresseVersCoordonnées(LOCAL P_adresse est une chaîne)
localisation est un gglCoordonnée
P_adresse= Remplace(P_adresse," ","+")

sUrlRequete est une chaîne = "http://maps.googleapis.com/maps/api/geocode/xml?address="+Caract(34)+"+"+ P_adresse +"+"+Caract(34)+"&sensor=false"
//sUrlRequete
="http://maps.googleapis.com/maps/api/geocode/xml?address="+Caract(34)+Caract(34)+"+"+p_adresse+"+"+Caract(34)+Caract(34)+"&sensor=false"


///sUrlRequete = "http://maps.googleapis.com/maps/api/geocode/xml?address=" + P_adresse + "&sensor=false"
sUrlRequete = "http://maps.googleapis.com/maps/api/geocode/xml?address=rue  de péronnes 13 7640  Antoing Belgique&sensor=false"
//surlrequete ="http://www.pcsoft.fr"
s_ret est une chaîne
SI HTTPRequête(sUrlRequete) = Vrai ALORS

    s_ret = HTTPDonneRésultat()
    SAI_Http=s_ret
    localisation.Latitude=XMLExtraitChaîne(s_ret ,"lat")
    localisation.Longitude=XMLExtraitChaîne(s_ret ,"lng")
FIN

//http://maps.googleapis.com/maps/api/geocode/xml?address="+rue+de+péronnes+13+7640+Antoing+Belgique+ "&sensor=false
```

SITUATION

Code

Déclarations globales de SITUATION

```
gsNumTag  est un entier
gsNomTag  est une chaîne
gbFirstView  est un booléen = True
CentreLatitude est un réel
CentreLongitude est un réel
gsNomClientEnCours  est une chaîne
gnNumClientEnCours est un entier
gnNumClientEnCousProxi  est un entier
grDistance_proxi  est un réel
gsDateTournée  est une chaîne
grCentreLatitudeHome  est un réel //= val( _Praram.latitude)
grCentreLongitudeHome  est un réel

GR_Carto..Visible = False
```

SITUATION

Code des champs

Clic sur BTN_SansNom1

```
IF gbModeEPCIS THEN
    gbModeEPCIS = False
    MoiMême..Libellé = "WIthout EPCIS"
    GR_NO_EPCIS..Visible = False
ELSE
    gbModeEPCIS = True
    MoiMême..Libellé = "WIth EPCIS"
    GR_NO_EPCIS..Visible = True

END
Exécute("IMG_Carte..cli")
```

Clic sur BTN_SansNom2

```
IF gbModeAutorisation THEN
    gbModeAutorisation= False
    MoiMême..Libellé = "WIthout authorization"
    TABLE_EPC_NAMUR1..Visible = False
    TABLE_EPC_NAMUR2..Visible = False

ELSE
    gbModeAutorisation= True
    MoiMême..Libellé = "WIth authorization"
    TABLE_EPC_NAMUR1..Visible = True
    TABLE_EPC_NAMUR2..Visible = True

END
Exécute("IMG_Carte..cli")
```

Clic sur IMG_Carte

```
nNbhisto  est un entier = 0
Cnx est un gglConnexion
Cnx..Email = "muwac60@gmail.com"
```

```
Cnx..MotDePasse = "muwac6060"
Cnx..NomApplication = "MonAppli-03"
IF NOT gbCNxGGL THEN
    SI GglConnecte(Cnx) = Faux ALORS
        Erreur(ErreurInfo())
    ELSE
        gbCNxGGL=True
    FIN
END
sOld_Time  est une chaîne = "0000"
nValZoom   est une entier
//rDistance  est un reel
rLatmax  , rLatMin , rLonMax , rLonMin  sont des réels
rLatmax  = 0
rLatMin  = 90
rLonMax  = 0
rLonMin  = 90
rDistmax  est une réel = 100



Sablier(Vrai)
IF gbFirstView  THEN
    CentreLatitude=0
    CentreLongitude=0
END
sCléMap  est une chaîne = "votre clé google"
//TABLE_Table1..Libellé="Datas"
LocHome est un gglCoordonnée
sSnomrecherche est une chaîne
LocHome = GglAdresseVersCoordonnées(sCléMap ,sSnomrecherche) //(là on obtient les coordonnées de la rue)
MapsAPIKey est une chaîne = "votre clé google"

// Paramétrage d'une carte (ici vous rajoutez un marqueur sur la rue)
Params est un gglParamètreCarte
// Définition des marqueurs
tabTab_M est un tableau de 50 gglMarqueur
i est un entier = 0
j est un entier = 0
k  est un entier

FOR k = 1 TO 3
    i++;
    tabTab_M[i].Latitude = TABLE_Coord.COL_Latitude[k]
    tabTab_M[i].Longitude = TABLE_Coord.COL_Longitute[k]
    IF gbFirstView  THEN
        IF tabTab_M[i].Latitude > rLatmax  THEN rLatmax  = tabTab_M[i].Latitude
        IF tabTab_M[i].Latitude < rLatMin  THEN rLatMin  = tabTab_M[i].Latitude
        IF tabTab_M[i].Longitude > rLonMax  THEN rLonMax  = tabTab_M[i].Longitude
```

```
        IF tabTab_M[i].Longitude < rLonMin  THEN rLonMin  = tabTab_M[i].Longitude
    END
    //  if modulo(i,2) = 0 then
    //      tabTab_M[i].Couleur = irougeClair
    //  else
    //      tabTab_M[i].Couleur = ibleuclair
    //  end


    tabTab_M[i].Taille =gglGrand

    IF gbModeEPCIS THEN tabTab_M[1].Couleur = VertClair
    tabTab_M[2].Couleur = BleuClair
    IF gbModeEPCIS THEN tabTab_M[3].Couleur = RougeClair
    IF gbModeEPCIS THEN tabTab_M[1].Lettre = Caract(48+gnNBM)
    tabTab_M[2].Lettre = Caract(48+gnNBB)
    IF gbModeEPCIS AND gbModeAutorisation THEN tabTab_M[3].Lettre = Caract(48+gnNBN)

    Ajoute(Params.Marqueur, tabTab_M[i])

    //  TableAjoute(TABLE_Table1,i+TAB+ Requête.ident)
    //  TABLE_Table1[i]..Couleur=iCyanClair


END



Params.Format = gglJPG
Params.Cadre = Vrai


IF i > 0 THEN
    nValZoom  = 8//Abs(POT_Potentiomètre1-19)
    IF nValZoom  > 19 THEN
        nValZoom  = 19
    END
    //MoiMême=GglRécupèreCarte(CentreLatitude, CentreLongitude ,nValZoom  , 640,640,gglPlan,Params)
    IF gbFirstView  THEN
        gbFirstView  = False
        CentreLatitude= (rLatMin  + rLatmax  ) / 2
        CentreLongitude= (rLonMin  + rLonMax ) / 2

        rDistmax  = CalculDistDeuxPointsVincenty(Val(rLatMin ),Val(rLonMin ),Val(rLatmax ),Val(rLonMax ))
        IF rDistmax  > 100 THEN
```

```
            IF nValZoom >9 THEN
                nValZoom  =9
                POT_Potentiomètre1=9
            END
        END
        IF rDistmax  > 200 THEN
            IF nValZoom >8 THEN
                nValZoom  =8
                POT_Potentiomètre1=8
            END
        END
    END
ELSE
    gbFirstView  = True
END
SELON SEL_Plan
    CAS 1
        MoiMême=GglRécupèreCarte(CentreLatitude, CentreLongitude ,nValZoom  , 640,640,gglPlan,Params)
    CAS 2
        MoiMême=GglRécupèreCarte(CentreLatitude, CentreLongitude ,nValZoom  , 640,640,gglSatellite,Params)
    CAS 3
        MoiMême=GglRécupèreCarte(CentreLatitude, CentreLongitude ,nValZoom  , 640,640,gglTerrain,Params)
    CAS 4
        MoiMême=GglRécupèreCarte(CentreLatitude, CentreLongitude ,nValZoom  , 640,640,gglHybride,Params)

    AUTRE CAS
        //IMG_Image2 = GglRécupèreCarte(40.6423, -73.7959, 12, 600, 600, gglHybride, Point)
FIN

Sablier(Faux)

dSauveImageJPEG(IMG_Carte,"Ecologisticis.jpg")
```

Clic sur IMG_Image10

```
//gbFirstView =true
CentreLongitude=CentreLongitude-(POT_Potentiomètre1/500)
Exécute("Img_Carte..Cli")
```

Clic sur IMG_Image11

```
//gbFirstView =true
CentreLatitude=CentreLatitude+(POT_Potentiomètre1/500)

Exécute("IMG_Carte..Cli")
```

Clic sur IMG_Image12

```
//gbFirstView =true
CentreLongitude=CentreLongitude+(POT_Potentiomètre1/500)
Exécute("IMG_Carto..Cli")
```

Clic sur IMG_Image13

```
//gbFirstView =true
CentreLatitude  = 50.0945
CentreLongitude  = 4.5036
Exécute("img_Carte..Cli")
```

Clic sur IMG_Image9

```
//gbFirstView =true
CentreLatitude=CentreLatitude-(POT_Potentiomètre1/500)
Exécute("IMG_Carte..cli")
```

Clic sur IMG_SansNom2

```
IF OuiNon ("Delete database Bruxelles & Namur"+CR+"Are you sure ?") THEN
    Connexions_serveurs(3)
    IF gbConnected THEN
        HSupprimeTout (event_objectevent)
        HSupprimeTout(event_objectevent_epcs)
    END

    Connexions_serveurs(2)
    IF gbConnected THEN
        HSupprimeTout (event_objectevent)
        HSupprimeTout(event_objectevent_epcs)
    END
    Connexions_serveurs(1)
    IF gbConnected THEN
        HSupprimeTout (event_objectevent)
        HSupprimeTout(event_objectevent_epcs)
    END

    InitFenêtre(Vrai)
    gnCpt=0
END
```

Clic sur IMG_SansNom3

```
IF GR_Carto..Visible = False THEN
```

```
    GR_Carto..Visible = True
ELSE
    GR_Carto..Visible = False
END
```

Bouton gauche relâché (WM_LBUTTONUP) de POT_Potentiomètre1

```
gbFirstView=True
Exécute("IMG_Carte..cli")
```

A chaque modification de SEL_Plan

```
gbFirstView=True
Exécute("IMG_carte..cli")
```

Initialisation de TABLE_Coord

```
TableSupprimeTout (TABLE_Coord)

    TableAjoute (MoiMême,"M"+TAB+50.454301+TAB+3.957345)
    TableAjoute (MoiMême,"B"+TAB+50.845655+TAB+4.351014)
    TableAjoute (MoiMême,"N"+TAB+50.466108+TAB+4.868982)


    Exécute("IMG_carte..cli")
```

SITUATION

Procédures

Procédure locale Atan2

```
//
PROCEDURE Atan2(x,y)
xRes est un réel sur 8
SI x= 0 ALORS
    xRes = sgn(y) * ValPI/2.
SINON
    IF x > 0 THEN
        xRes = ArcTang(y/x)
    SINON
        xRes = ArcTang(y/x)+ValPI*sgn(y)
    END
FIN
RENVOYER xRes
```

Procédure locale CalculDistDeuxPointsVincenty

```
// Résumé : <indiquez ici ce que fait la procédure>
// Syntaxe :
//[ <Résultat> = ] CalculDistDeuxPointsVincenty (<xLat1>, <xLong1>, <xLat2>, <xLong2>)
//
// Paramètres :
//  xLat1 : <indiquez ici le rôle de xLat1>
//  xLong1 : <indiquez ici le rôle de xLong1>
//  xLat2 : <indiquez ici le rôle de xLat2>
//  xLong2 : <indiquez ici le rôle de xLong2>
// Valeur de retour :
//  Type indéterminé : //   Aucune
//
// Exemple :
// Indiquez ici un exemple d'utilisation.
//
PROCEDURE CalculDistDeuxPointsVincenty(xLat1,xLong1,xLat2,xLong2)

xa,xb,xf,xl sont des réels
```

```
// valeur pour WGS-84 ellipsoid
xa = 6378137
xb = 6356752.3142
xf = 1/298.257223563
xl = (xLong2-xLong1)*ValPI/180 // écart en longitude

// U est réduite de latitude
xU1, xU2 sont des réels
xU1 = ArcTang((1-xf)*Tang(xLat1))
xU2 = ArcTang((1-xf)*Tang(xLat2))
xLambda est un réels = xl //première approximation
xLambdap est un réels
nIterLimit est un entier = 100
xSinSigma,xCosSigma est un réels
xsigma,xsinAlpha,xcosSqAlpha,xcos2SigmaM est un réels
xc est un réels

//itération jusqu'au changement de ? est négligeable (par exemple 10 -12 ~ 0.06mm)
BOUCLE
    xSinSigma = Racine((Cos(xU2)*Sin(toDeg(xLambda)))^2+(Cos(xU1)*Sin(xU2)-Sin(xU1)*Cos(xU2)*Cos(toDeg(xLambda)))^2)
    SI xSinSigma = 0 ALORS RENVOYER 0 //co-incidences des points
    xCosSigma = Sin(xU1) * Sin(xU2) + Cos(xU1) * Cos(xU2) * Cos(toDeg(xLambda))
    xsigma = Atan2(xCosSigma,xSinSigma)*ValPI/180
    xsinAlpha = Cos(xU1) * Cos(xU2) * Sin(toDeg(xLambda)) / xSinSigma;
    xcosSqAlpha = 1 - xsinAlpha * xsinAlpha;
    xcos2SigmaM = xCosSigma - 2 * Sin(xU1) * Sin(xU2)/xcosSqAlpha;
    SI PAS xcos2SigmaM ALORS xcos2SigmaM = 0 // ligne équatoriale: cosSqAlpha = 0 (§ 6)
    xc = xf/16 * xcosSqAlpha * (4 + xf * (4-3 * xcosSqAlpha))
    xLambdap = xLambda
    xLambda = xl + (1-xc) * xf * xsinAlpha* (xsigma + xc * xSinSigma * (xcos2SigmaM + xc * xCosSigma * (-1 +2 * xcos2SigmaM^2)))
    nIterLimit--
A FAIRE TANTQUE (Abs(xLambda-xLambdap) > 1e-12 ET nIterLimit>0)

SI nIterLimit=0 ALORS RENVOYER 0
//si nIterLimit = 0 renvoyer 0
xUCarré est un réels
xUCarré = xcosSqAlpha*(xa^2-xb^2)/xb^2
xAA,xBB,xDeltasigma sont des réels
s est un réels
xAA = 1 + xUCarré/16384 * (4096 + xUCarré * (-768 + xUCarré * (320-175 *xUCarré)))
xBB = xUCarré/1024 * (256 + xUCarré * (-128 + xUCarré * (74-47* xUCarré)))
xDeltasigma = xBB * xSinSigma * (xcos2SigmaM + xBB / 4 * (xCosSigma * (-1 +2 * xcos2SigmaM * xcos2SigmaM) -xBB / 6 * xcos2SigmaM * (-3 +4
 * xSinSigma * xSinSigma) * (-3 +4 * xcos2SigmaM * xcos2SigmaM)))
s = xb*xAA*(xsigma-xDeltasigma)
s = Arrondi(s/1000,3)
RENVOYER s
```

Procédure locale Refresh_WIndows

```
// Résumé : <indiquez ici ce que fait la procédure>
// Syntaxe :
//Refresh_WIndows ()
//
// Paramètres :
//  Aucun
// Valeur de retour :
//  Aucune
//
// Exemple :
// Indiquez ici un exemple d'utilisation.
//// Procédure automatique :
// La procédure est exécutée automatiquement, après le code d'initialisation de la fenêtre, avec un différé de 1 seconde
// Elle s'exécutera dans un thread (sans avoir besoin d'appeler la fonction ThreadExecute), sans utilisation de HFSQL
// Elle sera répétée en boucle, en attendant 5 secondes entre chaque appel
//

PROCEDURE Refresh_WIndows()

i, indice est une entier
bCLiImage est un boolean = False
nBl est un entier

sLigne  est une chaîne


gnCpt ++


Connexions_serveurs(2)
IF gbConnected THEN
    HLitPremier(event_objectevent_epcs,event_id)
    TANTQUE PAS HEnDehors AND gbConnected
        HLitRecherche(event_objectevent,id,event_objectevent_epcs.event_id)
        IF HTrouve THEN
            HLitRecherche (voc_bizstep,id,event_objectevent.bizStep)
            IF HTrouve THEN
//                HLitRecherche(ean128,epc,event_objectevent_epcs.epc)
//                IF HTrouve THEN
                    sLigne = event_objectevent.recordTime+TAB+event_objectevent_epcs.epc+TAB+ExtraitChaîne(voc_bizstep.uri,1,":",DepuisFin)
//                END
                IF TableCherche(TABLE_EPC_BXL2.Col_Date_Heure,event_objectevent.recordTime) = -1 THEN
                    TableAjoute(TABLE_EPC_BXL2,sLigne)
                END
            END
        END
        HLitSuivant(event_objectevent_epcs,event_id)
    END
    TableTrie(TABLE_EPC_BXL2,"-Col_Date_Heure")
```

```
    FOR i = 1 TO TableOccurrence(TABLE_EPC_BXL2)
        IF TABLE_EPC_BXL2.COL_BizStep[i] = "receiving" THEN
            IF TableCherche(TABLE_EPC_BXL1.Col_Date_Heure,TABLE_EPC_BXL2.Col_Date_Heure[i]) = -1 THEN
                sligne=TABLE_EPC_BXL2.Col_Date_Heure[i]+TAB+TABLE_EPC_BXL2.COL_Epc[i]+TAB+TABLE_EPC_BXL2.COL_BizStep[i]
                TableAjoute(TABLE_EPC_BXL1,sligne)
            END
        END

    END
    FOR i = 1 TO TableOccurrence(TABLE_EPC_BXL2)

        IF TABLE_EPC_BXL2.COL_BizStep[i] = "shipping" THEN
            indice = TableCherche(TABLE_EPC_BXL1.COL_Epc,TABLE_EPC_BXL2.COL_Epc[i])
            IF indice > 0 THEN
                TableSupprime(TABLE_EPC_BXL1,indice)
            END
        END
    END

    nbl = TableOccurrence (TABLE_EPC_BXL1)
    IF nbl <> gnNBB THEN
        bCLiImage=True
        gnNBB = nbl
    END
END
Connexions_serveurs(3)
IF gbConnected THEN
    HLitPremier(event_objectevent_epcs,event_id)
    TANTQUE PAS HEnDehors AND gbConnected
        HLitRecherche(event_objectevent,id,event_objectevent_epcs.event_id)
        IF HTrouve THEN
            HLitRecherche (voc_bizstep,id,event_objectevent.bizStep)
            IF HTrouve THEN
//              HLitRecherche(ean128,epc,event_objectevent_epcs.epc)
//              IF HTrouve THEN
                    sLigne = event_objectevent.recordTime+TAB+event_objectevent_epcs.epc+TAB+ExtraitChaîne(voc_bizstep.uri,1,":",DepuisFin)
//              END
                IF TableCherche(TABLE_EPC_NAMUR2.Col_Date_Heure,event_objectevent.recordTime) = -1 THEN
                    TableAjoute(TABLE_EPC_NAMUR2,sLigne)
                END
            END
        END
        HLitSuivant(event_objectevent_epcs,event_id)
    END
    TableTrie(TABLE_EPC_NAMUR2,"-Col_Date_Heure")
    FOR i = 1 TO TableOccurrence(TABLE_EPC_NAMUR2)
        IF TABLE_EPC_NAMUR2.COL_BizStep[i] = "receiving" THEN
            IF TableCherche(TABLE_EPC_NAMUR1.Col_Date_Heure,TABLE_EPC_NAMUR2.Col_Date_Heure[i]) = -1 THEN
```

```
                sLigne=TABLE_EPC_NAMUR2.Col_Date_Heure[i]+TAB+TABLE_EPC_NAMUR2.COL_Epc[i]+TAB+TABLE_EPC_NAMUR2.COL_BizStep[i]
                TableAjoute(TABLE_EPC_NAMUR1,sLigne)
            END
        END

    END
    FOR i = 1 TO TableOccurrence(TABLE_EPC_NAMUR2)

        IF TABLE_EPC_NAMUR2.COL_BizStep[i] = "storing" THEN
            indice = TableCherche(TABLE_EPC_NAMUR1.COL_Epc,TABLE_EPC_NAMUR2.COL_Epc[i])
            IF indice > 0 THEN
                TableSupprime(TABLE_EPC_NAMUR1,indice)
            END
        END
    END
    nBl = TableOccurrence (TABLE_EPC_NAMUR1)
    IF nBl <> gnNBN THEN
        bCLiImage=True
        gnNBN = nBl
    END
END

    Connexions_serveurs(1)
    IF gbConnected THEN
        HLitPremier(event_objectevent_epcs,event_id)
        TANTQUE PAS HEnDehors AND gbConnected
            HLitRecherche(event_objectevent,id,event_objectevent_epcs.event_id)
            IF HTrouve THEN
                HLitRecherche (voc_bizstep,id,event_objectevent.bizStep)
                IF HTrouve THEN
//                  HLitRecherche(ean128,epc,event_objectevent_epcs.epc)
//                  IF HTrouve THEN
                        sLigne = event_objectevent.recordTime+TAB+event_objectevent_epcs.epc+TAB+ExtraitChaîne(voc_bizstep.uri,1,":",
                        DepuisFin)
//                  END
                    IF TableCherche(TABLE_EPC_Mons2.Col_Date_Heure,event_objectevent.recordTime) = -1 THEN
                        TableAjoute(TABLE_EPC_Mons2,sLigne)
                    END
                END
            END
            HLitSuivant(event_objectevent_epcs,event_id)
        END
        TableTrie(TABLE_EPC_Mons2,"-Col_Date_Heure")
        FOR i = 1 TO TableOccurrence(TABLE_EPC_Mons2)
            IF TABLE_EPC_Mons2.COL_BizStep[i] = "receiving" THEN
                IF TableCherche(TABLE_EPC_Mons1.Col_Date_Heure,TABLE_EPC_Mons2.Col_Date_Heure[i]) = -1 THEN
                    sLigne=TABLE_EPC_Mons2.Col_Date_Heure[i]+TAB+TABLE_EPC_Mons2.COL_Epc[i]+TAB+TABLE_EPC_Mons2.COL_BizStep[i]
                    TableAjoute(TABLE_EPC_Mons1,sLigne)
                END
```

```
            END

        END
        FOR i = 1 TO TableOccurrence(TABLE_EPC_Mons2)

            IF TABLE_EPC_Mons2.COL_BizStep[i] = "shipping" THEN
                indice = TableCherche(TABLE_EPC_Mons1.COL_Epc,TABLE_EPC_Mons2.COL_Epc[i])
                IF indice > 0 THEN
                    TableSupprime(TABLE_EPC_Mons1,indice)
                END
            END
        END
        nBl = TableOccurrence (TABLE_EPC_Mons1)
        IF nBl <> gnNBM THEN
            bCLiImage=True
            gnNBM = nBl
        END
    END

IF bCLiImage THEN
    Exécute("IMG_Carte..cli")
END
```

Procédure locale Refresh_WIndows1

```
// Résumé : <indiquez ici ce que fait la procédure>
// Syntaxe :
//Refresh_WIndows ()
//
// Paramètres :
//  Aucun
// Valeur de retour :
//  Aucune
//
// Exemple :
// Indiquez ici un exemple d'utilisation.
//
PROCEDURE Refresh_WIndows1()

i, indice est une entier
bCLiImage est un boolean = False
nBl est un entier

sLigne  est une chaîne


gnCpt ++

//Connexions_serveurs(3)
```

```
//HLitPremier(event_objectevent_epcs,event_id)
//TANTQUE PAS HEnDehors and gbConnected
//  HLitRecherche(event_objectevent,id,event_objectevent_epcs.event_id)
//  IF HTrouve THEN
//      HLitRecherche (voc_bizstep,id,event_objectevent.bizStep)
//      IF HTrouve THEN
//          HLitRecherche(ean128,epc,event_objectevent_epcs.epc)
//          IF HTrouve THEN
//              sLigne = event_objectevent.recordTime+TAB+ean128.compl+TAB+ExtraitChaîne(voc_bizstep.uri,1,":",DepuisFin)
//          END
//          if TableCherche(TABLE_EPC_BXL2.Col_Date_Heure,event_objectevent.recordTime) = -1 then
//              TableAjoute(TABLE_EPC_BXL2,sLigne)
//          end
//      END
//  END
//  HLitSuivant(event_objectevent_epcs,event_id)
//END
//TableTrie(TABLE_EPC_BXL2,"-Col_Date_Heure")
//
//for i = 1 to tableoccurrence(TABLE_EPC_BXL2)
//  if TABLE_EPC_BXL2.COL_BizStep[i] = "receiving" then
//      IF TableCherche(TABLE_EPC_BXL1.Col_Date_Heure,TABLE_EPC_BXL2.Col_Date_Heure[i]) = -1 THEN
//          sligne=TABLE_EPC_BXL2.Col_Date_Heure[i]+TAB+TABLE_EPC_BXL2.COL_Epc[i]+TAB+TABLE_EPC_BXL2.COL_BizStep[i]
//          tableajoute(TABLE_EPC_BXL1,sligne)
//      end
//  END
//
//END
//FOR i = 1 TO TableOccurrence(TABLE_EPC_BXL2)
//
//  IF TABLE_EPC_BXL2.COL_BizStep[i] = "shipping" THEN
//      indice = TableCherche(TABLE_EPC_BXL1.COL_Epc,TABLE_EPC_BXL2.COL_Epc[i])
//      IF indice > 0 THEN
//          TableSupprime(TABLE_EPC_BXL1,indice)
//      END
//  END
//END
//nbl = tableoccurrence (TABLE_EPC_BXL1)
//if nbl <> gnnbb then
//  bCLiImage=true
//  gnNBB = nbl
//END

Connexions_serveurs(2)
HLitPremier(event_objectevent_epcs,event_id)
TANTQUE PAS HEnDehors AND gbConnected
    HLitRecherche(event_objectevent,id,event_objectevent_epcs.event_id)
    IF HTrouve THEN
        HLitRecherche (voc_bizstep,id,event_objectevent.bizStep)
```

```
        IF HTrouve THEN
            HLitRecherche(ean128,epc,event_objectevent_epcs.epc)
            IF HTrouve THEN
                sLigne = event_objectevent.recordTime+TAB+ean128.compl+TAB+ExtraitChaîne(voc_bizstep.uri,1,":",DepuisFin)
            END
            IF TableCherche(TABLE_EPC_NAMUR2.Col_Date_Heure,event_objectevent.recordTime) = -1 THEN
                TableAjoute(TABLE_EPC_NAMUR2,sLigne)
            END
        END
    END
    HLitSuivant(event_objectevent_epcs,event_id)
END
TableTrie(TABLE_EPC_NAMUR2,"-Col_Date_Heure")
FOR i = 1 TO TableOccurrence(TABLE_EPC_NAMUR2)
    IF TABLE_EPC_NAMUR2.COL_BizStep[i] = "receiving" THEN
        IF TableCherche(TABLE_EPC_NAMUR1.Col_Date_Heure,TABLE_EPC_NAMUR2.Col_Date_Heure[i]) = -1 THEN
            sLigne=TABLE_EPC_NAMUR2.Col_Date_Heure[i]+TAB+TABLE_EPC_NAMUR2.COL_Epc[i]+TAB+TABLE_EPC_NAMUR2.COL_BizStep[i]
            TableAjoute(TABLE_EPC_NAMUR1,sLigne)
        END
    END

END
FOR i = 1 TO TableOccurrence(TABLE_EPC_NAMUR2)

    IF TABLE_EPC_NAMUR2.COL_BizStep[i] = "shipping" THEN
        indice = TableCherche(TABLE_EPC_NAMUR1.COL_Epc,TABLE_EPC_NAMUR2.COL_Epc[i])
        IF indice > 0 THEN
            TableSupprime(TABLE_EPC_NAMUR1,indice)
        END
    END
END
nBl = TableOccurrence (TABLE_EPC_NAMUR1)
IF nBl <> gnNBN THEN
    bCLiImage=True
    gnNBN = nBl
END

//if gnCpt <2 then
//  Connexions_serveurs(1)
//  HLitPremier(event_objectevent_epcs,event_id)
//  TANTQUE PAS HEnDehors AND gbConnected
//      HLitRecherche(event_objectevent,id,event_objectevent_epcs.event_id)
//      IF HTrouve THEN
//          HLitRecherche (voc_bizstep,id,event_objectevent.bizStep)
//          IF HTrouve THEN
//              HLitRecherche(ean128,epc,event_objectevent_epcs.epc)
//              IF HTrouve THEN
//                  sLigne = event_objectevent.recordTime+TAB+ean128.compl+TAB+ExtraitChaîne(voc_bizstep.uri,1,":",DepuisFin)
//              END
```

```
//              IF TableCherche(TABLE_EPC_Mons2.Col_Date_Heure,event_objectevent.recordTime) = -1 THEN
//                  TableAjoute(TABLE_EPC_Mons2,sLigne)
//              END
//          END
//      END
//      HLitSuivant(event_objectevent_epcs,event_id)
//  END
//  TableTrie(TABLE_EPC_Mons2,"-Col_Date_Heure")
//  FOR i = 1 TO TableOccurrence(TABLE_EPC_Mons2)
//      IF TABLE_EPC_Mons2.COL_BizStep[i] = "receiving" THEN
//          IF TableCherche(TABLE_EPC_Mons1.Col_Date_Heure,TABLE_EPC_Mons2.Col_Date_Heure[i]) = -1 THEN
//              sLigne=TABLE_EPC_Mons2.Col_Date_Heure[i]+TAB+TABLE_EPC_Mons2.COL_Epc[i]+TAB+TABLE_EPC_Mons2.COL_BizStep[i]
//              TableAjoute(TABLE_EPC_Mons1,sLigne)
//          END
//      END
//
//  END
//  FOR i = 1 TO TableOccurrence(TABLE_EPC_Mons2)
//
//      IF TABLE_EPC_Mons2.COL_BizStep[i] = "shipping" THEN
//          indice = TableCherche(TABLE_EPC_Mons1.COL_Epc,TABLE_EPC_Mons2.COL_Epc[i])
//          IF indice > 0 THEN
//              TableSupprime(TABLE_EPC_Mons1,indice)
//          END
//      END
//  END
//  nBl = TableOccurrence (TABLE_EPC_Mons1)
//  IF nBl <> gnNBM THEN
//      bCLiImage=True
//      gnNBM = nBl
//  END
//end
IF bCLiImage THEN
    Exécute("IMG_Carte..cli")
END
```

Procédure locale sgn

```
// Résumé : <indiquez ici ce que fait la procédure>
// Syntaxe :
//[ <Résultat> = ] sgn (<nval>)
//
// Paramètres :
//  nval : <indiquez ici le rôle de nval>
// Valeur de retour :
//  réel : //   Aucune
//
// Exemple :
// Indiquez ici un exemple d'utilisation.
```

```
//

PROCEDURE sgn(nval)

xRes est un réel sur 8
SI nval = 0 ALORS
    xRes = 0
SINON
    xRes = nval/Abs(nval)
FIN
RENVOYER xRes
```

Procédure locale ToDeg

```
// Résumé : <indiquez ici ce que fait la procédure>
// Syntaxe :
//[ <Résultat> = ] ToDeg (<xRad>)
//
// Paramètres :
//  xRad : <indiquez ici le rôle de xRad>
// Valeur de retour :
//  variant : //    Aucune
//
// Exemple :
// Indiquez ici un exemple d'utilisation.
//

// Conversion radian -> degrés
PROCEDURE ToDeg(xRad)
RENVOYER Conversion(xRad,"rad","degré")
```

FEN_TDT1

Code des champs

Clic sur BTN_SansNom1

```
HCréationSiInexistant(prefix)
HSupprimeTout(prefix)

Ligne est une chaîne
nFic1 est un entier = fOuvre("C:\Aspire_Fosstrak\Documentation\GCPPrefixFormatList.xml",foLectureEcriture)
IF nfic1 > 0 THEN
    ligne = fLitLigne(nfic1)
    TANTQUE ligne <> EOT
        HRAZ (prefix)
        prefix.Prefix_Code = ExtraitChaîne(Ligne,2,Caract(34))
        prefix.Prefix_Length = Val (ExtraitChaîne(Ligne,4,Caract(34)))
        IF prefix.Prefix_Length > 0 THEN
            HAjoute(prefix)
        END
        ligne  = fLitLigne(nfic1)


    FIN

    fFerme(nfic1)
END
```

Initialisation de HTM_SansNom1

```
MoiMême = "http://193.190.194.228:8080/epcis-repository-0.5.0/capture?showCaptureForm=true"
```

Clic sur IMG_SansNom1

```
sResultat est une chaîne = Main_Code128_TO_EPCGen2__Convert(SAI_SGTIN)
SAI_EPC_URI = ExtraitChaîne(sresultat,1,";")
SAI_EPC_Tag_URI = ExtraitChaîne(sResultat,2,";")
SAI_EPC_Binary = ExtraitChaîne(sResultat,3,";")
```

```
SAI_EPC_HEXA = Binaire_to_Hexa_Convert(ExtraitChaîne(sResultat,3,";"))
```

Partie 4

Etat

ETAT_cb

Code des champs

Avant impression de CBA_1

```
MoiMême = "0110614141000415"
```

Partie 5

Collection de procédures

COL_ProcéduresGlobales

Code

Procédure globale AI_Value__Extract

```
// Résumé : <indiquez ici ce que fait la procédure>
// Syntaxe :
//[ <Résultat> = ] AI_Value__Extract (<AI>, <Code128>)
//
// Paramètres :
//  AI : <indiquez ici le rôle de AI>
//  Code128 : <indiquez ici le rôle de Code128>
// Valeur de retour :
//  chaîne : //     Aucune
//
// Exemple :
// Indiquez ici un exemple d'utilisation.
//
PROCEDURE AI_Value__Extract(ai,Code128)

sValRet est une chaîne = ""
sAi est une chaîne = ai
TANTQUE Taille (sai) < 2
    sai ="0"+sai
FIN

sAi  = "("+sai+")"


c est un caractère
P est un entier
bSortie est un boolean = False

P = Position(Code128,sai)
IF P > 0 THEN
    P+=4
    TANTQUE PAS bSortie
        c=Milieu(Code128,P,1)
        IF Asc(c) >47 AND Asc(c) <(58) THEN
            svalret += c
            P++
        ELSE
            bSortie =
```

```
True
        END
    END
END

RENVOYER(svalret)
```

Procédure globale Code128_To_SGTIN96__Convert

```
// Résumé : <indiquez ici ce que fait la procédure>
// Syntaxe :
//[ <Résultat> = ] Code128_To_SGTIN96__Convert (<cb>)
//
// Paramètres :
//  cb : <indiquez ici le rôle de cb>
// Valeur de retour :
//  chaîne : //     Aucune
//
// Exemple :
// Indiquez ici un exemple d'utilisation.
//
PROCEDURE Code128_To_SGTIN96__Convert(cb)

sValRet est une chaîne = ""
urn1 est une chaîne = "urn:epc:id:sgtin:"
urn2 est une chaîne = "urn:epc:tag:sgtin-96:3."
nHeader est un entier = 48
nFilter est un entier = 3
nPartition est un entier = 5
sCompany est une chaîne
sIndicator est une chaîne
sSerial est une chaîne


sVal1,sVal2,sVal3,sVal4,sVal5 sont des chaînes


sGtin est une chaîne =AI_Value__Extract(01,cb)
Tsgtin est un entier = Taille(sgtin)
sSerialNumber est une chaîne = AI_Value__Extract(21,cb)
sVal1 = urn1 + Milieu(sGtin,2,7)+"."+Gauche(sGtin,1)+Milieu(sGtin,9,Tsgtin-9)+"."+sSerialNumber
sVal2 = urn2 + Milieu(sGtin,2,7)+"."+Gauche(sGtin,1)+Milieu(sGtin,9,Tsgtin-9)+"."+sSerialNumber
scompany = Decimal_to_Base__Convert(2,Val(ExtraitChaîne(ExtraitChaîne(sval1,5,":"),1,".")),24)
sIndicator = Decimal_to_Base__Convert(2,Val(ExtraitChaîne(sVal1,2,".")),20)
sSerial = Decimal_to_Base__Convert(2,Val(ExtraitChaîne(sVal1,3,".")),38)
sval3 = Decimal_to_Base__Convert(2,nheader,8)+ Decimal_to_Base__Convert(2,nfilter,3)+ Decimal_to_Base__Convert(2,npartition,3)+ scompany
+ sindicator+sserial
```

```
svalret = sval1+";"+sval2+";"+sval3+";"+sval4+";"+sval5

RENVOYER (svalret)
```

Procédure globale Code128_To_SSCC96__Convert

```
// Résumé : <indiquez ici ce que fait la procédure>
// Syntaxe :
//[ <Résultat> = ] Code128_To_SSCC96__Convert (<cb>)
//
// Paramètres :
//  cb : <indiquez ici le rôle de cb>
// Valeur de retour :
//  chaîne ANSI : //    Aucune
//
// Exemple :
// Indiquez ici un exemple d'utilisation.
//
PROCEDURE Code128_To_SSCC96__Convert(cb)

sValRet est une chaîne = ""
cb=""
//urn1 est une chaine = "urn:epc:id:sscc:"
//urn2 est une chaine = "urn:epc:tag:sscc-96:3."
//
//sVal1,sVal2,sVal3,sVal4,sVal5 sont des chaines
//
//
RENVOYER(svalret)
```

Procédure globale Connexions_serveurs

```
PROCEDURE Connexions_serveurs(NumSrv)


sServeurAdresse1 est une chaîne
sServeurAdresse2 est une chaîne
sServeurAdresse3 est une chaîne

sAdmin1, sAdmin2, sAdmin3 est une chaîne
sPassword1, sPassword2, sPassword3 est une chaîne
nFic2 sont des entiers

//
nFic2 = fOuvre("SW_TRACK_MYSQL_CNX.txt",foLecture)
IF nFic2 > 0 THEN
```

```
    sServeurAdresse1 = fLitLigne(nFic2)
    sAdmin1 = fLitLigne(nFic2)
    sPassword1=fLitLigne(nFic2)
    sServeurAdresse2 = fLitLigne(nFic2)
    sAdmin2 = fLitLigne(nFic2)
    sPassword2=fLitLigne(nFic2)
    sServeurAdresse3 = fLitLigne(nFic2)
    sAdmin3 = fLitLigne(nFic2)
    sPassword3=fLitLigne(nFic2)
    fFerme(nFic2)
    gbConnected = True
    HFermeConnexion(cnxMySQL)
    SELON numsrv
        CAS 1
            HDécritConnexion(cnxMySQL,sAdmin1,sPassword1,sServeurAdresse1,"epcis",hAccèsNatifMySQL)
        CAS 2
            HDécritConnexion(cnxMySQL,sAdmin2,sPassword2,sServeurAdresse2,"epcis",hAccèsNatifMySQL)
        CAS 3
            HDécritConnexion(cnxMySQL,sAdmin3,sPassword3,sServeurAdresse3,"epcis",hAccèsNatifMySQL)


        AUTRE CAS

    FIN

    IF HOuvreConnexion(cnxMySQL) THEN
        HChangeConnexion(event_objectevent,cnxMySQL)
        HChangeConnexion(event_objectevent_epcs,cnxMySQL)
        HChangeConnexion(voc_bizloc,cnxMySQL)
        HChangeConnexion(voc_bizstep,cnxMySQL)
        HChangeConnexion(voc_disposition,cnxMySQL)
        HChangeConnexion(voc_readpoint,cnxMySQL)
    ELSE
        gbConnected = False
        DélaiAvantFermeture(150)
        SELON NumSrv


            CAS 1
                Info ("Pas de connexion avec le serveur  EPCIS MONS")
            CAS 2
                Info ("Pas de connexion avec le serveur  EPCIS BRUXELLES")
            CAS 3
                Info ("Pas de connexion avec le serveur  EPCIS NAMUR")

            AUTRE CAS

        FIN
```

```
        SI ErreurDétectée ALORS
            Erreur()
        FIN
    END
END
```

Procédure globale Create_ZPLFile

```
// Résumé : <indiquez ici ce que fait la procédure>
// Syntaxe :
//Create_ZPLFile (<Filename>, <CB>)
//
// Paramètres :
//  Filename : <indiquez ici le rôle de Filebname>
//  CB : <indiquez ici le rôle de CB>
// Valeur de retour :
//  Aucune
//
// Exemple :
// Indiquez ici un exemple d'utilisation.
//
PROCEDURE Create_ZPLFile(Filename,CB)

sLigne est une chaîne

nFic est un entier = fOuvre(filename,foCréation+foLectureEcriture+foAjout)
IF nfic > 1 THEN
    sligne = "^XA"; fEcritLigne(nFic,sLigne)
    sLigne = "^FO40,40^A0,45,55^FD" + cb +"^FS"; fEcritLigne(nFic,sLigne)
    sligne = "^XZ"; fEcritLigne(nFic,sLigne)
    fFerme(nfic)
END
```

Procédure globale GestionConnexion_TCP

```
// Traitement automatique des erreurs :   exécuter le bloc de code CAS ERREUR
//
// Traitement automatique des exceptions :   exécuter le bloc de code CAS EXCEPTION:
//

PROCEDURE GestionConnexion_TCP(sNomConnexion)
```

```
//sMessage est une chaine
sIp est une chaîne = SocketClientInfo(sNomConnexion,SocketAdresse)
sMsg est une chaîne  = ""
bTimeOut est un boolean = False
bSortie est boolean = True
//nFinAtt  est un entier = HeureVersEntier(now) + gnTimeOutSocket
//bModeDHCP est un boolean = false
// Lecture des messages
//NumBP, nEqt est un entier
//sNumPièce est une chaine
pos est un entier


BOUCLE
    SocketChangeModeTransmission(sNomConnexion,SocketSansMarqueurFin)
    gsMsgTCP = SocketLit(sNomConnexion,Faux,10)
    Pos = Position(gsMsgTCP,CR)
    IF ((Taille(gsMsgTCP)> 1) AND (Pos > 0)) THEN
        Trace(gsMsgTCP)
        SORTIR
    END
END

//FIN

// Fermeture de la socket
//

CAS ERREUR:
CAS EXCEPTION:
```

Procédure globale Lecture_TCP

```
// Procédure automatique :
// La procédure est exécutée automatiquement, après le code d'initialisation du projet
// Elle s'exécutera dans un thread (sans avoir besoin d'appeler la fonction ThreadExecute), avec utilisation de HFSQL
// Elle sera répétée en boucle, en attendant 1 seconde entre chaque appel
//
// Traitement automatique des erreurs :   exécuter le bloc de code CAS ERREUR
//
// Traitement automatique des exceptions :   exécuter le bloc de code CAS EXCEPTION:
//

PROCEDURE Lecture_TCP()


//SectionCritiqueDébut("Lecture_TCP")
//
```

```
//sNomCanal est une chaine
//if not gbSocketTCPBusy THEN
//  gbSocketTCPBusy = true
//
//  SI SocketAttendConnexion(gsNomSocketTCP) ALORS
//      sNomCanal = SocketAccepte(gsNomSocketTCP)
//
//      SI sNomCanal~="" ALORS
//          Erreur("Impossible de créer la socket nécessaire à la nouvelle connexion",ErreurInfo())
//      SINON
//          GestionConnexion_Tcp(sNomCanal)
//      FIN
//  FIN
//  gbSocketTCPBusy = false
//end
//SectionCritiqueFin ("Lecture_TCP")
//CAS ERREUR:
//CAS EXCEPTION:
```

Procédure globale Main_Code128_TO_EPCGen2__Convert

```
PROCEDURE Main_Code128_TO_EPCGen2__Convert(Code128)

sValRet est une chaîne = ""
urn1 est une chaîne = "urn:epc:id:sgtin:"


IF Position(Code128,"(00)") > 0 THEN sValRet = Code128_To_SSCC96__Convert(code128)
IF Position(Code128,"(01)") > 0 THEN sValRet = Code128_To_SGTIN96__Convert(code128)









RENVOYER(svalret)
```

Procédure globale Print_ZPLFile

```
// Résumé : <indiquez ici ce que fait la procédure>

PROCEDURE Print_ZPLFile(IP,Filename)

//SocketConnecte(csNomSocketTSC2410,cnPorTSC2410,ip)
//SI ErreurDétectée ALORS
//
//  Erreur("erreur de connexion " + ErreurInfo(errMessage))
//
//ELSE
//  Zone, ligne sont des chaines
//  nfic est un entier = fouvre(filename)
//  IF nfic > 1 THEN
//      ligne = fLitLigne(nfic)
//      TANTQUE ligne <> EOT
//          Zone += ligne
//          ligne = fLitLigne(nfic)
//      FIN
//  END
//  SocketEcrit(csNomSocketTSC2410,Zone)
//  SocketFerme(csNomSocketTSC2410)
//FIN
```

Partie 6

Table des matières

Table des matières

Projet TDT