“SAPping”

Notas técnicas de SAP - Tip en detalle Nro. 09

(Lo nuevo, lo escondido, o simplemente lo de siempre pero bien explicado)

SAPScript: Análisis de las alternativas para encarar modificaciones y agregados en formularios

Tema:	Impresión, SAPscript, “Enhancements”, Perform in Program, Smartforms, Impresión, Exits.
Descripción:	Analizar las distintas alternativas que existen para implementar modificaciones en los formularios, ventajas y desventajas. Cómo minimizar/evitar cambios en el programa de impresión estándar.
Nivel:	Intermedio.
Versión:	4.6
Fecha pub:	Octubre 2002

Tabla de contenido

I. Overview del procedimiento de impresión de formularios

SAPScript es el mecanismo previsto originalmente por SAP para generar formularios impresos, y, a pesar de la aparición de Smartforms para mejorar esta funcionalidad (ver apartado final), sigue siendo la herramienta más utilizada.

Cuando se va a imprimir cualquier documento, SAP ejecuta el **programa de impresión ("print program")** previsto a tal fin, que inicializa el proceso, recolecta y ordena los datos del formulario, e interactúa con SAPScript para derivarle la tarea de "formateo". Todos los documentos generados por SAP, facturas, cheques, recibos, etc. toman su aspecto y estructura desde las definiciones contenidas en los llamados "**layout sets",** o coloquialmente "formularios SAPScript"**.** SAP provee la herramienta para crear y modificar "layout sets" llamada **SAPScript Editor**.

Los programas de impresión son complejos programas ABAP. Los "layout set", en cambio, son en sí mismos archivos de texto, que contienen definiciones de ventanas, módulos de texto predefinidos, espacio reservado para datos variables, y elementos de formato (boxes, shading). El programa y el layout set interactúan a través de una componente de SAPScript llamada **composer** que combina los datos fijos con los variables y libera el documento al spool una vez completado.

Los print programs se comunican con los "layout set" a través de las funciones disponibles para este fin, y utilizan **estructuras de datos** o tablas, en algunos casos, para pasar al formulario el contenido de los campos.

Obviamente, SAP incluye un "layout set" estándar para cada documento a imprimir, de manera que un usuario no necesita crear "layout sets" desde cero. Desde ya, también provee SAP los llamados "print program" para generar la información de cada uno de los documentos e interactuar desde ABAP con el "layout set" correspondiente.

La customización de los formularios es una parte esencial de cualquier proceso de implementación de R/3 para adaptarlo a las necesidades del cliente, tanto si se trata de mínimos retoques como un rediseño mayor.

En muchos casos, las modificaciones de un formulario pueden resolverse dentro de los alcances del SAPScript editor. Los clientes trabajan con el editor para adecuar la presentación del formulario, reubicando líneas, campos, ventanas, leyendas en cabeceras pie y detalle, incluyendo logos, fonts, códigos de barras etc. Estas modificaciones son de bajo impacto en el sentido de que **NO** involucran a los programas de impresión, y **no requieren programación ABAP**.

Pero también en muchos casos, se requieren cambios que exceden un 'formateo' en la presentación. Por ejemplo, imprimir campos que no están en la estructura original del formulario, imprimir descripciones extraídas de una tabla, cambiar el agrupamiento o la apertura de las líneas de detalle, etc. **Estas modificaciones NO podrán lograrse mediante cambios al formulario desde el SAPScript editor**. Será necesario valerse de otros recursos que van desde agregar campos en la estructura, invocar subrutinas ABAP, utilizar User Exits o Customer Functions, hasta la situación más "traumática", que es modificar el programa de impresión. A pesar de que en muchas instalaciones se toma esta última opción como práctica frecuente, la realidad es que SAP recomienda tratar de evitarlo.

Alternativas para implementar modificaciones en un formulario

Vale la pena analizar cuáles son los distintos instrumentos para implementar las modificaciones en formularios, conocer a fondo ventajas y desventajas, para luego discutir la mejor conducta a seguir con cada formulario.

(a) SAPScript EDITOR: Cubre cambios puros al layout set: limitado sólo al ajuste de presentación, inclusión de logotipos, inclusión o exclusión de campos ya presentes en las estructuras de datos.

(b) RECURSOS DE "ENHANCEMENT" (AMPLIACIONES): SAP dispone de un conjunto de recursos para permitir ampliaciones "prolijas" al código estándar, ("enhancements"). SAP denomina "enhancement" a la funcionalidad NO desarrollada en el estándar, ni asequible a través del customizing, pero contemplada por SAP como un requerimiento potencial. Para esto, el código estándar puede prever

"exits" (salidas) dentro de los programas, de manera que los clientes puedan insertar codificación adicional sin "mezclarla" con el código desarrollado por SAP. También prevé recursos para extender tablas y estructuras de datos sin modificar los formatos originales. Todos los cambios implementados mediante "enhancements" están protegidos en los futuros upgrades, dado que SAP garantiza que las interfaces serán válidas en los próximos releases. **Algunos de estos recursos de "enhancement" están disponibles también en los programas de impresión y sus estructuras de comunicación**, y son un recurso válido para encarar modificaciones en los formularios.

(c) PERFORM IN PROGRAM: SAPScript ofrece a partir de la versión 3, la posibilidad de invocar rutinas ABAP desde el formulario, sin tocar el programa de impresión. Es muy poderosa también esta herramienta, pero debe usarse con cuidado dado que puede tener impacto sobre la performance.

(d) CAMBIOS EN EL PRINT PROGRAM: Por último, cuando las dos alternativas anteriores resultan insuficientes para implementar los cambios necesarios, existe el último recurso de crear una versión Z del programa de impresión, y trabajar los cambios desde este nivel. En general, caen en esta categoría las modificaciones del orden de impresión, los cortes de control, o el nivel de apertura de las líneas de detalle. Los programas de impresión son por lo general largos y complejos y SAP trata de desalentar las modificaciones en los mismos. Sin embargo, a veces es inevitable. Con SAPScript, la lógica del formulario, la recuperación de datos y la impresión están entrelazadas en un mismo programa ABAP o module pool. Por ejemplo, el orden en que las áreas de salida se imprimen queda establecida por el orden de la función WRITE_FORM del print program, y no puede cambiarse sin tocarlo.

(e) SMARTFORMS: A partir de la versión 4.6B, aparece como opción una herramienta alternativa a los formularios de SAPScript: los **SmartForms.** Aquellas instalaciones que estuvieran en condiciones de implementar SmartForms podrían lograr de manera mucho más sencilla cambios que antes obligaban a modificar el programa de impresión. A diferencia de SAPScript, los formularios definidos con SmartForms controlan también la lógica de impresión del documento, por fuera del programa ABAP. (Ver nota sobre SmartForms al final del artículo). Por esta razón, es que en el tiempo SmartForms terminará convirtiéndose en el mejor método para definir formularios complejos, y SAP dejará de introducir mejoras y actualizaciones a SAPScript.

La decisión de cuál de estos recursos utilizar no es inmediata, ni es posible establecer una regla. En cada caso debería considerarse factores como el tipo de modificación a implementar, la disponibilidad de recursos de enhancement tipo CUSTOMER FUNCTION, INCLUDE o USER EXITS para el formulario en cuestión , el impacto que tiene para ese cliente una ampliación o modificación del estándar, las restricciones de performance que puede introducir el PERFORM, etc. En líneas generales, se suele recomendar recorrer estas instancias en el orden que aquí las presentamos, es decir, tratar de resolverlo dentro del SAPScript editor, buscar opciones de "enhancements" o recurrir al PERFORM IN PROGRAM cuando esto no fuera posible, y reservar la decisión de modificar el programa de impresión cuando no hubiera otra alternativa. En el punto V incluimos algunas pautas para ayudar en esta decisión.

Analizaremos a continuación en detalle las alternativas (b) (Recursos de Enhancement) y (c) (Subrutinas ABAP), precisamente porque están orientadas a evitar los cambios en el programa de impresión ABAP para una buena parte de los casos. Presentamos también un mismo ejemplo implementado con cada uno de estos dos enfoques.

II. Facilidades de "enhancement" (ampliaciones) previstos por SAP

SAP dispone de un conjunto de recursos para permitir ampliaciones "prolijas" al código estándar, ("enhancements"). SAP denomina "enhancement" a la funcionalidad **NO desarrollada en el estándar**, ni asequible a través del customizing, pero **contemplada por SAP como un requerimiento potencial**.

Para esto, el código estándar puede prever **"exits" (salidas/bifurcaciones**) dentro de los programas, para que el cliente pueda insertar codificación adicional sin que ésta se "mezcle" con el código desarrollado por SAP. También prevé recursos para **extender tablas y estructuras de datos** sin perturbar los formatos originales. Todos los cambios implementados mediante "enhancements" están protegidos en los futuros upgrades, dado que SAP garantiza que las interfaces serán válidas en los próximos releases.

Algunos de estos recursos de "enhancement" están disponibles también en los programas de impresión y sus estructuras de comunicación. Por lo tanto, **antes de intentar una modificación directa de la lógica del programa de impresión, debería explorarse la posibilidad de aprovechas algún recurso de enhancement**.

Recursos de "Enhancement" aplicables a formularios

Supongamos que se desea imprimir el campo Período Contable (POPER) en la cabecera de la factura de SD. Este campo no está definido en el formulario, pero además, no integra la estructura de comunicación de datos para la cabecera de factura, VBDKR, que el programa de impresión le envía al formulario.

Como en este caso, la mayor parte de las modificaciones a un formulario **requiere el agregado de campos en la estructura de comunicación**, para hacerlos disponibles al formulario. Asimismo, se requerirá ingresar código ABAP, en alguna parte de print program, para darle valor a esos campos antes de que se transfiera el control al composer.

Extensión de tablas

Para extender tablas o estructuras sin modificar su estructura básica, existen en SAP dos posibilidades. *Si Ud. no está familiarizado con el uso de las mismas, sugerimos leer al pie de este documento una descripción más detallada.*

- **Componentes .INCLUDE**: Son campos **ya creados por SAP** en la tabla o estructura, preparados para referenciar otra estructura de datos **creada por el cliente,** con los campos adicionales. Los .INCLUDE previstos para "enhancements" del cliente están inicialmente "vacías", es decir, referencian un campo "dummy". Si el cliente crea una estructura para esa componente .INCLUDE, cuando se active la tabla, SAP combinará los campos de la estructura INCLUDE con los campos originales de la tabla. Una tabla puede incluir una o más componentes .INCLUDE. (SAP de hecho se vale de algunas estructuras .INCLUDE para implementar funciones de customizing y localizaciones). A la vez, una estructura .INCLUDE puede estar referenciada por más de una tabla. *Por ejemplo, la estructura de comunicación VBDKR utilizada para la cabecera de factura contiene una componente .INCLUDE con nombre VBDKRZ, preparada por SAP para agregar campos. En el caso comentado anteriormente, podría agregarse el campo ZZPOPER a través de este .INCLUDE, asignándole el elemento de datos POPER.*
- **Componentes .APPEND:** Las componentes .APPEND funcionan de forma similar, excepto que **NO están pre-planeadas en la tabla por SAP**. Permiten agregar campos a casi cualquier tabla sin modificar su estructura. El cliente debe agregar la componente .APPEND a la tabla y luego crear la estructura de datos asociada. Las estructuras APPEND también difieren de las .INCLUDE en que **sólo tienen validez para la tabla en la que se definen**, es decir, no son reusables a través de distintas tablas. Cuando se activa una tabla con una o más componentes .APPEND, SAP agrega los campos definidos en las estructuras APPEND como si se fueran campos de la misma tabla. *En el caso anterior, también podría utilizarse este recurso para agregar el período contable en la estructura de la cabecera de factura. Cuando no existen componentes .INCLUDE previstas por SAP, puede usarse una componente .APPEND para agregar el dato.*

Agregado de código ABAP

Además de hacer disponible el/los campos dentro de la estructura, es necesario prever su tratamiento dentro del programa para darles contenido. Para insertar código desarrollado por el cliente sin alterar la lógica básica del programa, SAP ha previsto en algunos casos **puntos de salida** dentro del código estándar a través de:

- **User-Exits:** Consiste en un recurso de ampliación usado básicamente en el módulo de SD, y considerado hoy algo rudimentario. Consisten en llamados a subrutinas ya implementados por SAP en puntos estratégicos del programa. Además, SAP crea un program include especial en el module pool que corresponda, que contienen una o más subrutinas que satisfacen la convención de nombres **userexit_<nombre>.** Estas subrutinas se encuentran vacías para que el usuario las llene con su propio código. El objetivo es que todos los cambios se almacenen fuera del código fuente original. *En el caso comentado de la cabecera de factura, se sabe que existen tres USER-EXITS dentro del*

programa de impresión. Uno de ellos, el program include V05NZZRK, se ejecuta al leer la cabecera del documento en el momento en el que se llenan los campos de la estructura VBDKR con los datos de la tabla VBRK correspondiente . Allí debería incluirse

MOVE VBRK-POPER TO VBDKR-ZZPOPER.

- **Customer Functions:** Son llamados a módulos de función ya pre-planeados dentro del estándar (CALL CUSTOMER FUNCTION), donde el código de dicha función será creado por el cliente. SAP decide dónde se incluyen las CUSTOMER-FUNCTIONS, definiendo además su interfaz, texto descriptivo, documentación y propósito. Las funciones de cliente están inactivas en le sistema original y se activan cuando se activa el "proyecto" de enhancement al cual están asociadas. Los customer functions son una herramienta más moderna y mucho prolija que los User-Exits.

Todos los recursos mencionados pueden combinarse para implementar modificaciones complejas, sin necesidad de modificar la estructura básica del programa de impresión.

Limitaciones de las técnicas de "enhancement"

Desafortunadamente, **las posibilidades de ampliación a través de las facilidades mencionadas, no son uniformes a lo largo y a lo ancho de SAP R/3.** Dependiendo del módulo en cuestión, de la época en que se desarrolló, puede que en el print program haya USER EXITS o CUSTOMER FUNCTIONS previstos, o ninguna de las dos cosas; puede que haya campos .INCLUDE en las tablas o estructuras de comunicación, pero muchos formularios no las tienen. No se puede enunciar una regla general sobre cómo encarar la modificación de un formulario. **Es necesario relevar cuáles de estos recursos pueden estar disponibles para el formulario que se necesita modificar, y analizar si soporta la modificación a efectuar.**

Algunos ejemplos de recursos de ampliación:

- (SD) la mayoría de los formularios de SD tienen previstos campos .INCLUDE en las estructuras de comunicación tanto de cabecera como de detalle, y luego tienen tres USER EXITS en el print program para procesar los campos agregados. (Puede encontrar un detalle de estructuras y EXITS de formularios de SD en el SAP Reference Implementation Guide; → SAP Reference IMG → Sales and Distribution → System modifications → User exits →Choose relevant component).
- (FI) El programa de impresión de factura de FI(J_1A007) tiene un CUSTOMER FUNCTION para poder modificar datos de las estructuras J_1AI02, SADR y DKADR, pero ninguna de las estructuras tiene .INCLUDE de usuario. Podría recurrirse a los .APPEND para extender las estructuras y así lograr cambios importantes SIN modificar el estándar en modo alguno. (Los campos .APPEND sí pueden utilizarse siempre, porque no necesitan estar pre-planeados en el estándar)
- (MM) El programa de impresión de vales de acompañamiento de mercancías (MM) tiene Customer Functions en varios lugares, y la posibilidad de ampliar la estructura de comunicación AM07M mediante el Include CI_AM07M. La ampliación para SAP es la MBCF0005.

Tal vez la única regla válida es que conviene evaluar primero todas las opciones de modificación de formularios antes de tomar una decisión.

Asimismo, es necesario tener en cuenta que a veces es engorroso buscar y utilizar los User-Exits o Customer Functions disponibles, si no se tiene el entrenamiento necesario.

Otras limitaciones asociadas a los recursos de enhancements, consisten en que obviamente no permitirán implementar modificaciones de secuencia, ordenamiento, etc. Estas últimas requieren indefectiblemente de la modificación del print program, o bien, replantear la elección de la herramienta pensando en SmartForms.

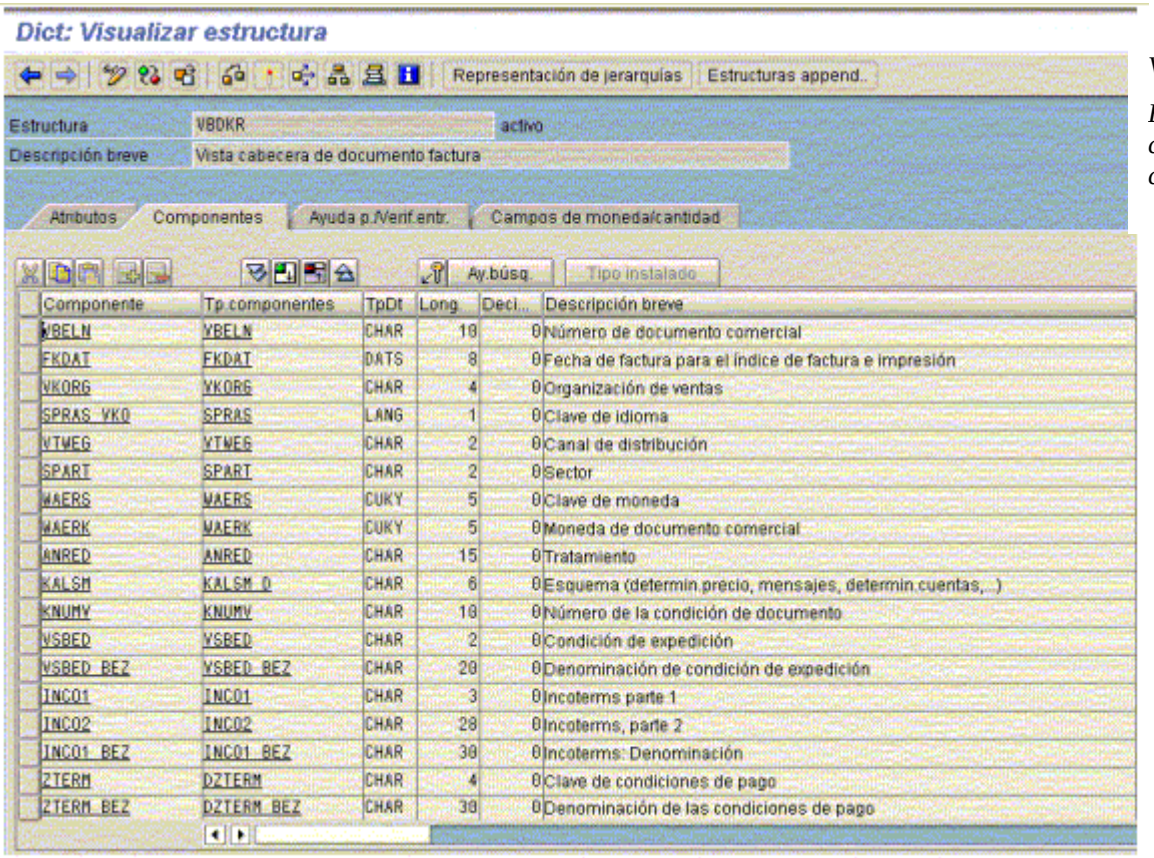
Ejemplo desarrollado .APPEND y USER EXITS

A continuación presentamos un ejemplo de modificación de un formulario, utilizando recursos de los descriptos anteriormente.

En este caso, se trata de agregar en la cabecera de factura de SD, un texto descriptivo para el código de moneda. La descripción de la moneda (KTEXT) se debe obtener de la tabla correspondiente (TCURT) a partir del código de moneda (WAERS). El formulario es el estándar *RVINVOICE01* (Release 4.6C.)

Como primer paso, es necesario generar el campo dentro de la estructura de datos que recibe el formulario con los campos de la cabecera de factura. Se trata de la estructura de datos VBDKR.

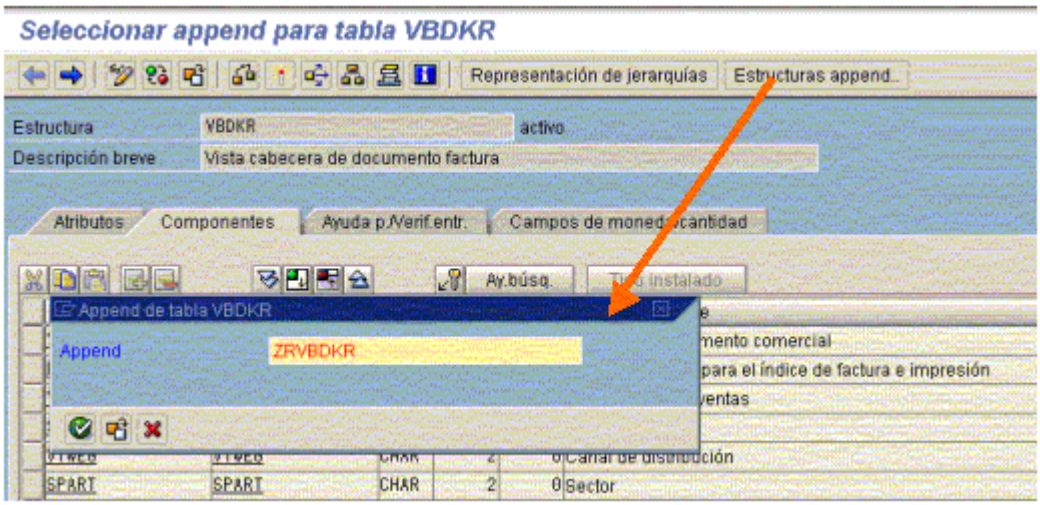
IMPORTANTE: Esta estructura tiene además una componente .INCLUDE VBDKRZ que permitiría agregar el campo en cuestión, pero en este ejemplo utilizaremos el recurso de componentes .APPEND para extender la estructura. Recordar que las componentes .APPEND sólo tienen valor para la tabla en la cuál se definen, y no necesita estar el campo pre-planeado por SAP.



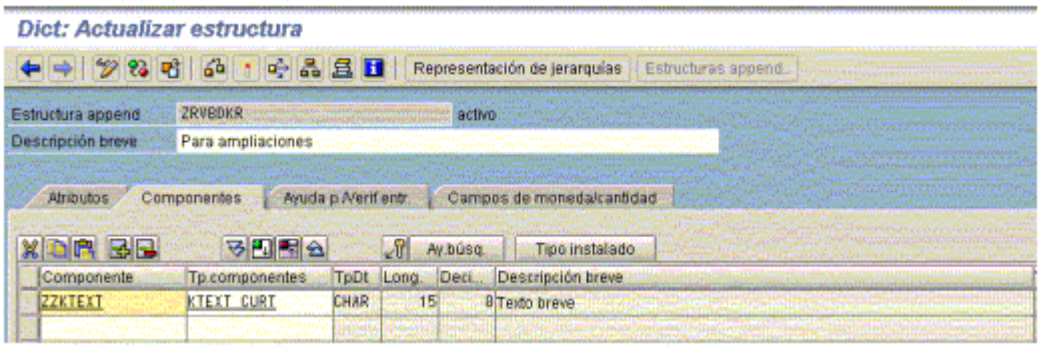
VBDKR

Estructura de comunicación para cabecera de factura

Crearemos entonces una estructura .APPEND con nombre *ZRVBDKR* para esta tabla. El nombre de la estructura APPEND, al igual que el de los campos que contenga, debe estar dentro del rango de nombres de usuario (ZZ o YY)



Luego, dentro de la estructura APPEND, se crea el campo ZZKTEXT para contener la descripción de la moneda.

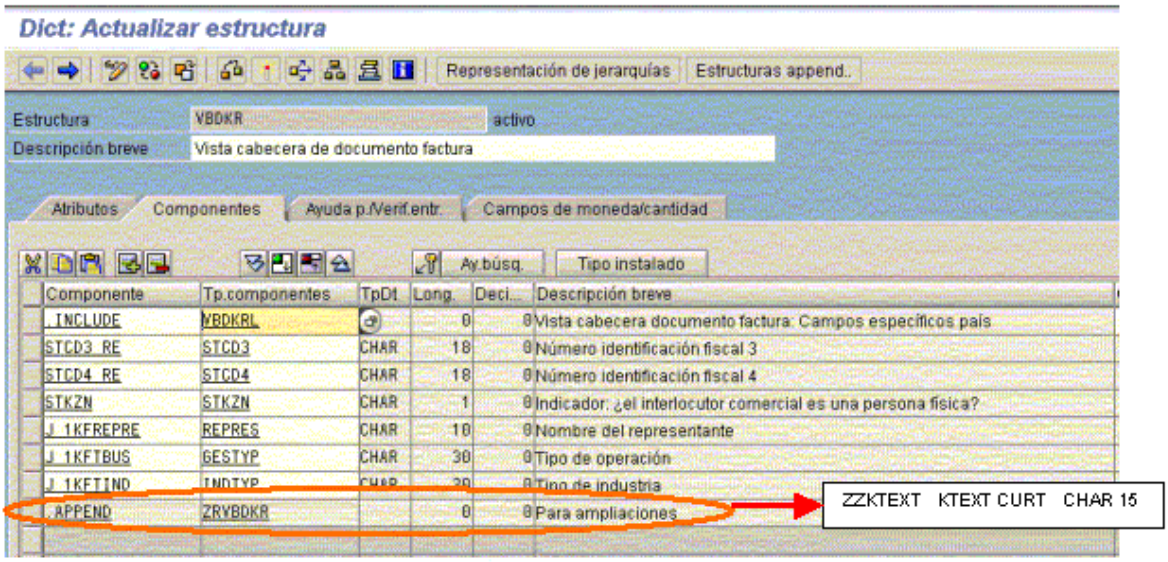


Al salvar y activar la estructura, el log indicará

Activación de tabla ZRVBDKR con éxito.

Estructura APPEND añadida a la tabla VBDKR.

La tabla queda:



A continuación, debe preverse el código ABAP necesario dentro del programa de impresión, para darle valor al campo agregado antes de componer la impresión. Para ello, pueden utilizarse los USER EXITS previstos por SAP.

Como se mencionara anteriormente, la mayoría de los print programas de SD tienen tres USER EXITS disponibles. La lista de todos los user exits se puede encontrar en el IMG (SAP Reference Implementation Guide).

En nuestro ejemplo, el programa estándar de impresión de la factura de SD, *RVADIN01*, contiene el "program include" *V05NZZRK* en el que se llena la estructura VBDKR.

Desde el programa principal, se accede al include mediante las subrutinas *PROCESSING, GET_DATA*, el módulo de función RV_BILLING_PRINT_VIEW y las subrutinas LESEN_VBDKR_VBRK_01 y MOVE_VBRK_TO_VBDKR.

```
***INCLUDE V05NZZRK.

MOVE-CORRESPONDING VBRK TO VBDKRZ.
MOVE-CORRESPONDING VBDKRZ TO VBDKR.
```

User exit para la cabecera de factura SD

Y se completan los campos de la nueva estructura ZRVBDKR: para ello se accede a la tabla de base de datos *TCURT* con el indicador de idioma, tomado de la variable del sistema *SY-LANGU* y el código de moneda del documento *WAERK* disponible en la tabla de cabecera de factura VBRK.

```
***INCLUDE V05NZZRK.

*   INSERT           T20K903404
TABLES: TCURT.
*   INSERT

MOVE-CORRESPONDING VBRK TO VBDKRZ.
MOVE-CORRESPONDING VBDKRZ TO VBDKR.

*   INSERT           T20K903404
* Obtención de la descripción de la moneda:
SELECT SINGLE * FROM TCURT WHERE SPRAS = SY-LANGU
                             AND   WAERS = VBRK-WAERK.
IF SY-SUBRC = 0.
  VBDKR-ZZKTEXT = TCURT-KTEXT.
ENDIF.

*   INSERT
```

Finalmente, el nuevo campo *ZZKTEXT* contiene la descripción de la moneda, tomada del campo *KTEXT* de la tabla *TCURT* y a partir de esta modificación se encuentra disponible para el formulario.

III. La alternativa PERFORM in Program

A partir de la versión 3 SAPScript ofrece un recurso valioso que consiste en la posibilidad de **invocar una rutina ABAP directamente desde el "layout set"** a través del comando de control PERFORM.

Las subrutinas invocadas desde aquí pueden usarse para ejecutar cálculos, tomar datos de la base de datos, etc. Además de evitar en muchos casos el manipuleo del programa estándar, este recurso permite lograr una cohesión de la subrutina dentro del formulario, de manera que si el mismo layout set es usado en diferentes programas, la subrutina será siempre la misma independientemente del programa de impresión.

Cuando se define un formulario, cada una de las ventanas tiene un área de edición de código donde se cargan los campos que imprime dicha ventana, y/o los comandos de control SAPScript que sean necesarios para armar la salida. Es en este área de trabajo donde se utilizaría el PERFORM.

Como todos los comandos de control SAPScript, el PERFORM se ejecuta cuando el formulario se formatea para impresión o visualización por pantalla. La comunicación de parámetros y variables entre el documento y la subrutina se materializa mediante **símbolos** como es habitual en la programación de formularios (Ver cuadro sobre SAPScript Symbols).

La sintaxis del comando PERFORM es la siguiente:

```
/: PERFORM <form> IN PROGRAM <prog>
/: USING &Invar1&
/: USING &Invar2&
...
/: CHANGING &Outvar1&
/: CHANGING &Outvar2&
...
/: ENDPERFORM
```

Invocación de la subrutina:

El siguiente ejemplo muestra como utilizar PERFORM desde un SAPScript. Al igual que hicimos con el ejemplo anterior, en este caso se ha usado este recurso para incluir en el formulario una descripción de la moneda más extensa que la que prevé SAP en forma estándar.

```
/* Busca descripción ampliada de moneda
/: DEFINE &moneda& = ''
/: PERFORM Busca_Desc_Mon IN PROGRAM ZRF00001
/: USING &Regud-Waers&
/: CHANGING &Moneda&
/: ENDPERFOM
```

&Regud-Waers& y &Moneda& son símbolos tipo texto de 80 caracteres. &Regud-Waers& se envía a la subrutina y &Moneda& recibe el valor para ser impreso en el formulario.

La subrutina

La subrutina ABAP que es invocada por SAPScript es como cualquier otro programa ABAP excepto que debe prever el intercambio de parámetros establecido en el SAPScript llamador. Todos los parámetros enviados a la subrutina con la cláusula USING son recibidos en una **tabla interna** que tiene el mismo formato que la estructura de diccionario **ITCSY**. (NAME: char, 32, Nombre del text symbol; VALUE: char, 80, Valor del text symbol).

Cada parámetro enviado es un registro en dicha tabla y éstos deben recuperarse exactamente en la misma secuencia con que fueron declarados en el USING. En caso de que se leyeran menos registros de los requeridos, la subrutina NO cancelará sino que ignorará el parámetro faltante.

Para devolver el valor en los parámetros declarados en el CHANGING, se utiliza el mismo mecanismo, esto es, deben cargarse en una tabla interna con el formato ITSCY, uno por registro, y en el mismo orden en que los espera el formulario. Análogamente al caso anterior, si se devuelven menos registros que los requeridos por el formulario, los parámetros faltantes quedarán sin valor alguno en el formulario. Debe prestarse especial atención a esto para evitar problemas posteriores.

En este ejemplo toma de la invocación un solo parámetro, el código de moneda &Regud-Waers& recibido en la tabla interna *Mon*. La rutina devuelve a través de la tabla interna *Desc* el valor tcurt-ktext que recupera de la base de datos, y el formulario lo recibe en el símbolo &Moneda&.

```
   REPORT ZRF00001

** Rutina que recupera datos para los formularios de FI.
   TABLES: Tcurt.
   FORM Busca_Descrip_Moneda
     TABLES: Mon  STRUCTURE ITCSY
             Desc STRUCTURE ITCSY.
     DATA:   Moneda LIKE Tcurt-Waers.

** Lee la clave para ingresar al archivo de monedas
     READ TABLE Mon INDEX 1.
     Moneda = Mon-Value.

** Busca la descripción de la moneda
     SELECT SINGLE * FROM tcurt
       WHERE Spras = SY-LANGU AND Waers = Moneda.

** Asigna valor para el formulario
     IF SY-SUBRC = 0.
       READ TABLE Desc INDEX 1.
       Desc-Value = Tcurt-Ktext.
       MODIFY Desc INDEX 1.
     ENDIF.
   ENDFORM.
```

Limitaciones del Perform in Program

La invocación de subrutinas como las comentadas en este tip **incrementa considerablemente** los tiempos de procesamiento del formulario. Debe evaluarse el impacto que esto puede tener para la instalación donde se utiliza. Es importante NO agregar esta clase de código si realmente no se necesita o puede ser reemplazado por otras técnicas.

Al igual que los recursos de enhancements, el Perform in Program no puede resolver cambios de ordenamiento.

IV. Modificaciones al programa de impresión

Cómo se mencionó al principio, cuando las dos alternativas anteriores resultan insuficientes para implementar los cambios necesarios, existe el último recurso que es alterar el programa de impresión. En general, caen en esta categoría las modificaciones del orden de impresión, los cortes de control, o el nivel de apertura de las líneas de detalle, es decir, todos aquellos cambios estructurales y también cuando **es necesario agregar elementos de textos.**

También existen instalaciones que no tienen una política tan estricta en cuanto a modificar el estándar, y prefieren abordar cambios en el programa de impresión, antes que los posibles inconvenientes de usar “enhancements”, o PERFORM in PROGRAM.

Los principales inconvenientes asociados a cambiar el programa de impresión tienen que ver con:

a) El skill requerido a nivel de ABAP para manipular programas largos y complejos.

b) Las posibles mejoras que SAP introduzca en dicho programa no se incorporarán al programa modificado por el cliente salvo que se lo haga manualmente. Asimismo, será necesario revisar el funcionamiento de los programas modificados cada vez que se actualiza el release.

Para implementar modificaciones al print program, **siempre** se lo debe copiar utilizando el editor ABAP del Workbench ABAP y asignando un nombre al programa copiado que comience con Z o Y, es decir, respetando la convención de nombres de clientes. Luego se debe editar el programa de copia Z, efectuar los cambios necesarios y activar el programa.

Por último se debe asociar el nuevo programa de impresión al formulario correspondiente, a través de las funciones de customizing. Por ejemplo, para asignar programas de impresión a formularios dentro de SD, se debe ingresar a:

Sales and Distribution → Basic Functions → Output control →Output Determination →Output Determination Using the Condition Technique → Maintain Output Determination for Sales Documents → Maintain Output Types.

Luego se debe seleccionar un output type y dar doble click en Output programs , para asignar el programa de impresión y el formulario a un tipo de mensaje.

V. Discusión sobre la elección de una alternativa

Como ya expresáramos anteriormente**, la decisión de cómo encarar una modificación mayor a un formulario no es inmediata, y requiere de un análisis de determinados factores**. Teniendo en cuenta las posibilidades de cada técnica aquí descriptas, puede ayudar también el planteo de los siguientes puntos:

- Considerar realmente si es necesario cambiar el programa o si los cambios pueden resolverse alterando el layout set con el SAPScript editor.
- Tratar de tener cabal conciencia de cuáles son los campos adicionales que se necesitan en el formulario. Están esos campos presentes en las tablas o estructuras usadas, o se necesitan tablas adicionales?. Si los campos están presentes, se ha definido contenido para los mismos?
- Existen enhancements disponibles en la forma de User Exits o Customer Functions?
- Puede obtenerse la información que se necesita desde otro programa usando PERFORM IN PROGRAM?
- Existen limitaciones de perfomance en la ejecución del formulario que pudiera verse afectado por el Perform In Program?
- Cuál es la política de desarrollo en ese cliente para introducir cambios y ampliaciones?. Se trabaja con enhancements habitualmente?. Se ha modificado el estándar de alguna otra manera.?

En algunos casos, puede recurrirse también a soluciones mixtas, ya que las técnicas aquí discutidas no son excluyentes entre sí. (Por ejemplo, pueden utilizarse componentes .APPEND o .INCLUDE para extender tablas aunque se modifique el print program y se utilice el PERFORM IN PROGRAM para tratar los campos).

VI. Para tener en cuenta

Es importante considerar que el layout set es dependiente del mandante, a diferencia del programa de impresión y de las rutinas invocadas mediante el perform in program.

VII. Conceptos ampliados

SapScript Symbols

Los símbolos SAPScript son referencias o marcadores utilizados para representar aquellos elementos del texto que son variables, es decir, que tomarán valor en el momento de la resolución final del documento. Dentro de un "layout set", cualquier referencia a datos se ingresa mediante SAPScript Symbols, y se trabaja con ellos de una forma similar a como se trabaja con variables.

Existen cuatro tipos distintos de symbols, dependiendo de dónde obtienen éstos su valor en la ejecución: los program symbols representan datos que provee el programa ABAP en ejecución cuando se resuelve el formulario; los text symbols toman valor de los comandos de control SAPScript dentro del texto mismo o por la función Include; los system symbols son reemplazados directamente por SAPScript y los standard symbols toman su valor de la tabla TTDTG. SAP determina en forma automática el tipo de un símbolo según el contexto.

Todos los símbolos pueden contener texto únicamente, de hasta 80 posiciones.

Es importante destacar que a partir de la versión 4 la comunicación entre el programa de impresión y el formulario admite diferentes tipos de datos (hasta la versión 3 sólo se podía disponer en el formulario, de aquellos campos correspondientes a las estructuras del diccionario ABAP provistas por SAP para cada tipo de programa, es decir, no se podían utilizar las variables del programa). Dentro de los tipos de datos permitidos se encuentran las tablas, estructuras, parámetros, opciones de selección, tablas internas, etc.

Componentes de tablas .INCLUDE (También llamados Customizing Includes)

En algunos casos, SAP ha previsto la posibilidad de extender la estructura de una tabla con campos específicos del cliente, a través de las componentes .INCLUDE. Los campos .INCLUDE aparecen listados como campos al visualizar una tabla o estructura, pero no son más que referencias o punteros hacia una estructura independiente. Cuando se activa una tabla que contiene un componente include, se mezclan las estructuras como si se tratara de una sola.

Esta facilidad permite extender tablas y estructuras del sistema sin necesidad de modificar la definición de las mismas. Esto significa que las mejoras y cambios introducidos de esta manera no se perderán durante un Upgrade. Cuando una estructura de R/3 esta modificada con campos del cliente a través de componentes .INCLUDE, éstos se agregan automáticamente a la nueva estructura creada durante el upgrade.

A diferencia de las estructuras append que se explican más adelante, un include se puede usar en múltiples tablas o estructuras, lo cual garantiza la consistencia de los objetos del diccionario afectados, cuando éste se modifica. También a diferencia de las componentes .APPEND, los INCLUDES tienen que estar previstos (pre-planeados) en la estructura.

La creación de .INCLUDES se hace a través de las funciones de customizing. SAP se vale también de componentes .INCLUDE para implementar funciones de customizing, o para las localizaciones, por lo que pueden aparecer campos .INCLUDE ya creados con contenido en tablas estándar.

Los nombres de los campos creados por el cliente dentro de la estructura INCLUDE siguen las reglas de nomenclatura de objetos de clientes (ZZ YY)

Componentes .APPEND

Las estructuras APPEND permiten ampliar tablas o estructuras agregando campos que no son parte del estándar, sin tener que modificar la tabla o estructura en sí misma. Se crean para ser utilizadas sólo en una tabla o estructura específica. Cuando se activa una tabla o estructura, el sistema activa todas las estructuras append asociadas. Si una estructura append es creada o modificada y luego activada, la tabla o estructura correspondiente también se activa y todas las modificaciones realizadas a las estructuras append toman efecto en el objeto al que pertenecen. Los campos de las estructuras append se utilizan en los programas ABAP como cualquier otro campo de la tabla o estructura original. Una tabla puede tener múltiples estructuras append asignadas.

Como toda ampliación, las estructuras append están protegidas frente a futuros upgrades. Las nuevas versiones de tablas y estructuras estándares se cargan durante los upgrades y los campos contenidos en estructuras append activas se incorporan a los nuevos elementos del diccionario cuando estos se activan por primera vez.

Los nombres de campo de las estructuras append deben respetar los estándares de nombres de clientes, es decir, deben comenzar con "YY" o "ZZ", para evitar posibles conflictos de nombres ante la posibilidad de que SAP inserte nuevos campos.

User Exits

Los "user exits" son un tipo de ampliación del sistema creados con el propósito de permitir al usuario realizar sus propias modificaciones. Fueron desarrollados originalmente para el módulo de SD (Ventas y Distribución) y se consideran técnicamente una "modificación", dado que alteran objetos dentro del espacio de nombre de SAP.

Normalmente SAP organiza los programas como un conjunto de programas "includes" para facilitar su comprensión. En el caso de los user exits, se crea un include especial en un module pool, con el objetivo de que todos los cambios se almacenen fuera del código fuente del programa. Los includes respetan la convención de nombres para programas y grupos de funciones y garantizan de esta forma que los desarrolladores SAP no toquen este include en el futuro y que no se ajuste durante los upgrades.

Los includes contienen una o más subrutinas que satisfacen la convención de nombres userexit_<nombre> y que se encuentran vacías para que el usuario las llene con su propio código. Las llamadas a estas subrutinas ya han sido implementadas en el programa R/3, su interfaz ya fue definida y generalmente sólo utilizan variables globales. Después de desarrollar estos includes, SAP nunca los altera, si un nuevo user exit debe ser desarrollado en un nuevo release, estos se ubican en un nuevo include.

La lista de todos los user exits se puede encontrar en el IMG (SAP Reference Implementation Guide).

Customer Functions (Function Module Exits)

Los customer Functions son en realidad parte de los llamados Customer Exits, es decir, salidas para el cliente. Técnicamente se denominan Function Module Exits.

Son llamados a módulos de función ya pre-planeados dentro del estándar (CALL CUSTOMER FUNCTION), donde el código de dicha función será creado por el cliente. SAP decide dónde se incluyen las CUSTOMER-FUNCTIONS, definiendo además su interfaz, texto descriptivo, documentación y propósito. Las funciones de cliente están inactivas en le sistema original y se activan cuando se activa el "proyecto" de enhancement al cual están asociadas. Los customer functions son una herramienta más moderna y mucho prolija que los User-Exits.

Los programadores SAP crean las ampliaciones utilizando la transacción SMOD.

Los clientes disponen de un catálogo de las ampliaciones SAP existentes y pueden combinar estos enhancement en un "**proyecto de ampliación**" mediante la **transacción CMOD**. Los

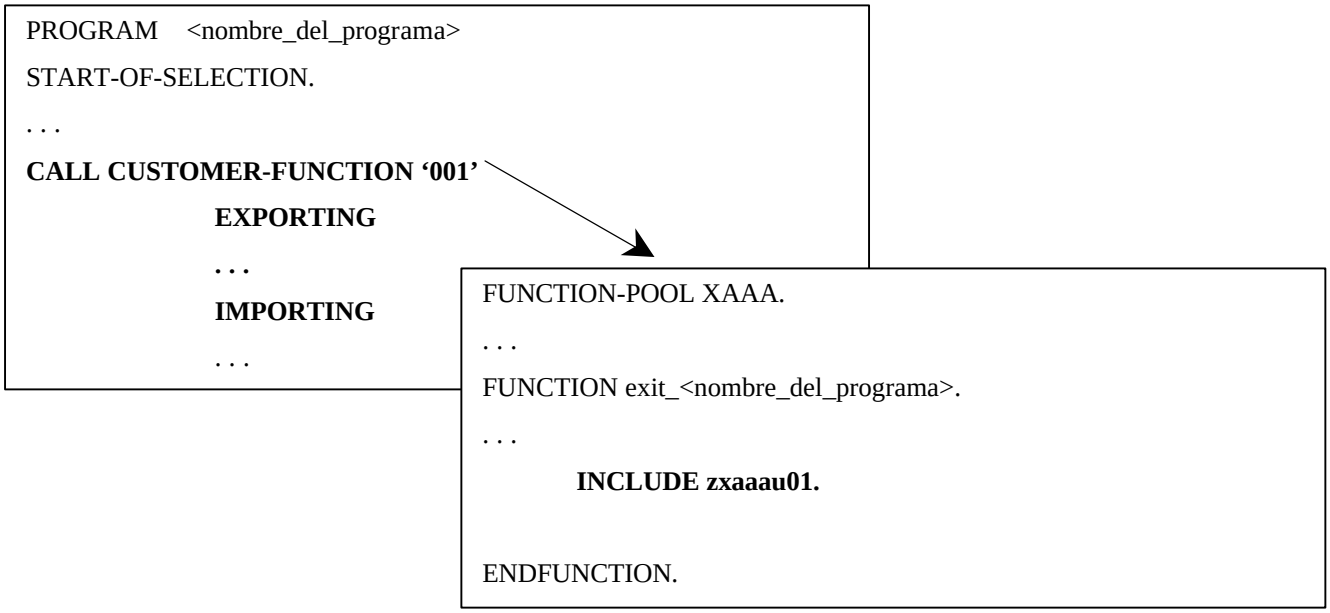
proyectos consisten de ampliaciones y estas de “componentes”. Cada componente es un exit de módulo de función, menú o pantalla.

Para crear, entonces, una ampliación se debe ingresar a la transacción CMOD, crear un proyecto, asignar las ampliaciones al proyecto (se pueden utilizar las funciones de búsqueda para obtener la lista de las ampliaciones existentes) y editar el componente. Dependiendo del componente seleccionado, el sistema nos conduce a la Biblioteca de Funciones, a una ventana para ingresar entradas de menú o al Screen Painter. Por último, una vez editado el componente, el proyecto debe ser activado (la activación afecta a todos los camponentes del proyecto). **Recién después de la activación, se ve el efecto de las ampliaciones en la aplicación**.

Cuando el proyecto de ampliación ha sido creado, de debe asignar a una orden de transporte. Se aconseja que cada componente (programas include, subscreen, menú, etc) se asigne a la misma orden, para asegurarse que la ampliación completa se transporte al mismo tiempo.

En el caso de los “function module exits”, los programadores SAP **definen dónde las funciones son insertadas y qué clase de datos se transfieren**. También crean un módulo de función completo con su descripción breve, su interfaz y su documentación. **El cliente escribe el código fuente para la función mediante un include, que respeta la convención de nombres de cliente (comienza con Z).**

Invocación y creación del módulo de función:



Smartforms

Smart Forms es la nueva solución de SAP para impresión de formularios, sucesor de SapScript. Todos los nuevos desarrollos de formularios, SAP los llevará a cabo usando esta herramienta. SAPScript seguirá siendo soportado, desde ya, para todos los formularios ya creados de esta manera.

A partir de la versión 4.6C, los clientes pueden utilizar tanto SAPScript como Smartforms, aunque SAP recomienda esto último para nuevos desarrollos.

La estructura básica de SmartForms abarca el Smart Form Builder, el Smart Form print Template, el Smart Form Function Module y el SmartForm Print Program. (Los print programs que usan Smart Forms NO son los mismos print programs de SAPScript, y no puede utilizarse uno en lugar del otro.)

Mas allá de las diferencias operativas entre una y otro herramienta, la diferencia más importante consiste en su arquitectura. **Smart Forms no sólo permite manejar la presentación del formulario sino también la lógica de procesamiento de los datos**. Como resultado de las definiciones creadas o editadas en el Form Builder, SmartForms genera un módulo de función que maneja TODO el procesamiento del formulario: campos requeridos, condiciones, recolección de datos, secuencia y ordenamiento, y por supuesto, presentación y componentes gráficos.

Por lo tanto, muchos cambios que con SAPScript requerían programación ABAP y modificaciones en el print program, para Smart Forms serán tan sólo cambios en las definiciones del Builder.

VIII. Dónde Obtener Información Adicional

Changing the SAP Standard (BC)

SAP Smart Forms (BC-SRV-SCR)

BC SAPScript: PRINTING with FORMS

IMPORTANTE

*Dudas o consultas: **sapping@teknoda.com***