# Inicia

!! Llena los pedidos que se pueden surtir

## DO Sr\_Llena\_Queue

!! Valida que haya registros a revisar

IF RECORDS(GloqPed) = 0 THEN

MESSAGE('No Se Localizó Ningún Pedido Para Poder Surtir En Este Momento.||Se Recomienta Intentar Nuevamente Más Tarde Por Si Hay Más Disponible','Imposible Continuar',ICON:HAND)

DO ProcedureReturn

END

!! Genera el botón automáticamente

POST(EVENT:ACCEPTED, ?B\_Proceso)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!! 1 - Rutina de Carga de Queue con Apartados

# Sr\_Llena\_Queue ROUTINE

DATA

LocF:AlmCEDIS BYTE !! Indica el Almacén CEDIS

LocF:YaPedidos DECIMAL(18,2) !! YaPedidos quitando el cero

LocF:Cant\_Faltante DECIMAL(18,2) !! Cantidad Faltante en Almacén Destino del Traspaso

LocF:Cant\_Disp DECIMAL(18,2) !! Cantidad Disponible en Almacén Origen del Traspaso

LocF:Cant\_Trasp DECIMAL(18,2) !! Cantidad a Traspasar

CODE

!! Limpiar Global

FREE(GloqPed)

CLEAR(GloqPed)

!! Inicializar el Valor del Almacén CEDIS

LocF:AlmCEDIS = 91

!! Indica que se está procesando

SETCURSOR(CURSOR:Wait)

!! Recorrer los Detalles del Pedido

CLEAR(DPDC:Record)

DPDC:Backorder = 0.01

SET(DPDC:LLS\_BackOrder,DPDC:LLS\_BackOrder)

Loop

Next(ELDPDC)

If ErrorCode() Then Break.

!! Revisar sólo las que tengan BackOrder

If ~DPDC:Backorder Then Break.

!! Si la partida está cancelada, no tomar en cuenta

IF DPDC:EdoDetalle = 3 then cycle .

!! Ajusta ya pedidos para evitar negativos

IF DPDC:YaPedidos < 0 THEN

LocF:YaPedidos = 0

ELSE

LocF:YaPedidos = DPDC:YaPedidos

END

!! Calcular cantidad Faltante

LocF:Cant\_Faltante = DPDC:Backorder - LocF:YaPedidos

If LocF:Cant\_Faltante <= 0 Then Cycle .

!! Buscar Cabecera de Pedidos

CLEAR(PDC:Record)

PDC:LFolio= DPDC:LFolio

PDC:NFolio= DPDC:NFolio

GET(ELPDC,PDC:LLP\_Folio)

If ErrorCode() Then Cycle.

!! Sólo verificar pedidos "Asigna, Traspasa y Compra (1) y No-Stock (2)"

If PDC:TPedido > 2 Then Cycle.

!! Si ya está facturado total, no analizar

If PDC:Estado = 2 Then Cycle.

!! Si está cancelado el pedido, no analizar

if ~PDC:Vigencia Then Cycle.

!! Si no está autorizado, no tomar en cuenta

If ~PDC:BndAutorizacion Then Cycle.

!! Verifica si tiene disponible en el almacén origen del traspaso

CLEAR(EXA:Record)

EXA:CArt = DPDC:CArt

EXA:NAlm = LocF:AlmCEDIS

GET(ELEXA,EXA:LLP\_CArt\_NAlm)

If ErrorCode() Then Cycle.

!! Calcula el disponible

LocF:Cant\_Disp = EXA:Existencia-EXA:Apartados

IF LocF:Cant\_Disp <= 0 Then Cycle.

! Buscar Artículo

CLEAR(ART:Record)

ART:CArt = DPDC:CArt

GET(ElArt,ART:LLP\_CArticulo)

If ErrorCode() Then Clear(Art:Record).

!! Valida que la cantidad no sea menos de la unidad de traspaso

IF LocF:Cant\_Disp < ART:FacTraspasos THEN CYCLE .

!! Ajusta el disponible a traspasos hacia abajo

LocF:Cant\_Disp = INT(LocF:Cant\_Disp / ART:FacTraspasos) \* ART:FacTraspasos

!! No realizar traspaso de Backorder de artículo de tramos

IF ART:Bnd\_Tramos THEN CYCLE .

! !! Ajustar a Empaques la cantidad Faltante

! IF LocF:Cant\_Faltante < ART:FacTraspasos THEN

! LocF:Cant\_Faltante = ART:FacTraspasos

! ELSE

! IF (LocF:Cant\_Faltante % ART:FacTraspasos) = 0 THEN

! !! Ya está ajustado a empaques

! ELSE

! !! Ajusta el faltante al factor de traspasos hacia arriba

! LocF:Cant\_Faltante = (INT(LocF:Cant\_Faltante / ART:FacTraspasos) + 1) \* ART:FacTraspasos

! END

! END

!! analiza después de pasar a factor de traspasos

If LocF:Cant\_Faltante <= 0 Then Cycle .

!! Añade al Queue

CLEAR(GloqPed)

GloqPed.GqLFolio = PDC:LFolio

GloqPed.GqNFolio = PDC:NFolio

GloqPed.GqNPartida = DPDC:NPartida

GloqPed.GqCArt = DPDC:CArt

GloqPed.GqNAlm = PDC:NAlm\_Sal

GloqPed.GqLFolioT = '' !! Aún no se sabe en qué traspaso quedará

GloqPed.GqNFolioT = 0 !! Aún no se sabe en qué traspaso quedará

GloqPed.GqBackOrder = LocF:Cant\_Faltante

GloqPed.GqDisponible = LocF:Cant\_Disp

GloqPed.GqFcAut = PDC:FcAutorizacion

GloqPed.GqHrAut = PDC:HrAutorizacion

GloqPed.GQBOFinal = LocF:Cant\_Faltante

GloqPed.GqResult = 'Pendiente'

ADD(GloqPed)

SORT(GloqPed, +GloqPed.GqLFolio, +GloqPed.GqNFolio, +GloqPed.GqNPartida)

END

!! Regresa el cursor normal

SETCURSOR(CURSOR:Arrow)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

OF ?B\_Proceso

! Inicio de "Control Handling, before event handling"

! [Priority 5000]

! Fin de "Control Handling, before event handling"

CASE EVENT()

OF EVENT:Accepted

! Inicio de "Control Event Handling, before generated code"

! [Priority 5000]

! Fin de "Control Event Handling, before generated code"

DO SyncWindow

! Inicio de "Control Event Handling, after generated code"

! [Priority 4000]

!! Confirmación e inicio del procedimiento

!IF MESSAGE('¿Está Seguro Que Desea Iniciar El Procedimiento?','Confirmación',ICON:QUESTION,'Sí|No',2) = 1 THEN

!! Deshabilitar Botones

DISABLE(?B\_Proceso)

DISABLE(?B\_Cancelar)

!! Indica que se conserve el queue global

Loc:DejarQueueGlobal = TRUE

!! Llamar el procedimiento

DO Sr\_Proceso

!! Llama la ventana de resultados

START(ElRstTpsBOCds, 25000, Loc:MenError)

!! Cierra esta ventana

DO ProcedureReturn

!END

! Fin de "Control Event Handling, after generated code"

OF EVENT:Selected

! Inicio de "Control Event Handling, before generated code"

! [Priority 5000]

! Fin de "Control Event Handling, before generated code"

! Inicio de "Control Event Handling, after generated code"

! [Priority 5000]

! Fin de "Control Event Handling, after generated code"

ELSE

! Inicio de "Other Control Event Handling"

! [Priority 5000]

! Fin de "Other Control Event Handling"

END

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!! 2 - Revisa el Queue e Intenta Generar el Traspaso

# Sr\_Proceso ROUTINE

DATA

LocF:Cx BYTE !! Contador

LocF:AlmCEDIS BYTE !! Indica el Almacén CEDIS

LocF:YaPedidos DECIMAL(18,2) !! YaPedidos quitando el cero

LocF:Cant\_Faltante DECIMAL(18,2) !! Cantidad Faltante en Almacén Destino del Traspaso

LocF:Cant\_Disp DECIMAL(18,2) !! Cantidad Disponible en Almacén Origen del Traspaso

LocF:Cant\_Trasp DECIMAL(18,2) !! Cantidad a Traspasar

LocF:Cant\_BOOriginal DECIMAL(18,2) !! Cantidad de BackOrder Original

LocF:Cant\_BOFinal DECIMAL(18,2) !! Cantidad Final para evitar negativos

LocF:Disp\_AlmOrigen DECIMAL(18,2) !! Disponible (sin ajustar a empaques) del almacén origen

LocF:Exceso DECIMAL(18,2) !! Cantidad de Exceso en el Almacén

LocF:DifExceso DECIMAL(18,2) !! Diferencia en exceso, variable auxiliar para obtener el exceso

LocF:LFolioT LIKE(TPS:LFolio)

LocF:NFolioT LIKE(TPS:NFolio)

CODE

!! Reorganizar el queue para asignar por fecha y hora de autorización

CLEAR(GloqPed)

SORT(GloqPed, +GloqPed.GqFcAut , +GloqPed.GqHrAut)

!! Indica el almacén considerado CEDIS

LocF:AlmCEDIS = 91

!! Inicia la transacción

LOGOUT(1, ELTCMI, ELTPS, ELDPDC, ELEXA, ELDTPS, ELPESA)

IF ERRORCODE() THEN

!! Guarda el Mensaje de error

Loc:MenError = 'Ocurrió el error (' & ErrorCode() & '): ' & clip(Error()) & ' al momento de iniciar la transacción.'

!! Termina la Transacción

ROLLBACK

!! Almacena la información en el Queue

GloqPed.GqResult = 'Error En Existencia'

PUT(GloqPed)

!! Muestra mensaje de error original (para evitar el error del rollback)

MESSAGE(Loc:MenError,'Imposible Continuar',ICON:HAND)

!! Termina el proceso

EXIT

END

!! Recorrer

LOOP LocF:Cx = 1 TO RECORDS(GloqPed)

CLEAR(GloqPed)

GET(GloqPed, LocF:Cx)

!! Busca el Detalle del Pedido

CLEAR(DPDC:Record)

DPDC:LFolio = GloqPed.GqLFolio

DPDC:NFolio = GloqPed.GqNFolio

DPDC:NPartida = GloqPed.GqNPartida

GET(ELDPDC, DPDC:LLP\_Folio\_NPartida)

IF ERRORCODE() THEN CYCLE .

!! \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

!! Revisar sólo las que tengan BackOrder

If ~DPDC:Backorder Then Break.

!! Si la partida está cancelada, no tomar en cuenta

IF DPDC:EdoDetalle = 3 then cycle .

!! Ajusta ya pedidos para evitar negativos

IF DPDC:YaPedidos < 0 THEN

LocF:YaPedidos = 0

ELSE

LocF:YaPedidos = DPDC:YaPedidos

END

!! Calcular cantidad Faltante

LocF:Cant\_Faltante = DPDC:Backorder - LocF:YaPedidos

LocF:Cant\_BOOriginal = DPDC:Backorder - LocF:YaPedidos

If LocF:Cant\_Faltante <= 0 Then Cycle .

!! Buscar Cabecera de Pedidos

CLEAR(PDC:Record)

PDC:LFolio= DPDC:LFolio

PDC:NFolio= DPDC:NFolio

GET(ELPDC,PDC:LLP\_Folio)

If ErrorCode() Then Cycle.

!! Sólo verificar pedidos "Asigna, Traspasa y Compra (1) y No-Stock (2)"

If PDC:TPedido > 2 Then Cycle.

!! Si ya está facturado total, no analizar

If PDC:Estado = 2 Then Cycle.

!! Si está cancelado el pedido, no analizar

if ~PDC:Vigencia Then Cycle.

!! Si no está autorizado, no tomar en cuenta

If ~PDC:BndAutorizacion Then Cycle.

!! Verifica si tiene disponible en el almacén origen del traspaso

CLEAR(EXA:Record)

EXA:CArt = DPDC:CArt

EXA:NAlm = LocF:AlmCEDIS

GET(ELEXA,EXA:LLP\_CArt\_NAlm)

If ErrorCode() Then Cycle.

!! Calcula el disponible (LocF:Cant\_Disp ajustado a empaques y LocF:Disp\_AlmOrigen sin ajustar)

LocF:Disp\_AlmOrigen = EXA:Existencia-EXA:Apartados

LocF:Cant\_Disp = EXA:Existencia-EXA:Apartados

IF LocF:Cant\_Disp <= 0 Then Cycle.

! Buscar Artículo

CLEAR(ART:Record)

ART:CArt = DPDC:CArt

GET(ElArt,ART:LLP\_CArticulo)

If ErrorCode() Then Clear(Art:Record).

!! Valida que la cantidad no sea menos de la unidad de traspaso

IF LocF:Cant\_Disp < ART:FacTraspasos THEN CYCLE .

!! Ajusta el disponible a traspasos hacia abajo

LocF:Cant\_Disp = INT(LocF:Cant\_Disp / ART:FacTraspasos) \* ART:FacTraspasos

!! No realizar traspaso de Backorder de artículo de tramos

IF ART:Bnd\_Tramos THEN CYCLE .

!! Ajustar a Empaques la cantidad Faltante

IF LocF:Cant\_Faltante < ART:FacTraspasos THEN

LocF:Cant\_Faltante = ART:FacTraspasos

ELSE

IF (LocF:Cant\_Faltante % ART:FacTraspasos) = 0 THEN

!! Ya está ajustado a empaques

ELSE

!! Ajusta el faltante al factor de traspasos hacia arriba

LocF:Cant\_Faltante = (INT(LocF:Cant\_Faltante / ART:FacTraspasos) + 1) \* ART:FacTraspasos

END

END

!! analiza después de pasar a factor de traspasos

If LocF:Cant\_Faltante <= 0 Then Cycle .

!! Compara el disponible contra el faltante, para ver cuánto se generará

if LocF:Cant\_Disp <= LocF:Cant\_Faltante

! traspasar todo el excedente del almacén de salida

LocF:Cant\_Trasp = LocF:Cant\_Disp

else

! traspasar solamente el faltante

LocF:Cant\_Trasp = LocF:Cant\_Faltante

end

!! \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

!! Busca en el Queue de Folios

CLEAR(qAlmFolio)

qAlmFolio.qNAlmD = GloqPed.GqNAlm

GET(qAlmFolio, qAlmFolio.qNAlmD)

IF ERRORCODE() THEN

!! buscar el folio

CLEAR(TCMI:Record)

TCMI:NAlm = LocF:AlmCEDIS

TCMI:CConcepto= 5

GET(ElTCmi, TCmi:LLP\_NAlm\_CConcepto)

IF ERRORCODE() THEN

!! Guarda el Mensaje de error

Loc:MenError = 'Ocurrió el error (' & ErrorCode() & '): ' & clip(Error()) & ' al momento de obtener el folio del traspaso.'

!! Termina la Transacción

ROLLBACK

!! Almacena la información en el Queue

GloqPed.GqResult = 'Error Al Obtener Folio Traspaso'

PUT(GloqPed)

!! Muestra mensaje de error original (para evitar el error del rollback)

MESSAGE(Loc:MenError,'Imposible Continuar',ICON:HAND)

!! Termina el proceso

EXIT

END

!! Almacena el folio del traspaso

LocF:LFolioT = TCMI:LFolio

LocF:NFolioT = TCMI:NFolio

!! Actualiza el folio

TCMI:NFolio += 1

put(ElTCmi)

if errorcode() then

!! Guarda el Mensaje de error

Loc:MenError = 'Ocurrió el error (' & ErrorCode() & '): ' & clip(Error()) & ' al actualizar el folio del traspaso.'

!! Termina la Transacción

ROLLBACK

!! Almacena la información en el Queue

GloqPed.GqResult = 'Error Al Asignar Folio Traspaso'

PUT(GloqPed)

!! Muestra mensaje de error original (para evitar el error del rollback)

MESSAGE(Loc:MenError,'Imposible Continuar',ICON:HAND)

!! Termina el proceso

EXIT

End

!! Crear cabecera

CLEAR(TPS:Record)

TPS:LFolio = LocF:LFolioT

TPS:NFolio = LocF:NFolioT

TPS:NAlm = LocF:AlmCEDIS

TPS:FcMov = TODAY()

!TPS:FcRec =

TPS:NAlmDest = PDC:NAlm\_Sal

TPS:Estado = 3 !! Estado en que se encuentra el Traspaso: 0- Sin Confirmar, 1- En Tránsito, 2- Recibido, 3- Asignado

TPS:TipoTsp = 1 !! 1 - Normal, 2 - Tiendas

TPS:OrigenTsp = 'Backorders'

!TPS:BndIncPVta =

TPS:IdUsuario = Das\_GroupUser.NetId

TPS:TotalCapas = 0

TPS:Apl\_PolizaC= false

ADD(ELTPS)

IF ERRORCODE() THEN

!! Guarda el Mensaje de error

Loc:MenError = 'Ocurrió el error (' & ErrorCode() & '): ' & clip(Error()) & ' al crear la cabecera del traspaso.'

!! Termina la Transacción

ROLLBACK

!! Almacena la información en el Queue

GloqPed.GqResult = 'Error En Cabecera de Traspaso'

PUT(GloqPed)

!! Muestra mensaje de error original (para evitar el error del rollback)

MESSAGE(Loc:MenError,'Imposible Continuar',ICON:HAND)

!! Termina el proceso

EXIT

END

!! Añadir al Histórico de Traspasos

CLEAR(HDT:Record)

HDT:LFolio = LocF:LFolioT

HDT:NFolio = LocF:NFolioT

HDT:NAlm = LocF:AlmCEDIS

HDT:Estado = 3 !! Asignado

HDT:IdUsuario\_Reg = Das\_GroupUser.NetId

HDT:FcRegistro = TODAY()

HDT:HrRegistro = CLOCK()

ADD(ELHDT)

IF ERRORCODE() THEN

!! Guarda el Mensaje de error

Loc:MenError = 'Ocurrió el error (' & ErrorCode() & '): ' & clip(Error()) & ' al momento de crear el histórico del traspaso.'

!! Añadir

IF ERRORCODE() = 90 THEN

Loc:MenError = clip(Loc:MenError) & '. Error Nativo(' & FILEERRORCODE() & '): ' & FILEERROR()

END

!! Termina la Transacción

ROLLBACK

!! Almacena la información en el Queue

GloqPed.GqResult = 'Error En Histórico del Traspaso'

PUT(GloqPed)

!! Muestra mensaje de error original (para evitar el error del rollback)

MESSAGE(Loc:MenError,'Imposible Continuar',ICON:HAND)

!! Termina el proceso

EXIT

END

!! añadir al Queue

CLEAR(qAlmFolio)

qAlmFolio.qNAlmD = GloqPed.GqNAlm

qAlmFolio.qLFolioT= LocF:LFolioT

qAlmFolio.qNFolioT= LocF:NFolioT

ADD(qAlmFolio)

SORT(qAlmFolio, +qAlmFolio.qNAlmD)

ELSE

!! Asignar directamente a variable

LocF:LFolioT = qAlmFolio.qLFolioT

LocF:NFolioT = qAlmFolio.qNFolioT

END !! Obtener el Folio a utilizar

!! Crear detalle de traspaso

CLEAR(DTPS:Record)

DTPS:LFolio = LocF:LFolioT

DTPS:NFolio = LocF:NFolioT

DTPS:CArt = DPDC:CArt

DTPS:NAlm = LocF:AlmCEDIS

GET(ELDTPS, DTPS:LLP\_NAlm\_Folio)

IF ERRORCODE() THEN

!! Generar el Detalle del Traspaso

CLEAR(DTPS:Record)

DTPS:LFolio = LocF:LFolioT

DTPS:NFolio = LocF:NFolioT

DTPS:CArt = DPDC:CArt

DTPS:NAlm = LocF:AlmCEDIS

DTPS:Cant\_Original= LocF:Cant\_Trasp

DTPS:Cantidad = LocF:Cant\_Trasp

DTPS:Costo = ART:CtoReposic

!DTPS:CostoCapas =

!DTPS:CostoTienda =

!DTPS:Cantidad\_Rec=

ADD(ELDTPS)

IF ERRORCODE() THEN

!! Guarda el Mensaje de error

Loc:MenError = 'Ocurrió el error (' & ErrorCode() & '): ' & clip(Error()) & ' al momento de crear el detalle del traspaso.'

!! Termina la Transacción

ROLLBACK

!! Almacena la información en el Queue

GloqPed.GqResult = 'Error En Detalle Traspaso'

PUT(GloqPed)

!! Muestra mensaje de error original (para evitar el error del rollback)

MESSAGE(Loc:MenError,'Imposible Continuar',ICON:HAND)

!! Termina el proceso

EXIT

END

ELSE

!! Sumar a la Cantidad del Detalle

DTPS:Cant\_Original+= LocF:Cant\_Trasp

DTPS:Cantidad += LocF:Cant\_Trasp

PUT(ELDTPS)

IF ERRORCODE() THEN

!! Guarda el Mensaje de error

Loc:MenError = 'Ocurrió el error (' & ErrorCode() & '): ' & clip(Error()) & ' al momento de actualizar el detalle del traspaso.'

!! Termina la Transacción

ROLLBACK

!! Almacena la información en el Queue

GloqPed.GqResult = 'Error En Detalle Traspaso'

PUT(GloqPed)

!! Muestra mensaje de error original (para evitar el error del rollback)

MESSAGE(Loc:MenError,'Imposible Continuar',ICON:HAND)

!! Termina el proceso

EXIT

END

END

!! Añadir a la liga de Traspaso-Pedido

CLEAR(ATA:Record)

ATA:LFolio = DPDC:LFolio

ATA:NFolio = DPDC:NFolio

ATA:NPartida = DPDC:NPartida

ATA:CArt = DPDC:CArt

ATA:NAlm = LocF:AlmCEDIS

ATA:NAlmDest = PDC:NAlm\_Sal

ATA:Cantidad = LocF:Cant\_Trasp

ATA:FcMov = TODAY()

ATA:LFolioT = LocF:LFolioT

ATA:NFolioT = LocF:NFolioT

!ATA:FolioOC =

!ATA:Observaciones=

ADD(ELATA)

IF ERRORCODE() THEN

!! Guarda el Mensaje de error

Loc:MenError = 'Ocurrió el error (' & ErrorCode() & '): ' & clip(Error()) & ' al momento de crear la asignación del traspaso.'

!! Termina la Transacción

ROLLBACK

!! Almacena la información en el Queue

GloqPed.GqResult = 'Error En Asignación del Traspaso'

PUT(GloqPed)

!! Muestra mensaje de error original (para evitar el error del rollback)

MESSAGE(Loc:MenError,'Imposible Continuar',ICON:HAND)

!! Termina el proceso

EXIT

END

!! Actualiza el detalle de pedidos, topado a BackOrder

IF LocF:Cant\_Trasp > DPDC:Backorder THEN

DPDC:YaPedidos = DPDC:Backorder

ELSE

DPDC:YaPedidos = LocF:YaPedidos + LocF:Cant\_Trasp

END

Put(ELDPDC)

IF ERRORCODE() THEN

!! Guarda el Mensaje de error

Loc:MenError = 'Ocurrió el error (' & ErrorCode() & '): ' & clip(Error()) & ' al momento de actualizar el detalle del pedido.'

!! Termina la Transacción

ROLLBACK

!! Almacena la información en el Queue

GloqPed.GqResult = 'Error En Detalle Pedido'

PUT(GloqPed)

!! Muestra mensaje de error original (para evitar el error del rollback)

MESSAGE(Loc:MenError,'Imposible Continuar',ICON:HAND)

!! Termina el proceso

EXIT

END

!! Recalcula Pronósticos

!! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

CLEAR(PESA:Record)

Pesa:NAlm = LocF:AlmCEDIS

Pesa:CArt = GloqPed.GqCArt

GET(ELPESA, Pesa:LLP\_NAlm\_CArt)

IF ERRORCODE() THEN

END

!! Existencia Disponible

Pesa:Existencia = LocF:Disp\_AlmOrigen - LocF:Cant\_Trasp !! La existencia en el pesa es en realidad el disponible, al disponible original, quitarle el trasp

Pesa:TraspasosRec += LocF:Cant\_Trasp !! Sumar la cantidad traspasada

!! calcular la compra sugerida

Pesa:Compra\_Sug= 0

if EXA:Clasificacion>=2

!! para arts. nuevos

! checar si la existencia es menor al punto de reorden

if (Pesa:Existencia+Pesa:Cord\_CPra+Pesa:TraspasosRec-Pesa:BKO\_Clientes)<PESA:ROP

! sugerir comprar el faltante para el inv. máximo

Pesa:Compra\_Sug= (Pesa:Inv\_Maximo+Pesa:BKO\_Clientes)-(Pesa:Existencia+Pesa:Cord\_CPra+Pesa:TraspasosRec)

end

! validación por seguridad

if Pesa:Compra\_Sug<0

Pesa:Compra\_Sug= 0

end

else

!! cualquier otro art.

Pesa:Compra\_Sug= 0

case CTRLN:CantCalROP

of 1

if (Pesa:Existencia+Pesa:Cord\_CPra+Pesa:TraspasosRec-Pesa:BKO\_Clientes)>round((Pesa:ROP+Pesa:Inv\_Maximo)/2, 0.01)

!! no se debe comprar nada

! inic. a cero la compra sugerida

Pesa:Compra\_Sug= 0

else

! sugerir compra

Pesa:Compra\_Sug= round((Pesa:Inv\_Maximo+Pesa:BKO\_Clientes)-(Pesa:Existencia+Pesa:Cord\_CPra+Pesa:TraspasosRec), Loc:Redondedo)

end

of 2

if (Pesa:Existencia+Pesa:Cord\_CPra+Pesa:TraspasosRec-Pesa:BKO\_Clientes)>Pesa:ROP

!! no se debe comprar nada

! inic. a cero la compra sugerida

Pesa:Compra\_Sug= 0

else

!! sugerir el máximo posible

! calcular la compra sugerida

Pesa:Compra\_Sug= round((Pesa:Inv\_Maximo+Pesa:BKO\_Clientes)-(Pesa:Existencia+Pesa:Cord\_CPra+Pesa:TraspasosRec), Loc:Redondedo)

end

end

end

!1. costo de la compra

!2. calcular el posible exceso

!3. checar si hay exceso

Pesa:Csto\_Compra= Pesa:Compra\_Sug\*Pesa:Csto\_Unitario

LocF:Exceso = (Pesa:Existencia+Pesa:Cord\_CPra+Pesa:TraspasosRec)-Pesa:BKO\_Clientes

if LocF:Exceso>Pesa:Inv\_Maximo

!! hay exceso

!1. calcular el exceso real contra el máximo

!2. compararlo contra lo ordenado al proveedor

LocF:DifExceso= LocF:Exceso-Pesa:Inv\_Maximo

if LocF:DifExceso>=(Pesa:Cord\_CPra+Pesa:TraspasosRec)

!! es mayor-igual, el exceso es lo ordenado al proveedor

Pesa:Exceso= Pesa:Cord\_CPra+Pesa:TraspasosRec

else

!! es menor, el exceso es la diferencia contra el máximo

Pesa:Exceso= LocF:DifExceso

end

Pesa:Csto\_Exceso= Pesa:Exceso\*Pesa:Csto\_Unitario

else

!! no hay exceso

Pesa:Exceso = 0

Pesa:Csto\_Exceso= 0

end

!! Columna de Maximo de Traspasos

CASE Pesa:Clasif\_Hits

OF 'Super Alto'

Pesa:Maximo\_Traspasos = Round(Pesa:Inv\_Maximo+(CTRLN:DiasMTras\_Super \* Pesa:Dem\_Dia),Loc:Redondedo)

OF 'Alto'

Pesa:Maximo\_Traspasos = Round(Pesa:Inv\_Maximo+(CTRLN:DiasMTras\_Alto \* Pesa:Dem\_Dia),Loc:Redondedo)

OF 'Medio'

Pesa:Maximo\_Traspasos = Round(Pesa:Inv\_Maximo+(CTRLN:DiasMTras\_Medio \* Pesa:Dem\_Dia),Loc:Redondedo)

OF 'Bajo'

Pesa:Maximo\_Traspasos = Round(Pesa:Inv\_Maximo+(CTRLN:DiasMTras\_Bajo \* Pesa:Dem\_Dia),Loc:Redondedo)

Else

Pesa:Maximo\_Traspasos = Pesa:Inv\_Maximo

End

PUT(ELPESA)

IF ERRORCODE() THEN

!! Guarda el Mensaje de error

Loc:MenError = 'Ocurrió el error (' & ErrorCode() & '): ' & clip(Error()) & ' al momento de actualizar los pronósticos.'

!! Termina la Transacción

ROLLBACK

!! Almacena la información en el Queue

GloqPed.GqResult = 'Error En Pronósticos'

PUT(GloqPed)

!! Muestra mensaje de error original (para evitar el error del rollback)

MESSAGE(Loc:MenError,'Imposible Continuar',ICON:HAND)

!! Termina el proceso

EXIT

End

!! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

!! Aumentar Apartados del almacén de origen

EXA:Apartados += LocF:Cant\_Trasp

PUT(ELEXA)

IF ERRORCODE() THEN

!! Guarda el Mensaje de error

Loc:MenError = 'Ocurrió el error (' & ErrorCode() & '): ' & clip(Error()) & ' al momento de actualizar la existencia.'

!! Termina la Transacción

ROLLBACK

!! Almacena la información en el Queue

GloqPed.GqResult = 'Error En Existencia'

PUT(GloqPed)

!! Muestra mensaje de error original (para evitar el error del rollback)

MESSAGE(Loc:MenError,'Imposible Continuar',ICON:HAND)

!! Termina el proceso

EXIT

END

!! Actualizar queue

!! Verificar que no sea negativo

LocF:Cant\_BOFinal = LocF:Cant\_BOOriginal - LocF:Cant\_Trasp

IF LocF:Cant\_BOFinal < 0 THEN

LocF:Cant\_BOFinal = 0

END

GloqPed.GqLFolioT = LocF:LFolioT

GloqPed.GqNFolioT = LocF:NFolioT

GloqPed.GQBOFinal = LocF:Cant\_BOFinal

GloqPed.GqResult = 'Traspasado: ' & LocF:Cant\_Trasp

PUT(GloqPed)

END

!! Indica el Mensaje a Mostrar

Loc:MenError = 'Se muestran los resultados del intento de asignación de traspasos de BackOrders desde CEDIS.'

!! Terminar transacción

COMMIT