GeneXus by Globant

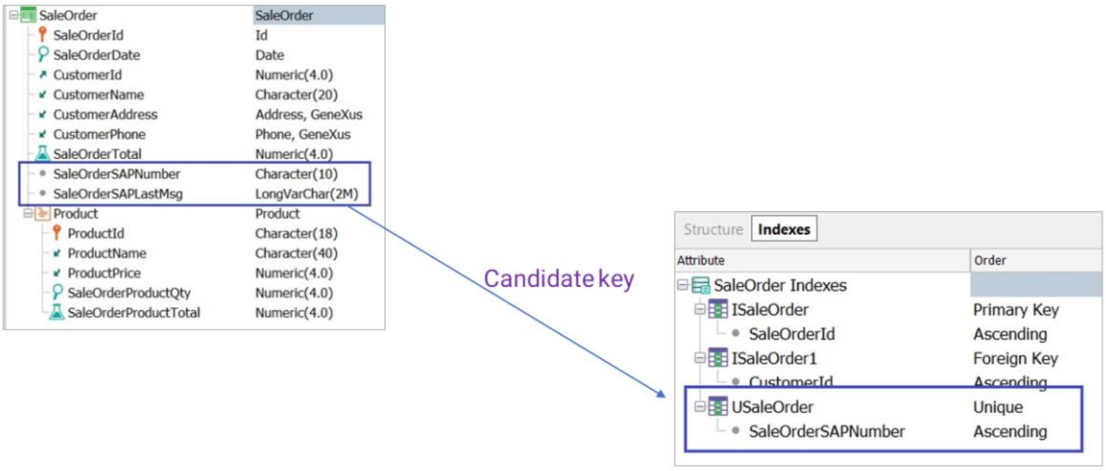
Integrating the KB to SAP ERP

Working with Sales Orders

Avanzando en nuestro desarrollo, vamos ahora a trabajar con lar órdenes de venta. Nuestro objetivo es poder generar un pedido en nuestra aplicación y crearlo en el ERP de SAP.

Si la operación es exitosa se nos devolverá un número de orden de venta creado por SAP. Y en caso de producirse algún problema con la conexión o creación del pedido, devolverá el error generado.

Working with Sales Orders

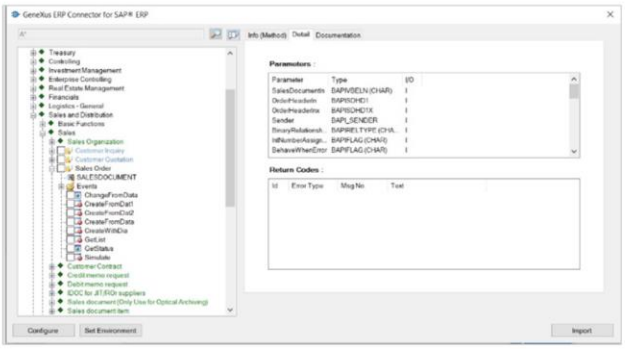


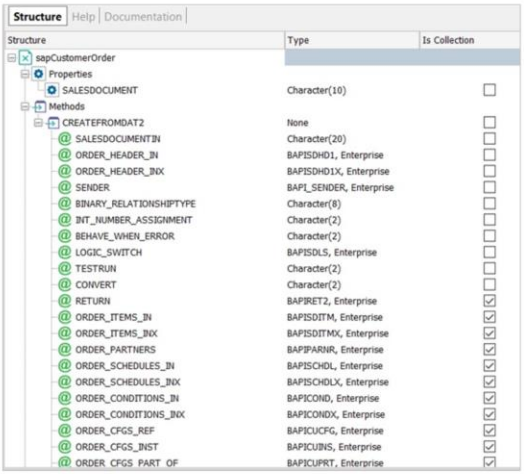
Así que necesitamos agregar un nuevo atributo a nuestra transacción SaleOrder para guardar el número devuelto por SAP. Le llamamos SaleOrderSAPNumber. Y definimos también otro atributo que guardará el mensaje de error en caso de ocurrir alguna falla. Le llamamos SaleOrderSAPLastMsg, de tipo LongVarChar.

Este atributo será una clave candidata en esta transacción. Esto significa que será un valor único que identificará a cada orden de venta, si bien no es la clave primaria definida en la transacción

Así que creamos el correspondiente índice unique en la tabla SALEORDER, sobre el nuevo atributo SaleOrderSAPNumber.

Working with Sales Orders





Bien. Necesitamos ahora importar la BAPI correspondiente para poder crear órdenes de venta en el ERP.

Así que vamos a Tools / Application integration / SAP BAPI Import, y presionamos Ok.

Vamos a Sales and Distribution, Sales, Sales Order, y seleccionamos este Create from. Vemos su información, y la lista de parámetros desde la solapa Detail:.

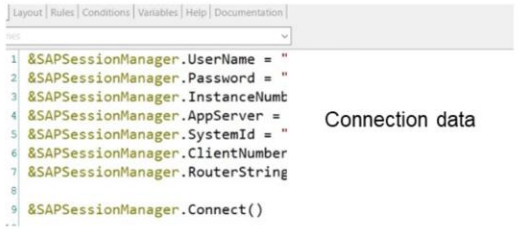
Anteriormente hemos analizado en detalle los parámetros, y objetos a importar por la bapi, por lo que no entraremos ahora en detalle. Importamos.

Y, como ya sabemos, los objetos quedarán en el módulo Enterprise.

Finalizado el proceso, si abrimos el módulo Enterpise vemos el conjunto de nuevos SDTs y un nuevo objeto externo sapCustomerOrder.

Este objeto externo tiene la propiedad SalesDocument, y el método de creación con su conjunto de parámetros.

Working with Sales Orders




```
If &SAPSessionManager.ErrorCode.IsEmpty()
    //Success

    for each SaleOrder
        where SaleOrderId = &SaleOrderId
        //   * Sales Order Header
        &BAPISDHD1.DOC_TYPE    = 'TA'
        &BAPISDHD1.SALES_ORG   = 'UY01'
        &BAPISDHD1.SALES_DIST  = ''
        &BAPISDHD1.DIVISION    = '01'
        &BAPISDHD1.DISTR_CHAN  = '01'
        &BAPISDHD1.DOC_DATE    = SaleOrderDate

        //   * Sales Order Partner
        &BAPIPARNRRow.PARTN_ROLE = 'AG'
        &BAPIPARNRRow.PARTN_NUMB = '000000' + CustomerId.ToString().Trim()
        &BAPIPARNR.Add(&BAPIPARNRRow)

        &SOLine.SetEmpty()
        for each SaleOrder.Product
            &SOLine += 10
            //   * Sales Order Item
            &BAPISDITMRow.ITM_NUMBER = &SOLine
            &BAPISDITMRow.MATERIAL   = ProductId
            &BAPISDITMRow.TARGET_QTY = SaleOrderProductQty
            &BAPISDITMRow.TARGET_VAL = '0'
            &BAPISDITM.Add(&BAPISDITMRow)

            &BAPISCHDLRow.ITM_NUMBER = &SOLine
            &BAPISCHDLRow.REQ_QTY    = SaleOrderProductQty
            &BAPISCHDL.Add(&BAPISCHDLRow)
        endfor
    endfor
```

Al igual que antes, crearemos un procedimiento que recibirá el número de pedido creado por nuestra aplicación, e intentará crearlo en el ERP de SAP. En caso de que la creación resulte exitosa, el procedimiento devolverá el número de SaleOrder devuelto por SAP. Y en caso de existir alguna falla devolverá el error.

El número devuelto por SAP será grabado en el nuevo atributo SaleOrderSAPNumber, y el mensaje obtenido de SAP al momento de creación del pedido será grabado en el atributo SaleOrderSAPLastMsg.

Bien, creamos entonces el procedimiento CreateSaleOrderSAP, y en la solapa Rules definimos la regla Parm para recibir el identificador de la orden de venta.

Definimos en primer lugar los datos de conexión y nos conectamos con el ERP de SAP. Si no hay errores en la conexión, se genera el cabezal de la orden

Estos datos dependen de la implantación de SAP en una empresa. Se crea un Id para el tipo de pedido, en este ejemplo “TA”, se indica la organización de venta, el canal de distribución, etc. Lo mismo ocurre con el Cliente:'AG' es el código que indica que esto es un Cliente.

Luego en este For each, se recorre cada línea de la orden de venta, y para cada línea se crea una para el producto y otra para la entrega

Working with Sales Orders

```
&SAPSessionManager.TransactionBegin()

&sapCustomerOrder.CREATEFROMDAT2(&SALESDOCUMENTIN,&BAPISDHD1, &BAPISDHD1X, &BAPI_SENDE

&SAPSessionManager.TransactionCommit()


&SaleOrderSAPNumber.SetEmpty()

for &BAPIRET2Row in &BAPIRET2
    if &BAPIRET2Row.TYPE <> 'S' and &BAPIRET2Row.NUMBER = 311
        // Save the document number created
        &SalesOrderSAPNumber = &BAPIRET2Row.MESSAGE_V2
    endif
endfor

if not &SaleOrderSAPNumber.IsEmpty()
    for each
        where SaleOrderId = &SaleOrderId
        SaleOrderSAPNumber = &SaleOrderSAPNumber
    endfor
else
    for each
        where SaleOrderId = &SaleOrderId
        SaleOrderSAPLastMsg = &BAPIRET2.ToXml()
    endfor
endif
```

Bien, luego para crear o modificar un dato en el ERP de SAP a través de una BAPI, se debe utilizar al método TransactionBegin y TransactionCommit. Algunas BAPIs no realizan commit a la base de datos, por lo que debe dispararse en forma explícita.

En el medio se realiza la creación del pedido, y para eso debemos utilizar la variable &sapCustomerOrder, definida en base a la BAPI ya que se trata de un método de instancia.

Luego se analiza la respuesta de la BAPI para obtener el número de SAP, y se guardan los valores correspondientes en los atributos SaleOrderSAPNumber y SaleOrderSAPLastMsg

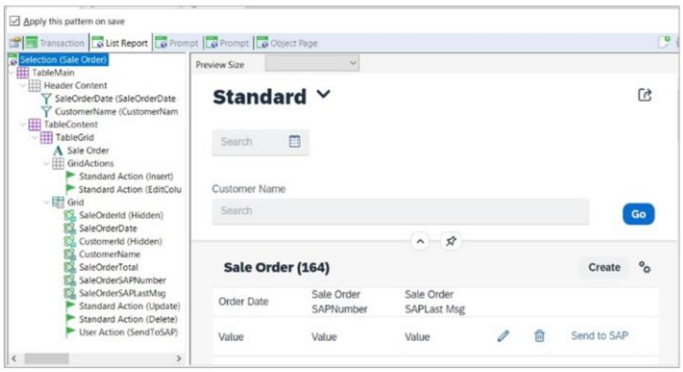
Finalmente se cargan los errores en caso de haberse generado.

Working with Sales Orders

```
CreateSaleOrderSAP(SaleOrderId) if Insert on AfterComplete;

noaccept(SaleOrderSAPNumber);

noaccept(SaleOrderSAPLastMsg);
```



Bien, Una vez finalizado este procedimiento, debemos llamarlo desde las reglas de la transacción SaleOrder de forma tal que para cada nueva orden que se genera, y luego de grabarse en las tablas de nuestra aplicación, lo mismo se cree en el ERP de SAP.

Para eso condicionamos la llamada al modo de ejeución insert de la transacción y lo condicionamos al momento on AfterComplete, o sea, luego de realizado el commit de la transacción.

Recordemos que le habíamos aplicado el floorplan ListReport al aplicar el pattern Fiori a esta transacción. Así que vamos a la solapa Pattern, y en el tab ListReport, agregamos los nuevos atributos a nivel del grid. Y agregamos también una user action que llamamos SendToSAP

Definimos esta acción como una imagen, y para eso marcamos la propiedad Control Type como Image. Le asignamos el nombre y también el código de un ícono

Working with Sales Orders

```
Event &SendToSAP.Click
    if SaleOrderSAPNumber.IsEmpty()
        CreateSaleOrderSAP(SaleOrderId)
    endif

    Grid.Refresh()

EndEvent
```

Event Grid.Load

```
if SaleOrderSAPNumber.IsEmpty()
    &SendToSAP.TooltipText = 'Send SO to SAP'
else
    &SendToSAP.TooltipText = 'SO alreday in SAP'
Endif
```

Abrimos el objeto, y vamos a los eventos. En el evento asociado, y en caso de que el atributo SaleOrderSAPNumber esté vacío, llamamos al procedimiento de creación de pedido en SAP.

De esta forma, si por algún motivo se genera algún error al crear la orden en SAP, se puede reintentar el envío en cualquier otro momento.

Si vamos ahora al evento Load, podemos cambiar el tooltip text, de acuerdo al estado del pedido, o sea, si está creado en SAP o está pendiente de enviar.

Bien. Para ver en ejecución, presionamos F5.

Vemos las órdenes de venta, habilitadas para ser enviadas. Enviamos una de ellas, y vemos el número devuelto por SAP.

Podríamos también, por ejemplo, cambiar el color de fondo según el estado, cambiar los íconos, etc.

GeneXus by Globant

GeneXus™
by Globant

training.genexus.com