Wolters Kluwer

a3ERP

Guía: Objetos a3ERP ACTIVE X

Introducción

La lista de objetos disponibles es la siguiente:

- **Enlace12:** Inicia y cierra la conexión con la base de datos de una empresa nexus. Mantiene las variables de error.
- **Enlace**: Inicia y cierra la conexión con la base de datos de una empresa nexus. Mantiene las variables de error.
- **Varios1201**: Objeto con varias funciones destinadas a: Obtención de precios, obtención de cambio entre monedas, obtención de identificadores únicos de nexus, obtención de cuentas y descripciones y llamadas a las distintas opciones visuales de nexus.
- **Varios**: Extensión del anterior con nuevos métodos disponibles desde versión 12.02 en adelante. Se añaden métodos para crear y borrar cierres de IVA \ IGIC.
- **Maestro**: Artículos, Proveedores, Clientes, Cuentas, Bancos, Almacenes, Representantes, Transportistas, Provincias, Países, Tarifas, Idiomas, etc... . Este objeto permite moverse por los ficheros mencionados, así como añadir nuevos y modificar los existentes.
  El maestro Contactos ya no existe a partir de versiones 12.00.0.0, en su lugar existe el maestro __Contactos.
- **Seleccion**: Este objeto permite presentar la pantalla de selección estándar de nexus, para mostrar los registros de cualquier tabla con el formato código, descripción, etc.
- **Factura**: Objeto que permite la creación, modificación y borrado de facturas.
- **Albaran**: Objeto que permite la creación, modificación y borrado de albaranes.
- **Depósito**: Objeto que permite la creación, modificación y borrado de depósitos.
- **Pedido**: Objeto que permite la creación, modificación y borrado de pedidos.
- **Oferta**: Objeto que permite la creación, modificación y borrado de ofertas (presupuestos).
- **Regularizacion**: Objeto que permite la creación, modificación y borrado de albaranes de regularización.
- **Traspaso**: Objeto que permite la creación, modificación y borrado de albaranes de traspaso entre almacenes.
- **Inventario**: Objeto que permite la creación de y modificación de inventarios.
- **MovimientoStock**: Objeto que permite la creación, modificación y borrado de líneas que generan movimientos en el stock por documentos no controlados por nexus.
- **ReservaStock**: Objeto que permite la creación, modificación y borrado de líneas que generan reservas en el stock por documentos no controlados por nexus.
- **Asiento**: Objeto que permite la creación, modificación y borrado de asientos. (
- **Cartera**: Objeto que permite la creación, modificación y borrado de vencimientos en cartera.
- **OperacionesCartera**: Objeto que permite realizar operaciones sobre los efectos en cartera a excepción de agrupaciones y remesas. Permite por ejemplo cobrar, pagar, devolver, recibir, etc...
- **Agrupacion**: Objeto que ayuda a gestionar las operaciones con las agrupaciones de efectos de cartera.
- **Remesa**: Objeto que ayuda a gestionar las operaciones con las remesas de efectos de cartera.
- **Opcion**: Objeto que permite llamar a cualquier opción de la aplicación nexus.

- **OpcionModal**: Objeto que permite llamar a determinadas opciones en modo modal. Devuelve un código de resultado. (Es operativa en sólo en: acciones de CRM, expedientes, cuotas y documentos de compra y venta.)
- **Listado**: Objeto que permite realizar informes pasando los parámetros deseados directamente.
- **OperacionesInmovilizado**: Objeto que permite realizar operaciones sobre el inmovilizado así como amortizar y repercutir las amortizaciones de las cuotas.
- **Presupuesto**: Permite crear, borrar o modificar el contenido de la contabilidad presupuestaria.
- **Estructura**: Permite crear, borrar o modificar el contenido de una estructura.
- **OrdenProduccion**: Permite crear, borrar, modificar o evolucionar una orden de producción.
- **Expediente**: Permite crear, borrar, modificar expedientes. Tambien facturarlos e imprimirlos**.**
- **Cuota**: Permiter crear, borrar y modificar cuotas.
- **CobroParcialRemesa**: Permite cobrar parcialmente una remesa.
- **PagoParcialRemesa**: Permite pagar parcialmente una remesa.
- **AnularCobroParcialRemesa**: Permite anular el cobro de una remesa (cobrada parcialmente).
- **AnularPagoParcialRemesa**: Permite anular el pago de una remesa (pagada parcialmente).
- **Acciones:** Permite generar acciones de tipo e-mail, cita y otras.
- **InmuebleCliente:** Permite indicar los inmuebles de un cliente.
- **InmuebleProveedor:** Permite indicar los inmuebles de un proveedor.
- **Se comparten con erp Windows y crm.** Si se tiene activado la sincronización con Outlook también se tendrá en cuenta a la hora de sincronizar.
- **EnlaceA3DOC:** Permite gestionar los elementos publicados en a3DOC.
- **Objeto Maestro DOMBANCA.** Gestionar los mandatos de las domiciliaciones de cliente cuando estemos conectados con a3ASESOR.
- **MaestroA3ASESOR.** El objeto implementa además de los metodos de la interfaz IMaestro, los propios de la interfaz IMaestroA3ASESOR. Deben usarse únicamente en aplicaciones integradas con a3ASESOR.
- **Agrupación:** Permite crear, borrar y modificar agrupaciones de efectos.
- **Asiento:** Permite crear, borrar y modificar asientos.
- **Traspaso:** Permite la creación, modificación y borrado de traspasos.
- **ContactoRelacion:** Permite crear, borrar y modificar las relaciones de un contacto.
- **Oportunidades**: Permite la creación, modificación y borrado de objetos oportunidad.
- **Oportunidad**: Información de la oportunidad más su conjunto de líneas.
- **LíneaOportunidad**: Permite la creación, modificación y borrado de líneas de oportunidad.
- **VinculosOportunidad**: Permite la creación, modificación y borrado de vínculos asociados a una oportunidad.
- **Servir**: Permite servir líneas de documentos. Debe usarse si hay doble unidad de stock.
- **ServirLineaParametros**: Permite indicar la información de la línea a servir.
- **OpcionesServir**: Permite indicar opciones a la acción de servir.
- **Servir2**: Permite servir documentos de forma completa.

- **LineaDetalle**: Permite indicar los detalles de una línea de documento. Debe usarse si hay doble unidad de stock.
- **LineaDetalleParametros**: Permite indicar la información de la línea de detalle.

A tener cuenta: **Desde versión 14**, todos los objetos implementan la interfaz **IMensajes**, para anular / habilitar el mostrado de mensajes visuales que puedan aparecer al operar con un objeto (por ejemplo, alarmas, avisos nif incorrecto, ect…) concreto.

Esta interfaz se compone de dos funciones:

FijarNivel(*Nivel: NivelMensaje***)** – El valor se ignora por el momento. Deshabilita los mensajes visuales. Funciona en modo pila, es decir, si se llama varias veces, para volver a habilitar se deberá llamar el mismo número de veces a AnularNivel.

AnularNivel – Habilita los mensajes visuales

A partir de la **versión 14.3**, el objeto Enlace lo implementa de modo global. Es decir, si se deshabilita en el objeto enlace se deshabilita durante toda la sesión (cualquier objeto). Si se habilita, se habilita globalmente (salvo que en un objeto concreto se haya deshabilitado).

A continuación, se detalla cada uno de los objetos:

1.- OBJETO ENLACE1450

Interface Enlace12

Propiedades	**Tipo**	**Descripción**
bError	Lógico	True si la última operación provocó un error.
Nerror	Entero	Código del último error (0 ningún error)
Estado	EstadoEnlace	Indica si la conexión con nexus se ha realizado.
SMensaje	Texto	Descripción del último error (en caso de error)
EmpresaActiva	Texto	Nombre de la empresa activa en nexus
RaiseOnException	Lógico	Para indicar si el lenguaje que se utiliza propaga las excepciones y desea aprovecharse esta ventaja. Disponible para Delphi. Desactivado en Visual Basic.
VentanasAbiertas	Integer	Indica nº de ventanas nexus abiertas
Usuario	Texto	Indica el usuario logado en a3ERP
ConexionSistema	Texto	Indica lista de parámetros de conexión a la base de datos de sistema
Conexion	Texto	Indica lista de parámetros de conexión a la base de datos
VerBarraDeProgreso	Lógico	Indica si se tiene que ver la barra de progreso mientras se inicializa en a3ERPActiveX

RutaDea3ERPActiveX	Texto	Indica la ruta donde está físicamente el fichero a3ERPActiveX.dll
VersionA3ERP	Texto	Devuelve la versión actual de a3ERP.
CodigoUsuarioValido	Texto	Devuelve el número de usuarios con ese código. Útil para verifcar si un usuario es válido en a3ERP.

EstadoEnlace (Tipo enumerado)		
	Valor	**Significado**
EstNOACTIVO	0	Objeto no activo. No se puede usar.
EstACTIVO	1	Objeto activo, se puede trabajar con él.
EstCONECTADO	2	Objeto activo y usuario logado correctamente.

Procedure Iniciar (sEmpresa, sUbicacion:String)		
Inicia la conexión con la empresa indicada.		
Parámetros:	sEmpresa	Nombre exacto de la empresa (distingue mayúsculas y minúsculas).
	sUbicacion	Obsoleto

Procedure Acabar
Termina la conexión con la empresa indicada y libera los recursos reservados.

Function ParamConexion (sEmpresa, sUbicacion: WideString);		
Retorna los parámetros de la conexión de la base de datos de la empresa seleccionada. Muy útil para crear una conexión directa con esa base de datos para consultas directas desde el programa que utiliza A3ERPACTIVEX.		
Parámetros:	sEmpresa	Nombre de la empresa
	sUbicacion	Obsoleto
Valor retornado	Variant que representa un array	[0] → Descripción de la empresa. [1] → Obsoleto [2] → Tipo de datos: MSSQL [3] → Servidor en el que se encuentra el sistema [4] → Nombre del servidor [5] → Nombre de la base de datos [6] → Obsoleto

Function Empresas: OleVariant;		
Retorna la relación de empresas del sistema por defecto con el que se encuentra conectado el terminal.		
Valor retornado	Variant que representa un array	[0]→ Número de elementos del array (-1 indica que ha habido un error) [1]→ Primer nombre de empresa. [2]→ Segundo nombre de empresa … [n]→ N-ésimo nombre de empresa

Function SelecEmpresa: WordBool		
Muestra la selección de empresas de nexus. Espera a que el usuario seleccione una empresa e inicia una conexión con su base de datos.		
Valor retornado	Lógico	Indica si se ha seleccionado la empresa

Function SelecEmpresaActiva: WordBool		
En nexus, cuando se entra en una empresa, se guarda su nombre en el registro. Esta función recupera la empresa indicada en el registro e inicia una conexión con su base de datos.		
Valor retornado	Lógico	Indica si se ha seleccionado la empresa

Function SelecMultiEmpresaActiva: WordBool		
En nexus configurado como multiempresas, cuando se entra en una empresa (multiempresa), se guarda el tipo contable (empresa) en el registro. Esta función recupera el tipo contable (empresa) indicada en el registro y activa dicho tipo contable.		
Valor retornado	Lógico	Indica si se ha seleccionado el tipo contable (empresa)

Function SelecMultiEmpresa: WordBool		
En nexus configurado como multiempresa, podemos activar el tipo contable (empresas) dentro de los tipos contables permitidos por el usuario.		
Valor retornado	Lógico	Indica si se ha seleccionado el tipo contable (empresa)

Procedure EliminarRecordatorioClave
Elimina el flag de recordar la contraseña del registro de Windows

Function EstacionActivada(const NumSerie: WideString):WordBool		
Parámetros:	NumSerie	Código de producto de tercero (6 primeros dígitos del número de serie)
Retorna si el número de serie de lPRODUCTO NEXUS está activado para la estación.		
Valor retornado	Lógico	Indica si la estacion está activada para algún número de serie del producto indicado (de un mismo producto se pueden registrar varios números de serie)

Procedure ActivarVentana
Pone en primer plano la ventana activa de nexus

Procedure CerrarVentanas
Cierra todas las ventanas abiertas de nexus

Function LoginUsuario(sUsuario, sPassword): WordBool		
Parámetros:	sUsuario	Usuario nexus
	sPassword	Password usuario nexus
Esta funcion la utilizaremos para evitar la pantalla visual donde se piden el usuario y password de entrada en nexus, y retorna si dichos valores son correctos para entrar en la aplicación.		
Valor retornado	Lógico	Indica si el usuario y password es correcto

Function ModulosActivos: OleVariant		
Esta función nos devuelve una lista de los modulos que están activos.		
Valor retornado	OleVariant	Nos devuelve una estructura donde el primer elemento indica cuantos módulos nos está retornando la función y a partir del segundo elemento tendremos el nombre del primer módulo.

Function ListaMaestros: OleVariant		
Esta función nos devuelve una lista de todos los ficheros maestros.		
Valor retornado	OleVariant	Nos devuelve una estructura donde el primer elemento indica cuantos maestros nos está retornando la función y a partir del segundo elemento tendremos el nombre del primer maestro.

Function GetConexionDB(const Nombre: WideString): OleVariant;		
Parámetros:	Nombre	Admite dos valores: “SISTEMA” o “EMPRESA”
Valor retornado	OleVariant	Nos devuelve el objeto conexión a la BBDD indicada. Es necesario saber que no se debe cerrar la conexión del objeto devuelto ya que cerraría la conexión de la BBDD. El objeto es del tipo ADOCONNECTION.

Function Versiona3ERP: WideString;		
Parámetros:	Sin parámetros	
Valor retornado	Texto	Esta función devuelve la versión del ejecutable de a3ERP, en formato MajorVersion.MinorVersion.Release.build Todo numéros separados por puntos

Function PermisosOpcion(opcion: widestring): Permisos;		
Parámetros:	Opcion, string	Opción de a3ERP.
Valor retornado	Enumerado Permisos -	Esta función devuelve los permisos que tiene el usuario para una opción de a3ERP. Para ver los posibles resultados, ver el enumerad ‘Permisos’. Como usuario se valida el que se utilizó al iniciar sesión en a3ERPActiveX

Permisos (Tipo enumerado)		
	Valor	**Significado**
permTodo	0	El usario tiene permisos totales en la opción
permSoloLectura	1	El usario puede entrar en la opción y ver la información, pero no puede realizar cambios/altas.
permNada	2	El usario no tiene permisos, ni siquiera de acceso.

Interface Enlace: Enlace12		
Propiedades	**Tipo**	**Descripción**

Gama	LongWord (32bits sin signo)	Gama registrada en la instalación:
		gamaPlus = $00000003;
		gamaProfesional = $00000002;
		gamaOne = $00000001;
		gamaNada = $00000000;
		gamaFreeware = $00000005;
		gamaBase = $00000007;
		gamaPremium = $00000008;
		gamaAsesorBusiness = $00000009;

Function ObtenerProgranasExternos(EntidadID: widestring): IIterador;		
Parámetros:	EntidadID, string	Tabla ERP para la que se hayan registrado programas externos en el mantenimiento de programas externos de la aplicación.
Valor retornado	IIterador	Iterador vacío (si no se encuentra activa la opción en datos generales, o no corresponde a la gama o se proporciona una EntidadID vacía) o iterador de elementos Idispatch que implementan IProgramaExterno

2.- OBJETO ENLACE

Interface Enlace: Enlace1450 [Nuevo 14.05.0.0]

Propiedades	Tipo	Descripción
TipoContableActivo	Texto	Cuando tenemos activado el entorno de Multiempresa, devuelve en que empresa (Tipo contable) estamos conectados. Si no estamos en el entorno Multiempresa devuelve el tipo contable seleccionado.
TipoContableActivoNombre	Texto	Cuando tenemos activado el entorno de Multiempresa, devuelve el nombre de la empresa que tenemos seleccionada. Si no estamos en el entorno Multiempresa devuelve el nombre de la empresa.
IDMultiempresa	Entero	Cuando tenemos activado el entorno de Multiempresa, devuelve el identificador de la empresa que tenemos seleccionada. Si no estamos en el entorno Multiempresa devuelve 0.

3.- OBJETO VARIOS

Para minimizar el impacto del añadido de nuevos métodos al objeto Varios, este implementa, a partir de versión 12.02 dos interfaces. La **Varios1201** (versión con los métodos hasta la versión 12.01 inclusive) y la **Varios** (versión que incluye los nuevos métodos añadidos en la 12.02) y que extiende la anterior. La coclase devolverá la nueva interfaz IVarios e implementa, además, la antigua IVarios1201.

En un futuro, cuando en una versión X.Y se añadan nuevos métodos, la Varios actual pasará a llamarse Varios1202 y la nueva quedará como Varios. Las interfaces existentes (en este caso la Varios1201) mantienen los GUIDs por lo que un módulo que haga uso de la versión antigua (use un

TLB de la versión 12.01 o anterior) no requerirá ser recompilado. Eso sí, no tendrá acceso a los nuevos métodos salvo que refresque las referencias al ActiveX más reciente.

Interface Varios1201

```
property Version: Integer   readonly;
property Revision: Integer readonly;
property Parche: WideString readonly;
function ObtPrcVenta(  const sCodArt: WideString; const sCodCli: WideString;
                       const sCodMon: WideString; const sTarifa: WideString;
                       nUnidades: Currency;  const sFecha: WideString): OleVariant;
function ObtPrcCompra(const sCodArt: WideString; const sCodPro: WideString;
                       const sCodMon: WideString; const sTarifa: WideString;
                       nUnidades: Currency; const sFecha: WideString) : OleVariant;
function ObtPrecioCoste(const sCodArt: WideString; const sCodAlm: WideString;
                       const sObtCoste: WideString): OleVariant;
function ObtPrecioMedio(const sCodArt: WideString; const sCodAlm: WideString): OleVariant;
function ObtComis(  const sCodRep: WideString; const sCodArt: WideString;
                      const sCodCli: WideString; nUnidades: Currency; nDescuento: Currency;
                      const sFecha: WideString): OleVariant;
function Cambio(const sCodMon: WideString; const sFecha: WideString): OleVariant;
function CambioMon(const sMonIni: WideString; const sMonFin: WideString;
                       const sFecha: WideString): OleVariant;
function MonedaPrincipal: WideString;
function FormatNum(nDecim: Integer): WideString;
function FormatEdiNum(nDecim: Integer): WideString;
function Redondear(nNumero: Currency; nDecima: Integer): Currency;
function SigId(const sId: WideString): OleVariant;
procedure DescArticulo( const sCodArt: WideString; const sCodIdi: WideString;
                       var sDescArt: WideString; var sTexto: WideString);
function ObtRepresentante(const sCodCli: WideString): WideString;
function CuentaArtV(const sCodArt: WideString; const sCodCli: WideString): WideString;
function CuentaArtC(const sCodArt: WideString; const sCodPro: WideString): WideString;
function TotalLinea(    nUni: Currency; Prc: Currency; desc1: Currency; desc2: Currency;
                       desc3: Currency; desc4: Currency; const sCodMon: WideString;
                       Iva: WordBool; const sTipIva: WideString; const sRegIva: WideString)
                      :Currency;
```

function Vencimientos(bEsCobro: WordBool; sCodCliPro: WideString; nTotMon, nTotDoc: Currency; sForPag, sDocPag, sCodMon, sFecha: WideString): OleVariant; **procedure** LlamarVisual(const OpcionNexus: WideString; Parametros: OleVariant); **procedure** PasarPotACteNuevo(const sCodCli: WideString; const sCodPos: WideString; bPasarVinculos: WordBool; bPasarObserv: WordBool) **procedure** PasarPotACteExis(const sCodCli: WideString; const sCodPos: WideString; bPasarVinculos: WordBool; bPasarObserv: WordBool) **function** EvaluarFiltro(const CodFiltro: WideString; const Alias: WideString): WideString

Interface Varios: IVarios1201 (Disponible desde version 12.02 en adelante)
procedure **CrearCierresIVAIGIC**(const Tipo: WideString; FechaInicio, FechaFin: TDateTime; HacerCierreDocumentos, HacerCierreContable: WordBool); procedure **BorrarCierresIVAIGIC**(const Tipo: WideString; FechaInicio, FechaFin: TDateTime); procedure **RecalcularStocks**(const CodArt, CodAlm: WideString); safecall; procedure **RecalcularStocksYPreciosMedios**(ActualizarCosteDocVenta: WordBool); safecall; function **Stock1Dia**(const Params: WideString): WideString; safecall;

Propiedades	**Tipo**	**Descripción**
Version	Texto	Devuelve la versión de la librería
Revision	Texto	Devuelve la revisión de la librería
Parche	Texto	Devuelve la letra del parche de la librería

Function ObtPrcVenta (sCodArt, sCodCli, sCodMon, sTarifa: String; nUnidades: Double; sFecha: String): Variant;		
Permite obtener el precio de venta dados los parámetros de la función		
Parámetros:	sCodArt	Código del artículo
	sCodCli	Código del cliente
	sCodMon	Código de la moneda
	sTarifa	Tarifa
	nUnidades	Unidades
	sFecha	Fecha
Valor retornado	Variant que representa un array	[0] → Precio (Currency) [1] → Total descuento (Currency) [2][0] → Descuento 1 (Currency) [2][1] → Descuento 2 (Currency) [2][2] → Descuento 3 (Currency) [2][3] → Descuento 4 (Currency)

Function ObtPrcCompra (sCodArt, sCodPro, sCodMon: String; nUnidades: Double; sFecha: String): Variant;		
Permite obtener el precio de compra dados los parámetros de la función		
Parámetros:	sCodArt	Código del artículo
	sCodPro	Código del proveedor
	sCodMon	Código de la moneda
	sTarifa	Tarifa
	nUnidades	Unidades
	sFecha	Fecha
Valor retornado	Variant que representa un array	[0] → Precio (Currency) [1] → Total descuento (Currency) [2][0] → Descuento 1 (Currency) [2][1] → Descuento 2 (Currency) [2][2] → Descuento 3 (Currency) [2][3] → Descuento 4 (Currency)

Function ObtPrcCoste (sCodArt,sCodAlm, sObtCoste: String): Double;		
Permite obtener el precio de coste dados los parámetros de la función		
Parámetros:	sCodArt	Código del artículo
	sCodAlm	Código del almacén
	sObtCoste	Tipo de coste
Valor retornado	Currency	Valor del precio de coste del artículo

Function ObtPrcMedio (sCodArt, sCodAlm: String): Double;		
Permite obtener el precio medio dados los parámetros de la función		
Parámetros:	sCodArt	Código del artículo
	sCodAlm	Código del Almacén
Valor retornado	Currency	Valor del precio medio del artículo en el almacén indicado

Function ObtComis (sCodrep, sCodArt, sCodCli: String; nUnidades, nDescuento: Double; sFecha: String): Variant;		
Permite obtener el porcentaje y el origen de comisión dados los parámetros de la función		
Parámetros:	sCodRep	Código del representante
	sCodArt	Código del artículo
	sCodCli	Código del cliente
	nUnidades	Unidades
	nDescuento	Descuento máximo de ventas, aplicado en %

	sFecha	Fecha
Valor retornado	Variant que representa un array	[0] → Comisión [1] → Margen o Precio (P/M)

Function Cambio(sCodMon, sFecha: String):Double		
Permite obtener el cambio entre una moneda y la moneda principal (EURO) en una fecha indicada.		
Parámetros:	sCodMon	Código de la moneda
	sFecha	Fecha
Valor retornado	Double	Cambio

Function CambioMon(sCodMonIni, sCodMonFin, sFecha: String):Double		
Permite obtener el cambio entre una moneda inicial y la moneda final en una fecha indicada.		
Parámetros:	sCodMonIni	Código de la moneda inicial
	sCodMonFin	Código de la moneda final
	sFecha	Fecha
Valor retornado	Double	Cambio

Function MonedaPrincipal: string		
Retorna el código de la moneda principal de la empresa activa. (siempre EURO)		
Valor retornado	String	Código de la moneda

Function FormatNum: string		
Retorna string que define el formato de un número en Delphi con los decimales indicados en el parámetro.		
Parámetros:	nDecim	Nº de decimales
Valor retornado	String	Formato

Function FormatEdiNum: string		
Retorna string que define el formato de un número para su edición en Delphi con los decimales indicados en el parámetro.		
Parámetros:	nDecim	Nº de decimales
Valor retornado	String	Formato

Function Redondear(nNumero:Double; nDecima:Integer):Double;		
Utiliza la misma función para redondear que nexus.		
Parámetros:	nNumero	Nº a redondear
	nDecima	Nº de decimales del resultado
Valor retornado	Double	Número redondeado con los decimales indicados

Function SigId(sId:String):Variant;		
Retorna un identificador único para el nombre que se pasa por parámetro.		
Parámetros:	sId	Nombre del identificador
Valor retornado	Variant	Identificador único

Function DescArticulo (sCodArt, sCodIdi: String; var sDescArt, sTexto: String);		
Retorna en los dos últimos parámetros de la función la descripción y el texto del artículo, en el idioma indicado.		
Parámetros:	sCodArt	Código del artículo
	sCodIdi	Código del idioma
Valor retornado	sDescArt: String	Descripción del artículo
	sTexto: string	Texto del artículo

Function ObtRepresentante(sCodCli: String): String;		
Retorna el representante de un cliente.		
Parámetros:	sCodCli	Código del cliente
Valor retornado	String	Código del representante

Function CuentaArtV(sCodArt, sCodCli: String): String;		
Retorna la cuenta de ventas asociada a un artículo para un cliente determinado. El proceso que sigue el siguiente esquema: • Cuenta de ventas del artículo (si la hay) • Cuenta de ventas del cliente (si la hay) • Cuenta de ventas de los datos generales		
Parámetros:	sCodArt	Código del artículo
	sCodCli	Código del cliente
Valor retornado	String	Cuenta de ventas

Function CuentaArtC(sCodArt, sCodPro: String): String;		
Retorna la cuenta de compras asociada a un artículo para un proveedor determinado. El proceso que sigue el siguiente esquema: • Cuenta de compras del artículo (si la hay) • Cuenta de compras del proveedor (si la hay) • Cuenta de compras de los datos generales		
Parámetros:	sCodArt	Código del artículo
	sCodPro	Código del proveedor
Valor retornado	String	Cuenta de compras

Function TotalLinea(nUni: Integer; Prc, desc1, desc2, desc3, desc4: Double; sCodMon: String;Iva: Boolean; sTipIva, sRegIva: String): Double		
El cálculo del total de una línea depende de muchas variables y en ocasiones es conveniente simular exactamente el mismo cálculo que realiza nexus.		
Parámetros:	nUni	Nº unidades
	Prc	Precio unitario
	Desc1	Descuento 1
	Desc2	Descuento2
	Desc3	Descuento3
	Desc4	Descuento4
	sCodMon	Código de la moneda
	Iva	IVA incluido (True → IVA incluido)
	sTipIva	Tipo de IVA
	sRegIva	Régimen de IVA
Valor retornado	Double	Total de la línea

function Vencimientos(bEsCobro: WordBool; sCodCliPro: WideString; nTotMon, nTotDoc: Currency; sForPag, sDocPag, sCodMon, sFecha: WideString): OleVariant;		
Crea un array con la información el conjunto de vencimientos con las condiciones especificadas.		
Parámetros:	bEsCobro	Indica si se trata de un cliente o proveedor. bEsCobro = True → Cliente. bEsCobro = False → Proveedor.
	sCodCliPro	Código del cliente o proveedor.
	nTotMon	Total del documento en la moneda indicada.
	nTotDoc	Total del documento en moneda principal.
	sForPag	Forma de pago.

	sDocPag	Documento de pago.
	sCodMon	Código de la moneda.
	sFecha	Fecha de la factura.
Valor retornado:	Variant array	[0] → numero de vencimientos creados (integer) [1][0][0] → Fecha vencimiento1 (fecha) [1] [0][1] → Importe vto1 (Currency) [2][0][0] → Fecha vencimiento2 (fecha) [2] [0][1] → Importe vto2 (Currency) …. [n][0][0] → Fecha vencimiento N (fecha) [n] [0][1] → Importe vto N (Currency)

procedure LlamarVisual(const OpcionNexus: WideString; Parametros: OleVariant);		
Muestra un pantalla visual de nexus		
Parámetros:	OpcionNexus	Identificador de la pantalla visual
	Parámetros	[0] → sin parametros [1][0] → Nombre del parámetro 1 [2][0] → Valor del parámetro 1 [1][1] → Nombre del parámetro 2 [2][1] → Valor del parámetro 2 …… Ver ejemplos en Objeto Opcion (llamada a opciones visuales)

procedure PasarPotACteNuevo (const sCodCli: WideString; const sCodPos: WideString; bPasarVinculos: WordBool; bPasarObserv: WordBool)		
Convertir un cliente potencial a cliente nuevo		
Parámetros:	sCodCli	Código de cliente nuevo
	sCodPos	Código de cliente potencial
	PasarVinculos	Traspasar los vínculos del cliente potencial a cliente
	PasarObserv	Traspasar las observaciones del cliente potencial al cliente

procedure PasarPotACteExis(const sCodCli: WideString; const sCodPos: WideString; bPasarVinculos: WordBool; bPasarObserv: WordBool)		
Convertir un cliente potencial a cliente ya existente		
Parámetros:	sCodCli	Código de cliente ya existente
	sCodPos	Código de cliente potencial
	PasarVinculos	Traspasar los vínculos del cliente potencial a cliente
	PasarObserv	Traspasar las observaciones del cliente potencial al cliente

function EvaluarFiltro(const CodFiltro: WideString; const Alias: WideString): WideString;		
Devuelve sentencia SQL de un filtro indicado		
Parámetros:	CodFiltro	Código de filtro
	Alias	Alias del filtro
Valor retornado:	String	Sentencia SQL resultado de la ejecución de filtro

CrearCierresIVAIGIC		
Delphi: procedure CrearCierresIVAIGIC(const Tipo: WideString; FechaInicio, FechaFin: TDateTime; HacerCierreDocumentos, HacerCierreContable: WordBool);		
C#: void CrearCierresIVAIGIC(string tipoContable, DateTime fechaInicio, Datetime fechaFin, bool hacerCierreDocumentos, bool hacerCierreContable);		
Crea los cierres de IVA / IGIC pare el tipo contable y cada uno de los meses del periodo indicado (fecha inicio – fecha fin). **A tener en cuenta:** Para crear un cierre de un mes dado debe existir previamente el del mes anterior. Es decir, no se puede crear, para un tipo contable dado, el cierre del mes de febrero si antes no se ha realizado el de enero para el mismo año. El único mes que no requiere un cierre anterior es el de enero. **Nota:** La información del día en la fecha de inicio o fecha de fin es irrelevante. Basta que sea cualquier día dentro del mes (para que la fecha sea correcta). Ejemplos: a) Cierre de febrero de 2019 (con cierre de documentos y contable): CrearCierresIVAIGIC('1', Date(1,2,2019), Date(1,2,2019), true, true); b) Cierre del año 2019 (con cierre de documentos y contable) CrearCierresIVAIGIC('1', Date(1,1,2019), Date(1,12,2019), true, true); c) Cierre de meses entre varios años (Junio/2018 a Mayo/2019 inclusive): CrearCierresIVAIGIC('1', Date(1,6,2018), Date(1,5,2019), true, true);		
Parámetros:	**Tipo**	Tipo contable
	FechaInicio	Fecha desde la cual crear cierres de IVA / IGIC. Se toma el mes y el año.
	FechaFin	Fecha hasta la cual crear cierres de IVA / IGIC. Se toma el mes y el año.
	HacerCierreDocumentos	Indica si se debe hacer un cierre de documentos para cada uno de los meses indicados por el intervalo de fechas.
	HacerCierreContable	Indica si se debe realizar un cierre contable en cada uno de los meses indicados por el intervalo de fechas.

BorrarCierresIVAIGIC		
Delphi: procedure BorrarCierresIVAIGIC(const Tipo: WideString; FechaInicio, FechaFin: TDateTime);		
C#: void BorrarCierresIVAIGIC(string tipoContable, DateTime fechaInicio, Datetime fechaFin);		
Borra los cierres de IVA / IGIC existentes para el tipo contable indicado entre los meses del periodo indicados (fecha inicio – fecha fin). **A tener en cuenta:** No tienen porque existir cierres de IVA / IGIC para los meses / ejercicios indicados por el periodo delimitado por fecha inicio y fecha fin. De existir alguno, será eliminado. La información del día en la fecha de inicio o fecha de fin es irrelevante. Basta que sea cualquier día dentro del mes (para que la fecha sea correcta). Ejemplos: a) Borra cierre de febrero de 2019 (si existiera): BorrarCierresIVAIGIC('1', Date(1,2,2019), Date(1,2,2019)); b) Borra cierres existentes en el año 2019 BorrarCierresIVAIGIC('1', Date(1,2,2019), Date(1,12,2019)); c) Borra cierre de meses existentes entre varios años (Junio/2018 a Mayo/2019 inclusive): BorrarCierresIVAIGIC('1', Date(1,2,2019), Date(1,2,2019));		
Parámetros:	**Tipo**	Tipo contable
	FechaInicio	Fecha desde la cual crear cierres de IVA / IGIC. Se toma el mes y el año.
	FechaFin	Fecha hasta la cual crear cierres de IVA / IGIC. Se toma el mes y el año.

RecalcularStocks		
Delphi: procedure RecalcularStocks(const CodArt, CodAlm: WideString);		
C#: void RecalcularStocks (string codArt, string codAlm);		
Recalcula las unidades de stock, solamente, para el artículo, almacén o combinación de ambos. **A tener en cuenta:** Si no se indicase el artículo o el almacén (cadena vacía), se entenderá que se pretende realizar un recálculo para todos ellos. Ejemplos: a) Recálculo de unidades de stock para el artículo 1 en el almacen 2: RecalcularStocks(' 1', ' 2'); b) Recálculo de unidades de stock para el artículo 1 en todos los almacenes: RecalcularStocks(' 1', ''); c) Recálculo de unidades de stock de todos los artículos en todos los almacenes: RecalcularStocks ('', '');		
Parámetros:	**CodArt**	Código de artículo
	CodAlm	Código de almacen

RecalcularStocksYPreciosMedios		
Delphi: procedure RecalcularStocksYPreciosMedios(ActualizarCosteDocVenta: WordBool);		
C#: void RecalcularStocksYPreciosMedios (bool actualizarCosteDocVenta);		
Recalcula las unidades y precio medio del stock para todos los artículos y almacenes. Opcionalmente se puede indicar si se desea que se actualicen los costes en los documentos de venta existentes. **A tener en cuenta:** Actualizar costes en documentos de venta relentizará el proceso de cálculo. Ejemplo: RecalcularStocksYPreciosMedios**(**false)		
Parámetro:	ActualizarCosteDocVenta	Indica si se debe o no actualizar costes en los documentos de venta.

Stock1Dia				
Delphi: function Stock1Dia(const Params: IDatos): WideString;				
C#:				
Genera el fichero de datos necesario para el listado de stock a un día. **A tener en cuenta:** Los parámetros deben pasarse usando una estructura IDatos. Ejemplo: Stock1Dia**(**Datos)				
Parámetro:	Params	Estructura IDatos que debe contener los siguientes valores:		
		AlmacenIni	Rango de almacenes cuyo stock obtener	Código almacén
		AlmacenFin		Código almacén
		FiltroAlmacen	Filtro (SQL) adicional sobre almacenes	
		ArticuloIni	Rango de artículos cuyo stock obtener	Código artículo
		ArticuloFin		Código artículo
		FiltroArticulo	Filtro (SQL) adicional sobre artículos	
		Fecha	Fecha del stock a un día (defecto: Hoy)	
		HayTallas	Si debe mostrarse tallas (defecto: false)	
		HayLotes	Si debe mostrarse lotes (defecto: false)	

		HayNumSerie	Si debe mostrarse números de serie (defecto: false)
		HayFecCaduc	Si debe mostrarse fechas de caducidad (defecto: false)
		HayUbicacion	Si debe mostrarse ubicaciones. (defecto: false)
		ArticulosConStock	Filtrar artículos con stock: 0 (defecto): Todos 1: Diferente de cero 2: Menor o igual que **Unidades** 3: Mayor que **Unidades** 4: Menor que **Unidades**
		Unidades	Unidades (obligatorio si…) (Por defecto: 0)
		ValorarStockPor	Modo valoración del stock: **vsNoValorar** (No valorar)-Por defecto **vsManual** (Manual) **vsPrcMedio** (por precio medio) **vsPrcCoste** (por precio de coste) **vsEstandar** (precio estándar) **vsUltPrcCompra** (último precio de compra neto) **vsPrcPondNoCon** (precio ponderado no continuo)
		Cambio	Cambio a aplicar (por defecto, 1)

4.- OBJETO MAESTRO

Este objeto permite moverse por los ficheros, así como añadir nuevos y modificar los existentes. Los ficheros son los siguientes: Artículos, Proveedores, Clientes, Cuentas, Bancos, Almacenes, Representantes, Transportistas, Provincias, Países, Tarifas, Idiomas, Domiciliaciones bancarias, etc... .

Interface Maestro
Property AsString[const sCampo: WideString]: WideString; **property** AsFloat[const sCampo: WideString]: Double; **property** AsInteger[const sCampo: WideString]: Integer; **property** AsBoolean[const sCampo: WideString]: WordBool; **property** AsVariant[const sCampo: WideString]: Variant; **property** EOF: WordBool **readonly**; **property** BOF: WordBool **readonly**; **property** Estado: EstadoMaestro **readonly**;

property Filtro: WideString;
property Filtrado: WordBool;
property OmitirMensajes: WordBool;
function Buscar(V: OleVariant): WordBool;
function NuevoCodigoNum: WideString;
function Seleccionar: WideString;
function ExisteCampo(const cCampo: WideString): WordBool;
procedure Iniciar(const sNombre: WideString);
procedure Acabar;
procedure Nuevo;
procedure Guarda(bSobreescribir: WordBool);
procedure Primero;
procedure Anterior;
procedure Siguiente;
procedure Ultimo;
procedure Edita;
procedure Cancelar;
procedure Borrar;
procedure CambiarCodigo(const sViejo: WideString; const sNuevo: WideString);
function Seleccionar2: OleVariant;
procedure Duplicar(CodigoOrigen:olevariant; codigoDuplicado: Olevariant, const Params: WideString);

Maestros posibles

- **Maestro artículos.**
- **Maestro clientes, clientes potenciales, proveedores.**

Propiedades	Tipo	Descripción
AsString	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsFloat	Tabla(float)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsInteger	Tabla(integer)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsBoolean	Tabla(lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsVariant	Tabla(lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
BOF	Lógico	Indica que estamos al principio de los registros del maestro.
EOF	Lógico	Al final.
Estado	EstadoMaestro	Devuelve en que estado se encuentra el maestro.
Filtro	String	Cadena de texto que permite poner un filtro en el maestro.

Filtrado	Lógico	Valor lógico para activar/desactivar el filtro
OmitirMensajes	Lógico	Asignar valor para ocultar mensajes de información

Métodos	**Tipo**	**Descripción**
Iniciar	Procedimiento	Inicia el maestro especificado. Pasa al estado estM_ACTIVO
Acabar	Procedimiento	Cierra el maestro. Estado estM_NOACTIVO.
Primero	Procedimiento	Ir al primer registro.
Anterior	Procedimiento	Al anterior.
Siguiente	Procedimiento	Al siguiente.
Ultimo	Procedimiento	Al último.
Buscar	Función	Busca un registro por clave principal.
Nuevo	Procedimiento	Pasa estado estM_NUEVO en el que se pueden asignar valores a los campos del registro.
Edita	Procedimiento	Pasa estado estM_EDICION en el que se pueden asignar valores a los campos del registro.
Guarda	Procedimiento	Inserta o modifica el registro actual del maestro.
Cancelar	Procedimiento	Pasa al estado estM_ACTIVO, descartando el registro actual.
Borrar	Procedimiento	Borra el registro actual del maestro.
NuevoCodigoNum	Función	Retorna un string con el siguiente código numérico para un alta.
Seleccionar	Función	Presenta una ventana de selección y retorna el código del registro seleccionado.
ExisteCampo	Función	Indica si existe un nombre de campo dentro del maestro.
CambiarCodigo	Procedimiento	Cambia la clave con el valor indicado por otro también indicado.
Seleccionar2	Función	Presenta una ventana de selección y retorna los códigos que forman parte de la clave, del registro seleccionado.
Duplicar	Procedimiento	Permite duplicar un registro existente sobre uno nuevo.

Interface IMaestroA3ASESOR	
El objeto mestro implementa además de los metodos antes indicados, los propios del interface IMaestroA3ASESOR. --->**A tener en cuenta:** Estos métodos deben usarse únicamente en aplicaciones integradas con a3ASESOR Este interface implementa métodos siguientes:	
CodCliAsesor	**Solo aplicable a clientes.** Permite indicar el valor de A3Asesor con el cual un determinado cliente de ERP está relaccionado. Es de lectura y escritura.
HabilitarValidacionValoresObligatorios	Habilita la validación de los valores obligatorios que se realiza durante el proceso de guardado.
DeshabilitarValidacionValoresObligatorios	Deshabilita la validación de los valores obligatorios que se realiza durante el proceso de guardado.

Las primeras funciones de la lista que se presentan a continuación se utilizan para asignar / leer valores de los campos de un maestro.

Deben tenerse en cuenta algunas normas importantes:

- Si el nombre del campo no existe se producirá un error.
- Si el valor asignado al campo especificado no es correcto, no se producirá un error. Este se producirá al intentar guardar el registro.

Hay que tener en cuenta lo siguiente:

- Los campos fecha se han de asignar como cadenas de texto como '1/1/2002'
- Los campos clave son obligatorios.
- Uso en Delphi: a3ERPACTIVEXMaestro.AsString['CodCli']
- Uso en Visual Basic: a3ERPACTIVEXMaestro.AsString('CodCli')

property AsString[sCampo:String]: String		
A través de esta propiedad podemos asignar valor a los campos especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsInteger[sCampo:String]: Integer		
A través de esta propiedad podemos asignar valor a los campos especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloat[sCampo:String]: Double		
Con esta propiedad podemos asignar valor a los campos especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBoolean[sCampo:String]: Boolean		
Con esta propiedad podemos asignar valor a los campos especificando que su tipo es lógico.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsVariant[sCampo:String]: Boolean		
A través de esta propiedad podemos asignar valor a los campos especificando que su tipo es variant.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Variant	Valor del campo

property EOF:Boolean		
Esta propiedad nos indica si nos encontramos al final del maestro.		
Valor retornado	Lógico	Indica si estamos al final del maestro.

property BOF:Boolean		
Esta propiedad nos indica si nos encontramos al principio del maestro.		
Valor retornado	Lógico	Indica si estamos al principio del maestro.

property Estado:EstadoMaestro		
Esta propiedad nos indica el estado del objeto. Posibles estados: ESTM_NOACTIVO: Maestro cerrado. No se puede usar. ESTM_ACTIVO: Maestro activo. Podemos añadir, borrar, modificar y movernos por los registros. ESTM_NUEVO: Maestro en inserción de un nuevo registro. ESTM_EDICION: Maestro en estado modificación, podemos asignar valores a los campos. Modificando el registro actual al guardarlo. Si sólo cambiásemos el campo clave crearíamos una copia con una clave diferente.		
Valor retornado	EstadoMaestro	Retorna el estado del maestro.

property Filtro:String
En esta propiedad podemos indicar el filtro que se aplica al fichero maestro. La sintaxis del filtro debe ser en sintaxis de la base de datos. En la actualidad en sintaxis de SQL Server. Si más adelante se pueden atacar otras bases de datos esta función se utilizará de distinta forma dependiendo de la base de datos.

property Filtrado:Boolean
Indica si se utiliza el contenido de la propiedad Filtro o no.

property OmitirMensajes:Boolean
Para activar o desactivar mensajes de información.

procedure Iniciar(sNombre: string);		
Este procedimiento es el que reserva los recursos necesarios para que el maestro pueda ser usado. El parámetro de esta función identifica el maestro con el que se va a trabajar y por tanto los campos disponibles. Maestros válidos: Clientes, Articulo, Proveed, Cuentas, Bancos, DocuPago, Almacenes, Represen, Transpor, Provinci, Paises, Idiomas, Tarifas, TarifasVe, PrcEsp, Apuntes. Al iniciar el maestro el estado cambia de ESTM_NOACTIVO a ESTM_ACTIVO.		
Parámetros:	sNombre	Nombre del fichero maestro que se iniciará.

procedure Acabar;
Este procedimiento es el que libera los recursos utilizados por el objeto. Al acabar el maestro el estado cambia de ESTM_ACTIVO a ESTM_NOACTIVO.

procedure Primero;
Mueve el puntero al primer registro del maestro ordenado por la clave principal y única.

procedure Siguiente;
Mueve el puntero al siguiente registro del maestro ordenado por la clave principal y única.

procedure Anterior;
Mueve el puntero al anterior registro del maestro ordenado por la clave principal y única.

procedure Ultimo;
Mueve el puntero al último registro del maestro ordenado por la clave principal y única.

function Buscar(V: OleVariant): WordBool;		
Mueve el puntero al primer registro del maestro con la clave indicada en el parámetro V. Cuando la clave es compuesta se debe pasar un array con los distintos valores. Cuando la clave es de un único campo se puede pasar en un string.		
Parámetros:	sClave	Contenido de la clave que se está buscando.
Valor retornado	Lógico	Indica si se ha encontrado la clave.

procedure Nuevo;
Inserta un nuevo registro y los prepara para ser editado. Pasa de estado ESTM_ACTIVO a ESTM_NUEVO.

procedure Edita;
Pone el registro actual en edición. Pasa de estado ESTM_ACTIVO a ESTM_EDICION.

Procedure Guarda(bSobreescribir: Boolean);		
Almacena los valores de los campos en la base de dato con los cambios introducidos por el programador. Pasa de los estados ESTM_NUEVO o EST_EDICION a ESTM_ACTIVO.		
Parámetros:	bSobreescribir	True: Se sobrescriben los valores que existen en el registro. Si se modifica el código se cambia el código del maestro. False: Se crea un nuevo registro con al contenido de la ficha. Si ya existe ese código da error.

Procedure Cancela;
Cancela la edición del registro. Pasa de los estados ESTM_NUEVO o EST_EDICION a ESTM_ACTIVO.

Procedure Borrar;
Borra el registro activo y pasa al siguiente registro.

function NuevoCodigoNum: String;		
Retorna el último registro numérico + 1. Este es el sistema que utiliza nexus para proponer el código de clave que toca al crear un nuevo registro en un maestro.		
Valor retornado	String	Valor del código.

function Seleccionar: String;		
Permite usar la selección por defecto de los maestros de nexus. Retorna el valor de la clave seleccionada. Esta función está pensada exclusivamente para maestros con un único campo clave. Aquellos maestros con más de un campo clave no tienen selección. Esta función no se coloca en el registro seleccionado, tras realizar la selección, con el resultado debe buscarse el registro seleccionado a través de la función Buscar.		
Valor retornado	String	Valor del código de la clave

function Seleccionar2: OleVariant;		
Permite usar la selección por defecto de los maestros de nexus. Retorna lista de valores de la clave seleccionada (para ficheros con claves múltiples) Esta función está pensada para maestros con clave múltiple. Esta función no se coloca en el registro seleccionado, tras realizar la selección, con el resultado debe buscarse el registro seleccionado a través de la función Buscar.		
Valor retornado	OleVariant	Lista de valores de lo campos que forman parte de la clave

function ExisteCampo(sCampo:string): boolean;		
En ocasiones un campo puede aparecer en un maestro en una revisión determinada y el programador deberá preguntarse por su existencia antes de usarlo. Para ello existe esta función que te indica si existe un campo cuyo nombre se pasa por parámetro.		
Parámetros:	sCampo	Nombre de l campo
Valor retornado	Lógico	Indica si se ha encontrado el campo en el maestro.

procedure CambiarCodigo(sViejo: String; sNuevo: String);		
Modifica la clave de un maestro. Esta función exclusivamente funciona con claves con un único campo.		
Parámetros:	sViejo	Valor de la clave anterior
	sNuevo	Valor de la nueva clave

procedure Duplicar(CodigoOrigen: Olevariant; CodigoDuplicado: Olevariant; Params:String);		
Permite duplicar el registro del codigoorigen sobre un registro nuevo con codidoduplicado. (Sólo para el maestro de artículos y clientes)		
Parámetros:	CodigoOrigen	Valor de la clave origen
	CodigoDuplicado	Valor de la nueva clave
	Maestro	**Params**
	Artículos	(Stock, Vinculos, Referencias, Idiomas, Escandallo, TarifasV, PreciosEspV, AlarmasV, ComisionesV, DescuentosV, TarifasC, PreciosEspC, AlarmasC, DescuentosC) p.e Stock=True,Idiomas=False,DescuentosC=False
	Clientes (VER 13.0.00)	(Contactos, DomBanca, Alarmas, Dirent, Vinculos, Referencias)
	Clientes potenciales (VER 13.0.00)	(Contactos, Vinculos)
	Proveedores (VER 13.0.00)	(Contactos, DomBanca, Alarmas, Dirent, Vinculos, Referencias)

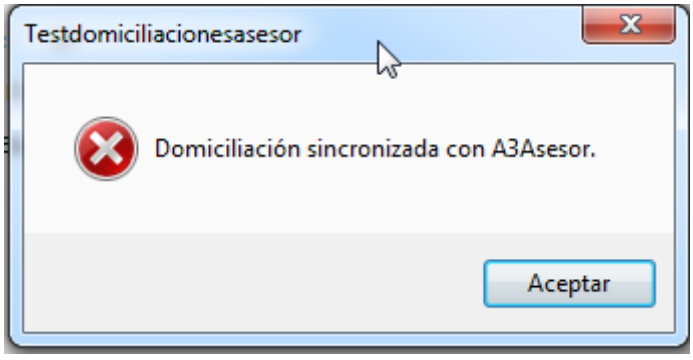
5.- OBJETO MAESTRO DOMBANCA

Cuando no estamos conectados con el a3Asesor el comportamiento es el general del Objeto Maestro.

Si estamos conectados con a3Asesor y usamos el Maestro “DomBancaV”, si el registro a editar proviene del a3Asesor, únicamente podremos modificar los campos de mandatos. El resto de campos relativos a la domiciliación bancaria són gestionados desde el a3asesor y no pueden ser modificados.

Véase: 'MANDATOREF', 'MANDATOCADUCADO', 'MANDATOCONFIRMADO', MANDATOENVIADO', 'MANDATOFECHACADUCADO', 'MANDATOCADUCADOMOTIVO', 'MANDATOFECHACONFIRMADO', 'MANDATOFECHAENVIO', 'MANDATOFECHAFIRMA', 'MANDATONOMBRE', 'MANDATOTIPO', 'MANDATOTIPOADEUDO', ‘MANDATOEMAILENVIADO', 'MANDATOFORMATOMODELO'.

Cuando hayamos modificado un campo que no esté en esa lista, al guardar se mostrará la excepción.



6.- OBJETO SELECCIÓN

Este objeto permite presentar la pantalla de selección estándar de nexus, para mostrar los registros de cualquier tabla con el formato código, descripción, etc…

Interface Selección
property Tabla: WideString; **property** CampoResultado: WideString; **property** Filtro: WideString; **function** Ejecutar: WideString; **function** Ejecutar2: OleVariant;

Propiedades	**Tipo**	**Descripción**
Tabla	String	En esta propiedad podemos indicar la tabla que queremos seleccionar.
CampoResultado	String	En esta propiedad podemos indicar el nombre del campo y la selección devolverá el valor de este campo.
Filtro	String	En esta propiedad podemos indicar el filtro que se aplica a la tabla seleccionada La sintaxis del filtro debe ser en sintaxis de la base de datos. En la actualidad en sintaxis de SQL Server. Si más adelante se pueden atacar otras bases de datos esta función se utilizará de distinta forma dependiendo de la base de datos.

Métodos	**Tipo**	**Descripción**
Ejecutar	Función	Presenta una ventana de selección y retorna el código del registro seleccionado.
Ejecutar2	Función	Presenta una ventana de selección y retorna los códigos que forman parte de la clave, del registro seleccionado.

Property Tabla:String
En esta propiedad podemos indicar la tabla que queremos seleccionar.

Property CampoResultado:String
En esta propiedad podemos indicar el nombre del campo y la selección devolverá el valor de este campo.

property Filtro:String
En esta propiedad podemos indicar el filtro que se aplica a la tabla seleccionada La sintaxis del filtro debe ser en sintaxis de la base de datos. En la actualidad en sintaxis de SQL Server. Si más adelante se pueden atacar otras bases de datos esta función se utilizará de distinta forma dependiendo de la base de datos.

function Ejecutar: String;		
Permite usar la selección estándar de los maestros de nexus. Retorna el valor de la clave seleccionada. Esta función está pensada ficheros con campos código, descripción, etc. con un único campo clave. Aquellos maestros con más de un campo clave utilizaremos la funcion Ejecutar2		
Valor retornado	String	Valor del código de la clave

function Ejecutar2: OleVariant;		
Permite usar la selección estándar de los maestros de nexus. Retorna lista de valores de la clave seleccionada (para ficheros con claves múltiples) Esta función está pensada para maestros con clave múltiple.		
Valor retornado	OleVariant	Lista de valores de lo campos que forman parte de la clave

7.- OBJETO FACTURA 1302

Objeto que permite la creación, modificación y borrado de facturas. (Definición hasta versión 13.02)

Interface Factura
property Estado: EstadoMaestro readonly; **property** AsStringCab[const sCampo: WideString]: WideString; **property** AsFloatCab[const sCampo: WideString]: Double; **property** AsIntegerCab[const sCampo: WideString]: Integer; **property** AsBooleanCab[const sCampo: WideString]: WordBool; **property** AsCurrencyCab[const sCampo: WideString]: Currency; **property** AsVariantCab[const sCampo: WideString]: Currency; **property** AsStringLin[const sCampo: WideString]: WideString; **property** AsFloatLin[const sCampo: WideString]: Double; **property** AsIntegerLin[const sCampo: WideString]: Integer; **property** AsBooleanLin[const sCampo: WideString]: WordBool; **property** AsCurrencyLin[const sCampo: WideString]: Currency; **property** AsVariantLin[const sCampo: WideString]: Currency; **property** AsBooleanComp[const sCampo: WideString]: WordBool; **property** AsStringComp[const sCampo: WideString]: WideString; **property** AsFloatComp[const sCampo: WideString]: Double; **property** AsIntegerComp[const sCampo: WideString]: Integer; **property** AsCurrencyComp[const sCampo: WideString]: Currency; **property** AsVariantComp[const sCampo: WideString]: Currency; **property** AsBooleanDes[const sCampo: WideString]: WordBool; [SOLO LECTURA] **property** AsStringDes[const sCampo: WideString]: WideString; [SOLO LECTURA] **property** AsFloatDes[const sCampo: WideString]: Double; [SOLO LECTURA]

```
property AsIntegerDes[const sCampo: WideString]: Integer; [SOLO LECTURA]
property AsCurrencyDes[const sCampo: WideString]: Currency; [SOLO LECTURA]
property AsVariantDes[const sCampo: WideString]: OleVariant; [SOLO LECTURA]
property OmitirMensajes:WordBool;
property ActivarAlarmaCab : WordBool;
property ActivarAlarmaLin: WordBool;
property ValidarPrecios: WordBool;
property ValidarArtBloqueado: WordBool;
property AvisarStock: WordBool;
property AvisarRiesgo: WordBool;
property AvisarCentroCosteCabecera: WordBool;
procedure Iniciar;
procedure Acabar;
procedure Cancela;
procedure Nuevo(  sFecha: WideString; sCodCliPro: WideString; bEsDeCompra: WordBool;
                  bEsContable: WordBool; bConReper: WordBool; bConCartera: WordBool);
procedure Borra(IdDocu: Currency; bEsDeCompra: WordBool);
procedure Modifica(IdDocu: Currency; bEsDeCompra: WordBool);
procedure ModificaParcial(IdDocu: Currency; NumLin: Integer, EsDeCompra: WordBool);
function Anade: Currency;
procedure NuevaLinea;
procedure AnadirLinea;
procedure CancelaLin;
procedure NuevaLineaArt(const sCodArt: WideString; nUnidades: Currency);
procedure EditarLinea(nIdLinea: Currency);
procedure BorrarLinea(nIdLinea: Currency);
procedure IniciarServir(const sIdDocu: WideString; nIdDocu: Currency; bConMensaje:
WordBool);
procedure ServirDocumento;
procedure FinServir;
procedure ServirLinea( nNumGru, nNumLin, nNumBul, nNumPaq: Currency;
                       nNumUni, nPrcMed: Double;
                       sNSerie: WideString; sLote: WideString; sUbicac: WideString;
                       sFecCad: WideString);
procedure AnularLinea( nNumLin: Currency; nNumBul: Currency; nNumPaq: Currency;
                        nNumUni: Double);
procedure AnadirDetalle( nUnidades: Double; sNumSerie: WideString; sLote: WideString;
                         sUbicacion: WideString; sFecCadu: WideString);
procedure CambiarDetalle ( nIdLine: Double; nUnidades: Double; sNumSerie: WideString;
                           sLote: WideString; sUbicacion: WideString;  sFecCadu: WideString);
procedure BorrarDetalle(nIdLine: Double);
procedure AnadirTalla(sCodFamTallaH, sCodFamTallaV, sCodTallaH,
    sCodTallaV: WideString; nUnidades, nPrcMoneda: Double);
 procedure CambiarTalla(nIdLine, nUnidades, nPrcMoneda: Double);
```

```
procedure BorrarTalla(nIdLine: Double);
procedure NuevoComponente;
procedure EditarComponente(nIdLine:Currency);
procedure BorrarComponente(nIdLine:Currency);
procedure AnadirComponente;
procedure CancelarComponente;
procedure ConsumirFacturaAnticipo(IdFactura:Currency; Importemoneda:currency;
Importe:currency);
procedure AnadirTallaColor(const sCodTallaH: WideString; const sCodTallaV: WideString;
nUnidades: Double);
procedure RectificarPeriodo(const FechaIni, FechaFin: WideString); safecall;
procedure AnadirRectificada(IdFac: Double);
procedure EliminarRectificada(IdFac: Double);
procedure ACuenta(bEsCompra: WordBool; nIdFact, NumCarteraAnti,
    nImpACuentaMon: Currency);
procedure CalcularImpuestosyTotales;
function ObtenerComponentesLinea: OleVariant;
function ObtenerDetalleComponentesLinea: OleVariant;
function ObtenerDetallesLinea: OleVariant;
```

Interface ILineasDocumento (*cast* Factura a ILineasDocumento)
function ObtenerLineas: OleVariant; **procedure AnadirSubdetalle(const ParametrosDetalle: ILineaDetalleParametros);** **procedure BorrarSubdetalle(IdLinea: Currency);** **procedure CambiarSubdetalle(const ParametrosDetalle: ILineaDetalleParametros);**

Propiedades	Tipo	Descripción
AsStringCab	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsIntegerCab	Tabla(Integer)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsFloatCab	Tabla(Float)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsBooleanCab	Tabla(Lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsCurrencyCab	Tabla(Currency)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsStringLin	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsIntegerLin	Tabla(Integer)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsFloatLin	Tabla(Float)	Asigna/lee valores a/de los campos del registro nuevo/actual.

AsCurrencyLin	Tabla(Currency)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsBooleanLin	Tabla(Lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
Estado	EstadoMaestro	Devuelve el estado en que se encuentra el objeto
AsStringComp	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsIntegerComp	Tabla(Integer)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsFloatComp	Tabla(Float)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsBooleanComp	Tabla(Lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsCurrencyComp	Tabla(Currency)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsStringDes	Tabla(string)	Lee valores de los campos del registro nuevo/actual.
AsIntegerDes	Tabla(Integer)	Lee valores de los campos del registro nuevo/actual.
AsFloatDes	Tabla(Float)	Lee valores de los campos del registro nuevo/actual.
AsBooleanDes	Tabla(Lógico)	Lee valores de los campos del registro nuevo/actual.
AsCurrencyDes	Tabla(Currency)	Lee valores de los campos del registro nuevo/actual.
AsVariantDes	Tabla(Variant)	Lee valores de los campos del registro nuevo/actual.
OmitirMensajes	Lógico	Asignar valor para ocultar mensajes de información
ActivarAlarmaCab	Lógico	Asignar valor para activar o no alarmas en documentos
ActivarAlarmaLin	Lógico	Asignar valor para activar o no alarmas en documentos
ValidarPrecios	Lógico	Asignar valor para activar o no los mensajes de validación de precios
ValidarArtBloqueado	Lógico	Asignar valor para activar o no los mensajes de los artículos bloqueados
AvisarStock	Lógico	Asignar valor para activar o no los mensajes de aviso de stock
AvisarRiesgo	Lógico	Asignar valor para activar o no los mensajes de aviso de riesgo
AvisarCentroCosteCabecera	Lógico	Asignar valor para activar o no el mensaje que aparece al cambiar algún nivel de analítica y pregunta si se desa cambiar en las líneas.
CalcularImpuestosyTotales	Lógico	Realiza el cálculo de los impuestos y los totales del documento.

*EstadoMaestro: Ver objeto Maestro.

Método	Tipo	Descripción
Iniciar	Procedimiento	Inicia el objeto
Acabar	Procedimiento	Cierra el objeto
Nuevo	Procedimiento	Inicia un nuevo documento
Borra	Procedimiento	Borra el documento
Modifica	Procedimiento	Prepara el documento para poderlo modificar.
ModificaParcial	Procedimiento	Prepara el documento para poder cambiar solamente el detalle de una línea
Cancela	Procedimiento	Cancela la edición del documento.
Anade	Función	Añade los datos editados del documento a la base de datos y devuelve el identificador con el que se almacenará.
NuevaLinea	Procedimiento	Inicia una nueva línea
NuevaLineaArt	Procedimiento	Inicia una nueva línea aplicando las políticas de nexus, dados el código del artículo y las unidades pasadas como parámetro.
AnadirLinea	Procedimiento	Añade la línea
CancelaLin	Procedimiento	Cancela la línea en edición
EditarLinea	Procedimiento	Permite editar una línea ya existente del documento.
BorrarLinea	Procedimiento	Permite borrar una línea del documento.
IniciarServir	Procedimiento	Inicia el proceso de servir líneas de otro documento origen.
ServirDocumento	Procedimiento	Sirve el documento origen entero.
ServirLinea	Procedimiento	Sirve una línea del documento origen.
AnularLinea	Procedimiento	Anula una línea del documento origen conforme no podrán servirse las unidades indicadas.
FinServir	Procedimiento	Finaliza el proceso de servir.
AnadirDetalle	Procedimiento	Identifica los nº de serie, lotes, fechas de caducidad y/o ubicaciones de una línea del documento.
CambiarDetalle	Procedimiento	Cambia los nº de serie, lotes, fechas de caducidad y/o ubicaciones de una línea del documento.
BorrarDetalle	Procedimiento	Borra el detalle los nº de serie, lotes, fechas de caducidad y/o ubicaciones de una línea del documento.
AnadirTalla	Procedimiento	Identifica las tallas de una línea del documento.
CambiarTalla	Procedimiento	Cambia las tallas de una línea del documento.
BorrarTalla	Procedimiento	Borra el detalle de las tallas de una línea del documento.
NuevoComponente	Procedimiento	Inicia una nueva línea componente de un kit.
EditarComponente	Procedimiento	Permite editar una línea componente de un kit.

BorrarComponente	Procedimiento	Borra una línea componente de un kit.
AnadirComponente	Procedimiento	Añade una línea componente de un kit.
CancelarComponente	Procedimiento	Cancela una línea componente de un kit.
ConsumirFacturaAnticipo	Procedimiento	Permite consumir una factura de anticipo
RectificarPeriodo	Procedimiento	Marca la factura como rectificativa de periodo, y permite introducir el periodo rectificado.
AnadirRectificada	Procedimiento	Marca la factura como rectificativa, si no lo es ya, y añade la factura a la que rectifica.
EliminarRectificada	Procedimiento	Elimina una factura rectificada en una factura rectificativa.
ACuenta	Procedimiento	Permite usar un anticipo para cobrar una factura de venta, o para pagar una factura de compra.
AnadirSubdetalle	Procedimiento	Permite añadir líneas de detalle (lotes, num serie…) de la línea del componente de un kit
CambiarSubdetalle	Procedimiento	Permite cambiar la línea de detalle (lotes, num serie…) de la línea del componente de un kit
BorrarSubdetalle	Procedimiento	Premite borrar la línea de detalle (lotes, num serie..) de la línea del componente de un kit

Funciones que se obtienen al convertir (*cast*) a ILineasDocumento		
ObtenerComponentesLinea	Función	Permite obtener todas las líneas de componentes del artículo kit posicionado
ObtenerDetalleComponentesLinea	Función	Permite obtener todas las líneas de detalle del componente del kit posicionado
ObtenerDetallesLinea	Función	Permite obtener todas las líneas de detalle del artículo padre posicionado
ObtenerLineas	Función	Permite obtener todas las líneas sin detalle/componentes del documento

Las primeras funciones de la lista que se presentan a continuación se utilizan para asignar / leer valores de la cabecera o de las líneas del documento.

Deben tenerse en cuenta algunas normas importantes:

- Si el nombre del campo no existe se producirá un error.
- Si el valor asignado al campo especificado no es correcto, no se producirá un error. Este se producirá al intentar guardar el documento o la línea.

Hay que tener en cuenta lo siguiente:

- Los campos fecha se han de asignar como cadenas de texto como ‘01/01/2003’
- Uso en Delphi: a3ERPACTIVEXDocumento.AsStringCab[‘CodCli’]
- Uso en Visual Basic: a3ERPACTIVEXDocumento. AsStringCab (‘CodCli’)

property AsStringCab[sCampo:String]: String		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerCab[sCampo:String]: Integer		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatCab[sCampo:String]: Double		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanCab[sCampo:String]: Boolean		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es lógico.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsCurrencyCab[sCampo:String]: Currency		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es currency.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsVariantCab[sCampo:String]: OleVariant		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es variant.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Variant	Valor del campo

property AsStringLin[sCampo:String]: String		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerLin[sCampo:String]: Integer		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatLin[sCampo:String]: Double		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanLin[sCampo:String]: Boolean		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es lógico.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsCurrencyLin[sCampo:String]: Currency		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es currency.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsVariantLin[sCampo:String]: OleVariant		
A través de esta propiedad podemos asignar valor a los campos de la linea especificando que su tipo es variant.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Variant	Valor del campo

property AsStringComp[sCampo:String]: String		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerComp[sCampo:String]: Integer		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatComp[sCampo:String]: Double		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanComp[sCampo:String]: Boolean		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es lógico.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsCurrencyComp[sCampo:String]: Currency		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es currency.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsVariantComp[sCampo:String]: OleVariant		
A través de esta propiedad podemos asignar valor a los campos de la linea especificando que su tipo es variant.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Variant	Valor del campo

property AsStringDes[sCampo:String]: String [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerDes[sCampo:String]: Integer [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatDes[sCampo:String]: Double [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanDes[sCampo:String]: Boolean [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es lógico.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsCurrencyDes[sCampo:String]: Currency [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es currency.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsVariantDes[sCampo:String]: OleVariant [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es variant.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Variant	Valor del campo

property ValidarPrecios: WordBool		
A través de esta propiedad podemos asignar valor si queremos mostrar o ocultar los avisos de precios por debajo del precio mínimo		
Valor retornado	Lógico	Valor del campo

property ValidarArtBloqueado: WordBool		
A través de esta propiedad podemos asignar valor si queremos mostrar o ocultar los avisos de artículos bloqueados.		
Valor retornado	Lógico	Valor del campo

property AvisarStock: WordBool		
A través de esta propiedad podemos asignar valor si queremos mostrar o ocultar los avisos del stock		
Valor retornado	Lógico	Valor del campo

property AvisarRiesgo: WordBool		
A través de esta propiedad podemos asignar valor si queremos mostrar o ocultar los avisos de riesgo		
Valor retornado	Lógico	Valor del campo

property AvisarCentroCosteCabecera: WordBool		
A través de esta propiedad podemos asignar valor si queremos mostrar o ocultar el aviso para cambiar el centro de coste en las líneas cuando se cambia desde la cabecera		
Valor retornado	Lógico	Valor del campo

A continuación, se presentan las **propiedades y métodos necesarios para crear, borrar y modificar facturas.**

property Estado:EstadoMaestro		
Esta propiedad nos indica el estado del objeto. Posibles estados: ESTM_NOACTIVO: Documento cerrado. No se puede usar. ESTM_ACTIVO: Documento activo. Podemos añadir, borrar, modificar. ESTM_NUEVO: Documento en estado de inserción de un nuevo registro. ESTM_EDICION: Documento en estado modificación, podemos asignar valores a los campos.		
Valor retornado	EstadoMaestro	Retorna el estado del documento.

procedure Iniciar;
Este procedimiento es el que reserva los recursos necesarios para que el documento pueda ser usado. Al iniciar el documento el estado cambia de ESTM_NOACTIVO a ESTM_ACTIVO.

procedure Acabar;
Este procedimiento es el que libera los recursos utilizados por el objeto. Al acabar el documento el estado cambia de ESTM_ACTIVO a ESTM_NOACTIVO.

procedure Nuevo(sFecha: String; sCodCliPro: String; EsDeCompra: Boolean; EsContable: Boolean; ConReper: Boolean; ConCartera: Boolean);		
Inserta un nuevo registro y los prepara para ser editado. Pasa de estado ESTM_ACTIVO a ESTM_NUEVO.		
Parámetros:	sFecha	Fecha del documento
	sCodCliPro	Código del cliente o proveedor
	bEsDeCompra	True → Compra False → Venta
	bEsContable	True → Contable False → De gestión
	bConReper	Indica si se desean generar repercusiones contables
	bConCartera	Indica si se desean generar los vencimientos en función de la forma de pago.

procedure Modifica(IdDocu: Currency; EsDeCompra: WordBool);		
Pone el documento que tiene el identificador indicado en edición.		
Parámetros:	nIdDocu	Identificador del documento
	bEsDeCompra	True → Compra False → Venta

procedure ModificaParcial(IdDocu: Currency; NumLin: Integer; EsDeCompra: WordBool);		
Este método se utiliza únicamente para cambiar el detalle de una línea de un documento. Se puede utilizar en documentos con muchas líneas, mejorando la velocidad, ya que solo carga el detalle de la línea que se indica. Despues de llamar a este método solo se pueden utilizar los métodos AnadirDetalle, CambiarDetalle, BorrarDetalle, AnadirLinea, CancelaLin, Anade y Cancela. Pone el documento que tiene el identificador indicado en edición y la línea que tiene el Numlin indicado en edición		
Parámetros:	IdDocu	Identificador del documento
	NumLin	Numero de línea
	EsDeCompra	True → Compra False → Venta

Procedure Anade;
Almacena los valores de los campos en la base de dato con los cambios introducidos por el programador. Genera las repercusiones del documento como pueden ser: actualización de stocks, repercusiones contables, generación de vencimientos, actualización de estadísticas y almacenamiento de los datos de IVA. Pasa de los estados ESTM_NUEVO o EST_EDICION a ESTM_ACTIVO.

Procedure Cancela;
Cancela la edición del documento. Pasa de los estados ESTM_NUEVO o EST_EDICION a ESTM_ACTIVO.

procedure Borra(IdDocu: Currency; EsDeCompra: Boolean);		
Elimina el documento con el identificador indicado eliminando también sus repercusiones en stocks, contabilidad, cartera, estadísticas e IVA.		
Parámetros:	nIdDocu	Identificador del documento
	bEsDeCompra	True → Compra False → Venta

Procedure NuevaLinea;
Crea una línea con los valores por defecto. A partir de ese momento podremos editar los valores de la línea.

Procedure NuevaLineaArt(sCodArt: string; nUnidades: Currency);		
Crea una línea con el artículo y las unidades indicadas realizando las búsquedas necesarias de los valores correspondientes a cuentas, precios y descuentos, etc... A partir de ese momento podremos editar los valores de la línea.		
Parámetros:	sCodArt	Código del artículo
	nUnidades	Unidades

Procedure AnadirLinea;
Almacena los valores de los campos de la línea en la base de datos.

Procedure CancelaLin;
Cancela la edición de la línea activa.

Procedure EditarLinea(nIdLinea: Currency);		
Pone la línea indicada en el parámetro nIdLinea en edición, preparada para editar el contenido de sus campos. Al final de las modificaciones deberá llamarse a AnadirLinea.		
Parámetros:	nIdLinea	Identificador de la línea. Se corresponde con el campo NUMLINFAC del fichero LINEFACT. Para localizarlo debe realizarse una consulta de este campo (directamente a la base de datos) en los registros con IdFacV = a3ERPACTIVEXFactura.AsStringCab['IdFacV'] en ventas o IdFacC = a3ERPACTIVEXFactura.AsStringCab['IdFacC'] en compras.

procedure BorrarLinea(nIdLinea: Currency);		
Borra la línea con el identificador indicado en el parámetro nIdLinea.		
Parámetros:	nIdLinea	Identificador de la línea. Se corresponde con el campo NUMLINFAC del fichero LINEFACT. Para localizarlo debe realizarse una consulta de este campo (directamente a la base de datos) en los registros con IdFacV = a3ERPACTIVEXFactura.AsStringCab['IdFacV'] en ventas o IdFacC = a3ERPACTIVEXFactura.AsStringCab['IdFacC'] en compras.

En la siguiente colección de procedimientos se encapsula el funcionamiento de las utilidades para **servir cualquier documento sobre las facturas**. En primer lugar, debe llamarse al procedimiento IniciarServir. A continuación, pueden utilizarse los procedimientos ServirDocumento, ServirLinea o AnularLinea y finalmente debe llamarse a FinServir.

procedure IniciarServir(sIdDocu: string; nIdDocu: Currency; bConMensaje: Boolean);		
Prepara el documento indicado para servirse. Reserva los recursos necesarios.		
Parámetros:	sIdDocu	'O' → Oferta 'P' → Pedido 'A' → Albarán. 'D' → Depósito
	nIdDocu	En función del valor de sIdDocu (O,P,A,D) representa el identificador de la oferta (IdOfeV o IdOfeC), del pedido (IdPEdV o IdPedC), del albarán (IdAlbV o IdAlbC) o del depósito (IdDepV o IdDepC) respectivamente.

	bConMensaje	En caso en que se produzcan diferencias entre las condiciones del documento origen y el destino (Por ejemplo, la forma de pago, el documento de pago, el transportista, etc....), el programa puede actuar de dos formas distintas: - bConMesnaje = True: Preguntar al usuario si se desean aplicar las condiciones del documento destino o respetar las del documento origen. - bConMensaje = False: Aplicar las condiciones del documento destino.

procedure ServirDocumento;
Sirve todo el material pendiente de todo del documento origen indicado en la función IniciarServir.

procedure ServirLinea(nNumGrupo, nNumLin, nNumBul, nNumPaq: Currency; nNumUni, nPrcMedio: Double; sNSerie: WideString; sLote: WideString; sUbicac: WideString; sFecCad: WideString);		
Sirve la línea con los parámetros indicados.		
Parámetros:	nNumGrupo	Campo obsoleto. PONER UN 0.
	nNumLin	Nº de línea. NUMLINOFE en ofertas, NUMLINPED en pedidos, NUMLINALB en albaranes y NUMLINDEP en depósitos.
	nNumBul	Nº de bultos de la línea que desean servirse.
	nNumPaq	Nº de paquetes que desean servirse.
	nNumUni	Nº de unidades que desean servirse.
	nPrcMedio	Campo obsoleto. PONER UN 0.
	sNSerie	Nº de serie.
	sLote	Lote.
	sUbicac	Ubicación
	sFecCad	Fecha de caducidad.

procedure AnularLinea(nNumLin: Currency; nNumBul: Currency; nNumPaq: Currency; nNumUni: Double);		
Anula las unidades indicadas de la línea pasada por parámetro. Esas unidades no estarán disponibles para ser servidas posteriormente.		
Parámetros:	nNumLin	Nº de línea del documento origen (NUMLINOFE, NUMLINPED, NUMLINALB o NUMLINDEP).
	nNumBul	Nº de bultos de la línea que desean servirse.
	nNumPaq	Nº de paquetes que desean servirse.
	nNumUni	Nº de unidades que desean servirse.

procedure FinServir;
Realiza las operaciones realizadas desde que se llamó a IniciarServir de forma definitiva. Libera los recursos necesarios.

Los procedimientos que se describen a continuación se utilizan **para especificar los lotes, nº de serie, fechas de caducidad y ubicaciones de una línea** a no ser que esta selección se realice de forma automática. Para consultar si ya se han detallado todos los lotes, nº de serie, fechas de caducidad y ubicaciones necesarias se debe llamar a la función ConsultarDetalle;

procedure AnadirDetalle(nUnidades: Double; sNumSerie: WideString; sLote: WideString; sUbicacion: WideString; sFecCadu: WideString);		
Indica las unidades necesarias y su nº serie, lote, fecha de caducidad y ubicación de la línea activa. Para conocer los nº serie, lote, fecha de caducidad y ubicaciones disponibles en stock hay que llamar a la función **Varios.DetalleStockDisponible**..		
Parámetros:	nNunidades	Nº de unidades de la línea de detalle.
	sNumSerie	Nº de serie.
	sLote	Lote.
	sUbicacion	Ubicación.
	sFecCadu	Fecha de caducidad.

procedure CambiarDetalle(nIdLine: Double; sNumSerie: WideString; sLote: WideString; sUbicacion: WideString; sFecCadu: WideString);		
Indica las unidades necesarias y su Nº serie, lote, fecha de caducidad y ubicación de la línea activa. Para conocer los nº serie, lote, fecha de caducidad y ubicaciones disponibles en stock hay que llamar a la función **Varios.DetalleStockDisponible**. Para consultar el detalle de la línea activa hay que llamar a la función **ConsultarDetalle**.		
Parámetros:	nIdLine	Identificador de la línea. Se corresponde con el campo IDLIN de la tabla LINEFACT.
	nUnidades	Nº de unidades de la línea de detalle.
	sNumSerie	Nº de serie.
	sLote	Lote.
	sUbicacion	Ubicación.
	sFecCadu	Fecha de caducidad.

procedure BorrarDetalle(nIdLine: Double);		
Elimina la línea de detalle indicada en el parámetro.		
Parámetros:	nIdLine	Identificador de la línea. Se corresponde con el campo IDLIN de la tabla LINEFACT.

procedure AnadirTalla(sCodFamTallaH, sCodFamTallaV, sCodTallaH, sCodTallaV: WideString; nUnidades, nPrcMoneda:Double);		
Indica las unidades y el precio necesarios, junto con sus tallas y colores de la línea activa.		
Parámetros:	sCodFamTallaH	Familia de talla horizontal
	sCodFamTallaV	Familia de talla vertical
	sCodTallaH	Código de talla horizontal
	sCodTallaV	Código de talla vertical
	nUnidades	Nº unidades de la línea de talla
	nPrcMoneda	Precio de la línea de talla.

procedure CambiarTalla(nIdLine, nUnidades, nPrcMoneda: Double);		
Indica las unidades y el precio de la línea activa.		
Parámetros:	nIdLine	Identificador de la línea. Se corresponde con el campo IDLIN de la tabla LINEFACT.
	nUnidades	Nº de unidades de la línea de talla.
	nPrcMoneda	Precio de la línea de talla.

procedure BorrarTalla(nIdLine: Double);		
Elimina la línea de talla indicada en el parámetro.		
Parámetros:	nIdLine	Identificador de la línea. Se corresponde con el campo IDLIN de la tabla LINEFACT.

Procedure NuevoComponente;
Crea una línea con los valores por defecto. A partir de ese momento podremos editar los valores de la línea componente

Procedure AnadirComponente;
Almacena los valores de los campos de la línea componente en la base de datos.

Procedure CancelaComponente;
Cancela la edición de la línea componente activa.

procedure EditarComponente(nIdLinea: Currency);		
Pone la línea componente indicada en el parámetro nIdLinea en edición, preparada para editar el contenido de sus campos. Al final de las modificaciones deberá llamarse a AnadirComponente.		
Parámetros:	nIdLinea	Identificador de la línea componente. Se corresponde con el campo NUMLINFAC del fichero LINEFACT.

procedure BorrarComponente(nIdLinea: Currency);		
Borra la línea componente con el identificador indicado en el parámetro nIdLinea.		
Parámetros:	nIdLinea	Identificador de la línea componente. Se corresponde con el campo NUMLINFAC del fichero LINEFACT.

procedure ConsumirFacturaAnticipo(IdFactura: Currency; Importemoneda:currency; Importe:currency);		
Permite consumir una factura de anticipo		
Parámetros:	IdFactura	Identificador de la factura de anticipo que queremos consumir.
	ImporteMoneda	Importe a consumir en la moneda de la factura
	Importe	Importe a consumir en EUROS de la factura

procedure AnadirTallaColor(sCodTallaH, sCodTallaV: WideString; nUnidades:Double);		
Indica las unidades, junto con su talla y color de la línea activa.		
Parámetros:	sCodTallaH	Código de talla horizontal
	sCodTallaV	Código de talla vertical
	nUnidades	Nº unidades de la línea de talla

procedure RectificarPeriodo(const FechaIni, FechaFin: WideString);		
Convierte la factura en una rectificativa de periodo		
Parámetros:	FechaIni	Inicio del periodo a rectificar
	FechaFin	Fin del periodo a rectificar

procedure AnadirRectificada(IdFac: Double);		
Marca la factura como rectificativa, si no lo es ya. Añade una factura a la lista de facturas rectifacadas, en la factura que estamos editando.		
Parámetros:	IdFac	Identificador de la factura rectificada.

procedure EliminarRectificada(IdFac: Double);		
Elimina una factura a la lista de facturas rectifacadas, en la factura que estamos editando. Si la rectificada no existe da error.		
Parámetros:	IdFac	Identificador de la factura rectificada.

procedure ACuenta(bEsCobro: WordBool; nIdFact: Currency; nNumAnti: Currency; nImpACuentaMon: Currency);		
Procedimiento para cancelar un anticipo contra un vencimiento de una factura. Se puede consumir el anticipo parcialmente. Si el anticipo ya se ha usado en esa misma factura, el método se puede utilizar para modificar el importe consumido. Sustituye al método del mismo nombre en OperacionesCartera.		
Parámetros:	bEsCompra	Es una factura de compra (True) o de venta (False).
	nIdFact	Identificador de la factura. Se identifica con el campo IDFACV o IDFACC de las tablas CABEFACV o CABEFACC respectivamente en función de si se trata de una venta o una compra.
	nNumCarteraAnti	Identificador del anticipo. Se corresponde con el campo NUMCARTERA de la tabla CARTERA.
	nImpACuentaMon	Importe a cuenta en la moneda indicada.

Procedure CalcularImpuestosyTotales;
Realiza el cálculo de los impuestos y los totales del documento.

procedure AnadirSubdetalle(const ParametrosDetalle: ILineaDetalleParametros);		
Permite añadir líneas de detalle (lotes, num serie…) de la línea del componente de un kit		
Parámetros:	ParametrosDetalle	Clase que implementa la interfaz ILineaDetalleParametros. La que nos permite informar del detalle a añadir

procedure BorrarSubdetalle(IdLinea: Currency);		
Premite borrar la línea de detalle (lotes, num serie...) de la línea del componente de un kit		
Parámetros:	IdLinea	Identificador de la línea (campo IDLIN) para localizar el detalle y borrarlo

procedure CambiarSubdetalle(const ParametrosDetalle: ILineaDetalleParametros);		
Permite cambiar la línea de detalle (lotes, num serie…) de la línea del componente de un kit		
Parámetros:	ParametrosDetalle	Clase que implementa la interfaz ILineaDetalleParametros. La que nos permite informar del detalle a modificar. Si el valor de la línea es -1 cambiará el registro posicionado sin tener necesidad de buscarlo. Pensado para cuando sólo hay un detalle

function ILineasDocumento.ObtenerComponentesLinea: OleVariant
Permite obtener todas las líneas de componentes del artículo kit posicionado Devuelve un array de arrays con la siguiente estructura: A[0] => Numero líneas de componentes A[1..N][0] => Número de campos. Ejemplo 150 campos A[1..n][1..n][0] => Nombre del campo. Ejemplo 'CodArt' A[1..n][1..n][1] => Valor del campo. Ejemplo ' 1'

function ILineasDocumento.ObtenerDetalleComponentesLinea: OleVariant;
FunciónPermite obtener todas las líneas de detalle del componente del kit posicionado Devuelve un array de arrays con la siguiente estructura: A[0] => Numero líneas de detalle A[1..N][0] => Número de campos. Ejemplo 150 campos A[1..n][1..n][0] => Nombre del campo. Ejemplo 'CodArt' A[1..n][1..n][1] => Valor del campo. Ejemplo 'LOTE100'

function ILineasDocumento.ObtenerDetallesLinea: OleVariant;
Permite obtener todas las líneas de detalle del artículo padre posicionado Devuelve un array de arrays con la siguiente estructura: A[0] => Numero líneas de detalle A[1..N][0] => Número de campos. Ejemplo 150 campos A[1..n][1..n][0] => Nombre del campo. Ejemplo 'CodArt' A[1..n][1..n][1] => Valor del campo. Ejemplo 'LOTE100'

function ILineasDocumento.ObtenerLineas: OleVariant;
Permite obtener todas las líneas sin detalle/componentes del documento Devuelve un array de arrays con la siguiente estructura: A[0] => Numero líneas del documento A[1..N][0] => Número de campos. Ejemplo 150 campos A[1..n][1..n][0] => Nombre del campo. Ejemplo 'CodArt' A[1..n][1..n][1] => Valor del campo. Ejemplo 'B10

8.- OBJETO FACTURA140501

Extiende Factura1302 y añade nuevos métodos.

Interface Factura [Nuevo 14.0.0.0]
function AnadeComoBorrador: Currency;

function AnadeComoBorrador: Currency;		
Mediante este procedimiento se termina la edición de una factura de venta y la guarda como borrador. Almacena los valores de los campos en la base de dato con los cambios introducidos por el programador. Genera las repercusiones del documento como pueden ser: actualización de stocks, actualización de estadísticas y almacenamiento de los datos de IVA, excepto repercusiones contables y generación de vencimientos. La serie es modificada por la serie borrador.		
Valor retornado:	Currency	Añade los datos editados del documento a la base de datos y devuelve el identificador con el que se almacenará.

Procedure Anade;
Almacena los valores de los campos en la base de dato con los cambios introducidos por el programador. Genera las repercusiones del documento como pueden ser: actualización de stocks, repercusiones contables, generación de vencimientos, actualización de estadísticas y almacenamiento de los datos de IVA. Pasa de los estados ESTM_NUEVO o EST_EDICION a ESTM_ACTIVO.

9.- OBJETO FACTURA

Extiende Factura140501 y añade nuevos métodos.

Interface Factura
procedure Clonar(const DocumentoOrigenTipo: WideString; DocumentoOrigenID: Currency; BorrarLineasDelDocumentoDestino: WordBool; ClonarVariablesDeDocumentos: WordBool); [Nuevo 14.05.02]
procedure Copiar(const DocumentoOrigenTipo: WideString; DocumentoOrigenID: Currency; BorrarLineasDelDocumentoDestino: WordBool; CopiarCondicionesFinancieras: WordBool; CopiarLosPortes: WordBool; CopiarPreciosYDescuentos: WordBool; CopiarDescripcionesYObservacionesLineas: WordBool); [Nuevo 14.05.02]

procedure Clonar(const DocumentoOrigenTipo: WideString; DocumentoOrigenID: Currency; BorrarLineasDelDocumentoDestino: WordBool; ClonarVariablesDeDocumentos: WordBool);		
Mediante este procedimiento se permite clonar un documento sobre otro.		
Parámetros:		
DocumentoOrigenTipo	'OV' Oferta venta 'PV' Pedido de venta 'AV' Albarán de venta 'DV' Depósito de venta 'FV' Factura de venta	'OC' Oferta compra 'PC' Pedido de compra 'AC' Albarán de compra 'DC' Depósito de compra 'FC' Factura de compra
DocumentoOrigenID	Identificador del documento de origen	
BorrarLineasDelDocumentoDestino	Antes de realizar la operación, borra las lineas del documento destino	
ClonarVariablesDeDocumentos	Se clonarán las variables de los documentos	

procedure Copiar(const DocumentoOrigenTipo: WideString; DocumentoOrigenID: Currency; BorrarLineasDelDocumentoDestino: WordBool; CopiarCondicionesFinancieras: WordBool; CopiarLosPortes: WordBool; CopiarPreciosYDescuentos: WordBool; CopiarDescripcionesYObservacionesLineas: WordBool);		
Mediante este procedimiento se permite copiar un documento sobre otro.		
Parámetros:		
DocumentoOrigenTipo	'OV' Oferta venta 'PV' Pedido de venta 'AV' Albarán de venta 'DV' Depósito de venta 'FV' Factura de venta	'OC' Oferta compra 'PC' Pedido de compra 'AC' Albarán de compra 'DC' Depósito de compra 'FC' Factura de compra
DocumentoOrigenID	Identificador del documento de origen	
BorrarLineasDelDocumentoDestino	Antes de realizar la operación, borra las lineas del documento destino	
CopiarCondicionesFinancieras	Copiar condiciones financieras del documento origen. Se copiarán del documento origen la forma de pago, documento de pago, descuento de pronto pago o recargo financiero y si éste afecta a la base imponible.	
CopiarLosPortes	Copiar los portes al documento destino	

CopiarPreciosYDescuentos	Copiar precios y descuentos del documento origen
CopiarDescripcionesYObservacionesLineas	Copiar descripciones y observaciones de las líneas del documento origen

10.- OBJETO ALBARAN140501

Objeto que permite la creación, modificación y borrado de albaranes.

Interface Albaran
property Estado: EstadoMaestro readonly; **property** AsStringCab[const sCampo: WideString]: WideString; **property** AsFloatCab[const sCampo: WideString]: Double; **property** AsIntegerCab[const sCampo: WideString]: Integer; **property** AsBooleanCab[const sCampo: WideString]: WordBool; **property** AsCurrencyCab[const sCampo: WideString]: Currency; **property** AsVariantCab[const sCampo: WideString]: Currency; **property** AsStringLin[const sCampo: WideString]: WideString; **property** AsFloatLin[const sCampo: WideString]: Double; **property** AsIntegerLin[const sCampo: WideString]: Integer; **property** AsBooleanLin[const sCampo: WideString]: WordBool; **property** AsCurrencyLin[const sCampo: WideString]: Currency; **property** AsVariantLin[const sCampo: WideString]: Currency; **property** AsBooleanComp[const sCampo: WideString]: WordBool; **property** AsStringComp[const sCampo: WideString]: WideString; **property** AsFloatComp[const sCampo: WideString]: Double; **property** AsIntegerComp[const sCampo: WideString]: Integer; **property** AsCurrencyComp[const sCampo: WideString]: Currency; **property** AsVariantComp[const sCampo: WideString]: Currency; **property** AsBooleanDes[const sCampo: WideString]: WordBool; [SOLO LECTURA] **property** AsStringDes[const sCampo: WideString]: WideString; [SOLO LECTURA] **property** AsFloatDes[const sCampo: WideString]: Double; [SOLO LECTURA] **property** AsIntegerDes[const sCampo: WideString]: Integer; [SOLO LECTURA] **property** AsCurrencyDes[const sCampo: WideString]: Currency; [SOLO LECTURA] **property** AsVariantDes[const sCampo: WideString]: OleVariant; [SOLO LECTURA] **property** OmitirMensajes:WordBool; **property** ActivarAlarmaCab : WordBool; **property** ActivarAlarmaLin: WordBool; **property** ValidarPrecios: WordBool; **property** ValidarArtBloqueado: WordBool; **property** AvisarStock: WordBool; **property** AvisarRiesgo: WordBool;

```
property AvisarCentroCosteCabecera: WordBool;
procedure Iniciar;
procedure Acabar;
procedure Cancela;
procedure Nuevo(  sFecha: WideString; sCodCliPro: WideString; bEsDeCompra: WordBool);
procedure Borra(IdDocu: Currency; bEsDeCompra: WordBool);
procedure Modifica(IdDocu: Currency; bEsDeCompra: WordBool);
procedure ModificaParcial(IdDocu: Currency; NumLin: Integer, EsDeCompra: WordBool);
function Anade: Currency;
procedure NuevaLinea;
procedure AnadirLinea;
procedure CancelaLin;
procedure NuevaLineaArt(const sCodArt: WideString; nUnidades: Currency);
procedure EditarLinea(nIdLinea: Currency);
procedure BorrarLinea(nIdLinea: Currency);
procedure IniciarServir(const sIdDocu: WideString; nIdDocu: Currency; bConMensaje:
WordBool);
procedure ServirDocumento;
procedure FinServir;
procedure ServirLinea( nNumGru: Currency; nNumLin: Currency; nNumBul: Currency;
                       nNumPaq: Currency; nNumUni: Double; nPrcMed: Double;
                       sNSerie: WideString; sLote: WideString; sUbicac: WideString;
                       sFecCad: WideString);
procedure AnularLinea( nNumLin: Currency; nNumBul: Currency; nNumPaq: Currency;
                        nNumUni: Double);
procedure AnadirDetalle( nUnidades: Double; sNumSerie: WideString; sLote: WideString;
                         sUbicacion: WideString; sFecCadu: WideString);
procedure CambiarDetalle( nIdLine: Double; nUnidades: Double; sNumSerie: WideString;
                          sLote: WideString; sUbicacion: WideString;  sFecCadu: WideString);
procedure BorrarDetalle(nIdLine: Double);
procedure AnadirTalla(sCodFamTallaH, sCodFamTallaV, sCodTallaH,
    sCodTallaV: WideString; nUnidades, nPrcMoneda: Double);
 procedure CambiarTalla(nIdLine, nUnidades, nPrcMoneda: Double);
 procedure BorrarTalla(nIdLine: Double);
procedure NuevoComponente;
procedure EditarComponente(nIdLine:Currency);
procedure BorrarComponente(nIdLine:Currency);
procedure AnadirComponente;
procedure CancelarComponente;
procedure AnadirTallaColor(const sCodTallaH: WideString; const sCodTallaV: WideString;
nUnidades: Double);
procedure Calcularimpuestosytotales;
function ObtenerComponentesLinea: OleVariant;
function ObtenerDetalleComponentesLinea: OleVariant;
function ObtenerDetallesLinea: OleVariant;
```

Interface ILineasDocumento (*cast* Albaran a ILineasDocumento)
function ObtenerLineas: OleVariant; **procedure AnadirSubdetalle(const ParametrosDetalle: ILineaDetalleParametros);** **procedure BorrarSubdetalle(IdLinea: Currency);** **procedure CambiarSubdetalle(const ParametrosDetalle: ILineaDetalleParametros);**

Propiedades	Tipo	Descripción
AsStringCab	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsIntegerCab	Tabla(Integer)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsFloatCab	Tabla(Float)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsBooleanCab	Tabla(Lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsCurrencyCab	Tabla(Currency)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsStringLin	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsIntegerLin	Tabla(Integer)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsFloatLin	Tabla(Float)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsBooleanLin	Tabla(Lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsCurrencyLin	Tabla(Currency)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsStringDes	Tabla(string)	Lee valores de los campos del registro nuevo/actual.
AsIntegerDes	Tabla(Integer)	Lee valores de los campos del registro nuevo/actual.
AsFloatDes	Tabla(Float)	Lee valores de los campos del registro nuevo/actual.
AsBooleanDes	Tabla(Lógico)	Lee valores de los campos del registro nuevo/actual.
AsCurrencyDes	Tabla(Currency)	Lee valores de los campos del registro nuevo/actual.
AsVariantDes	Tabla(Variant)	Lee valores de los campos del registro nuevo/actual.
Estado	EstadoMaestro	Devuelve el estado en que se encuentra el objeto
AsStringComp	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsIntegerComp	Tabla(	Asigna/lee valores a/de los campos del registro

	Integer)	nuevo/actual.
AsFloatComp	Tabla(Float)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsBooleanComp	Tabla(Lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsCurrencyComp	Tabla(Currency)	Asigna/lee valores a/de los campos del registro nuevo/actual.
OmitirMensajes	Lógico	Asignar valor para ocultar mensajes de información
ActivarAlarmaCab	Lógico	Asignar valor para activar o no alarmas en documentos
ActivarAlarmaLin	Lógico	Asignar valor para activar o no alarmas en documentos
ValidarPrecios	Lógico	Asignar valor para activar o no los mensajes de validación de precios
ValidarArtBloqueado	Lógico	Asignar valor para activar o no los mensajes de los artículos bloqueados
AvisarStock	Lógico	Asignar valor para activar o no los mensajes de aviso de stock
AvisarRiesgo	Lógico	Asignar valor para activar o no los mensajes de aviso de riesgo
AvisarCentroCosteCabecera	Lógico	Asignar valor para activar o no el mensaje que aparece al cambiar algún nivel de analítica y pregunta si se desa cambiar en las líneas.
CalcularImpuestosyTotales	Lógico	Realiza el cálculo de los impuestos y los totales del documento.

*EstadoMaestro: Ver objeto Maestro.

Método	Tipo	Descripción
Iniciar	Procedimiento	Inicia el objeto
Acabar	Procedimiento	Cierra el objeto
Nuevo	Procedimiento	Inicia un nuevo documento
Borra	Procedimiento	Borra el documento
Modifica	Procedimiento	Prepara el documento para poderlo modificar.
ModificaParcial	Procedimiento	Prepara el documento para poder cambiar solamente el detalle de una línea
Cancela	Procedimiento	Cancela la edición del documento.
Anade	Función	Añade los datos editados del documento a la base de datos y devuelve el identificador con el que se almacenará.
NuevaLinea	Procedimiento	Inicia una nueva línea
NuevaLineaArt	Procedimiento	Inicia una nueva línea aplicando las políticas de nexus, dados el código del artículo y las unidades pasadas como parámetro.

AnadirLinea	Procedimiento	Añade la línea
CancelaLin	Procedimiento	Cancela la línea en edición
EditarLinea	Procedimiento	Permite editar una línea ya existente del documento.
BorrarLinea	Procedimiento	Permite borrar una línea del documento.
IniciarServir	Procedimiento	Inicia el proceso de servir líneas de otro documento origen.
ServirDocumento	Procedimiento	Sirve el documento origen entero.
ServirLinea	Procedimiento	Sirve una línea del documento origen.
AnularLinea	Procedimiento	Anula una línea del documento origen conforme no podrán servirse las unidades indicadas.
FinServir	Procedimiento	Finaliza el proceso de servir.
AnadirDetalle	Procedimiento	Identifica los nº de serie, lotes, fechas de caducidad y/o ubicaciones de una línea del documento.
CambiarDetalle	Procedimiento	Cambia los nº de serie, lotes, fechas de caducidad y/o ubicaciones de una línea del documento.
BorrarDetalle	Procedimiento	Borra el detalle los nº de serie, lotes, fechas de caducidad y/o ubicaciones de una línea del documento.
AnadirTalla	Procedimiento	Identifica las tallas de una línea del documento.
CambiarTalla	Procedimiento	Cambia las tallas de una línea del documento.
BorrarTalla	Procedimiento	Borra el detalle de las tallas de una línea del documento.
NuevoComponente	Procedimiento	Inicia una nueva línea componente de un kit.
EditarComponente	Procedimiento	Permite editar una línea componente de un kit.
BorrarComponente	Procedimiento	Borra una línea componente de un kit.
AnadirComponente	Procedimiento	Añade una línea componente de un kit.
CancelarComponente	Procedimiento	Cancela una línea componente de un kit.
AnadirSubdetalle	Procedimiento	Permite añadir líneas de detalle (lotes, num serie…) de la línea del componente de un kit
CambiarSubdetalle	Procedimiento	Permite cambiar la línea de detalle (lotes, num serie…) de la línea del componente de un kit
BorrarSubdetalle	Procedimiento	Premite borrar la línea de detalle (lotes, num serie..) de la línea del componente de un kit

Funciones que se obtienen al convertir (*cast*) a ILineasDocumento		
ObtenerComponentesLinea	Función	Permite obtener todas las líneas de componentes del artículo kit posicionado
ObtenerDetalleComponentesLinea	Función	Permite obtener todas las líneas de detalle del componente del kit posicionado
ObtenerDetallesLinea	Función	Permite obtener todas las líneas de detalle del artículo padre posicionado
ObtenerLineas	Función	Permite obtener todas las líneas sin detalle/componentes del documento

Las primeras funciones de la lista que se presentan a continuación se utilizan para asignar / leer valores de la cabecera o de las líneas del documento.

Deben tenerse en cuenta algunas normas importantes.

- Si el nombre del campo no existe se producirá un error.
- Si el valor asignado al campo especificado no es correcto, no se producirá un error. Este se producirá al intentar guardar el documento o la línea.

Hay que tener en cuenta lo siguiente:

- Los campos fecha se han de asignar como cadenas de texto como '01/01/2003'
- Uso en Delphi: a3ERPACTIVEXDocumento.AsStringCab['CodCli']
- Uso en Visual Basic: a3ERPACTIVEXDocumento. AsStringCab ('CodCli')

property AsStringCab[sCampo:String]: String		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerCab[sCampo:String]: Integer		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatCab[sCampo:String]: Double		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanCab[sCampo:String]: Boolean		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es lógico.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsCurrencyCab[sCampo:String]: Currency		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es currency.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsVariantCab[sCampo:String]: OleVariant		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es variant.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Variant	Valor del campo

property AsStringLin[sCampo:String]: String		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerLin[sCampo:String]: Integer		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatLin[sCampo:String]: Double		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanLin[sCampo:String]: Bolean
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es lógico.

Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsCurrencyLin[sCampo:String]: Currency		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es currency.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsVariantLin[sCampo:String]: OleVariant		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es variant.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Variant	Valor del campo

property AsStringComp[sCampo:String]: String		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerComp[sCampo:String]: Integer		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatComp[sCampo:String]: Double		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanComp[sCampo:String]: Boolean		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es lógico.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsCurrencyComp[sCampo:String]: Currency		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es currency.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsVariantComp[sCampo:String]: OleVariant		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es variant.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Variant	Valor del campo

property AsStringDes[sCampo:String]: String [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerDes[sCampo:String]: Integer [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatDes[sCampo:String]: Double [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanDes[sCampo:String]: Boolean [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es lógico.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsCurrencyDes[sCampo:String]: Currency [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es currency.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsVariantDes[sCampo:String]: OleVariant [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es variant.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Variant	Valor del campo

property ValidarPrecios: WordBool		
A través de esta propiedad podemos asignar valor si queremos mostrar o ocultar los avisos de precios por debajo del precio mínimo		
Valor retornado	Lógico	Valor del campo

property ValidarArtBloqueado: WordBool		
A través de esta propiedad podemos asignar valor si queremos mostrar o ocultar los avisos de artículo bloqueado.		
Valor retornado	Lógico	Valor del campo

property AvisarStock: WordBool		
A través de esta propiedad podemos asignar valor si queremos mostrar o ocultar los avisos del stock		
Valor retornado	Lógico	Valor del campo

property AvisarRiesgo: WordBool		
A través de esta propiedad podemos asignar valor si queremos mostrar o ocultar los avisos de riesgo		
Valor retornado	Lógico	Valor del campo

property AvisarCentroCosteCabecera: WordBool		
A través de esta propiedad podemos asignar valor si queremos mostrar o ocultar el aviso para cambiar el centro de coste en las líneas cuando se cambia desde la cabecera		
Valor retornado	Lógico	Valor del campo

A continuación. se presentan las propiedades y métodos necesarios para **crear, borrar y modificar albaranes.**

property Estado:EstadoMaestro		
Esta propiedad nos indica el estado del objeto. Posibles estados: ESTM_NOACTIVO: Documento cerrado. No se puede usar. ESTM_ACTIVO: Documento activo. Podemos añadir, borrar, modificar. ESTM_NUEVO: Documento en estado de inserción de un nuevo registro. ESTM_EDICION: Documento en estado modificación, podemos asignar valores a los campos.		
Valor retornado	EstadoMaestro	Retorna el estado del documento.

procedure Iniciar;
Este procedimiento es el que reserva los recursos necesarios para que el documento pueda ser usado. Al iniciar el documento el estado cambia de ESTM_NOACTIVO a ESTM_ACTIVO.

procedure Acabar;
Este procedimiento es el que libera los recursos utilizados por el objeto. Al acabar el documento el estado cambia de ESTM_ACTIVO a ESTM_NOACTIVO.

procedure Nuevo(sFecha: String; sCodCliPro: String; bEsDeCompra: Boolean);		
Inserta un nuevo registro y los prepara para ser editado. Pasa de estado ESTM_ACTIVO a ESTM_NUEVO.		
Parámetros:	sFecha	Fecha del documento
	sCodCliPro	Código del cliente o proveedor
	bEsDeCompra	True → Compra False → Venta

procedure Modifica(nIdDocu: Currency; bEsDeCompra: WordBool);		
Pone el documento que tiene el identificador indicado en edición.		
Parámetros:	nIdDocu	Identificador del documento
	bEsDeCompra	True → Compra False → Venta

procedure ModificaParcial(IdDocu: Currency; NumLin: Integer; EsDeCompra: WordBool);		
Este método se utiliza únicamente para cambiar el detalle de una línea de un documento. Se puede utilizar en documentos con muchas líneas, mejorando la velocidad, ya que solo carga el detalle de la línea que se indica. Despues de llamar a este método solo se pueden utilizar los métodos AnadirDetalle, CambiarDetalle, BorrarDetalle, AnadirLinea, CancelaLin, Anade y Cancela. Pone el documento que tiene el identificador indicado en edición y la línea que tiene el Numlin indicado en edición		
Parámetros:	IdDocu	Identificador del documento
	NumLin	Numero de línea
	EsDeCompra	True → Compra False → Venta

Procedure Anade;
Almacena los valores de los campos en la base de dato con los cambios introducidos por el programador. Genera las repercusiones del documento como pueden ser: actualización de stocks y actualización de estadísticas. Pasa de los estados ESTM_NUEVO o EST_EDICION a ESTM_ACTIVO.

Procedure Cancela;
Cancela la edición del documento. Pasa de los estados ESTM_NUEVO o EST_EDICION a ESTM_ACTIVO.

procedure Borra(nIdDocu: Currency; bEsDeCompra: Boolean);		
Elimina el documento con el identificador indicado eliminando también sus repercusiones en stocks, contabilidad, cartera, estadísticas e IVA.		
Parámetros:	nIdDocu	Identificador del documento
	bEsDeCompra	True → Compra False → Venta

Procedure NuevaLinea;
Crea una línea con los valores por defecto. A partir de ese momento podremos editar los valores de la línea.

Procedure NuevaLineaArt(sCodArt: string; nUnidades: Currency);		
Crea una línea con el artículo y las unidades indicadas realizando las búsquedas necesarias de los valores correspondientes a cuentas, precios y descuentos, etc... A partir de ese momento podremos editar los valores de la línea.		
Parámetros:	sCodArt	Código del artículo
	nUnidades	Unidades

Procedure AnadirLinea;
Almacena los valores de los campos de la línea en la base de datos.

Procedure CancelaLin;
Cancela la edición de la línea activa.

procedure EditarLinea(nIdLinea: Currency);		
Pone la línea indicada en el parámetro nIdLinea en edición, preparada para editar el contenido de sus campos. Al final de las modificaciones deberá llamarse a AnadirLinea.		
Parámetros:	nIdLinea	Identificador de la línea. Se corresponde con el campo NUMLINALB del fichero LINEALBA. Para localizarlo debe realizarse una consulta de este campo (directamente a la base de datos) en los registros con IdAlbV = a3ERPACTIVEXAlbaran.AsStringCab['IdAlbV'] en ventas o IdAlbC = a3ERPACTIVEXAlbaran.AsStringCab['IdAlbC'] en compras.

procedure BorrarLinea(nIdLinea: Currency);		
Borra la línea con el identificador indicado en el parámetro nIdLinea.		
Parámetros:	nIdLinea	Identificador de la línea. Se corresponde con el campo NUMLINALB del fichero LINEALBA. Para localizarlo debe realizarse una consulta de este campo (directamente a la base de datos) en los registros con IdAlbV = a3ERPACTIVEXAlbaran.AsStringCab['IdAlbV'] en ventas o IdAlbC = a3ERPACTIVEXAlbaran.AsStringCab['IdAlbC'] en compras.

En la siguiente colección de procedimientos se encapsula el funcionamiento de las utilidades para servir cualquier documento sobre los albaranes. En primer lugar, debe llamarse al procedimiento IniciarServir. A continuación, pueden utilizarse los procedimientos ServirDocumento, ServirLinea o AnularLinea y funalmente debe llamarse a FinServir.

procedure IniciarServir(sIdDocu: string; nIdDocu: Currency; bConMensaje: Boolean);		
Prepara el documento indicado para servirse. Reserva los recursos necesarios.		
Parámetros:	sIdDocu	'O' → Oferta 'P' → Pedido
	nIdDocu	En función del valor de sIdDocu (O,P) representa el identificador de la oferta (IdOfeV o IdOfeC) o del pedido (IdPEdV o IdPedC) respectivamente.
	bConMensaje	En caso en que se produzcan diferencias entre las condiciones del documento origen y el destino (Por ejemplo, la forma de pago, el documento de pago, el transportista, etc....), el programa puede actuar de dos formas distintas: - bConMesnaje = True: Preguntar al usuario si se desean aplicar las condiciones del documento destino o respetar las del documento origen. - bConMensaje = False: Aplicar las condiciones del documento destino.

procedure ServirDocumento;
Sirve todo el material pendiente de todo del documento origen indicado en la función IniciarServir.

procedure ServirLinea(nNumGrupo nNumLin: Currency; nNumBul: Currency; nNumPaq: Currency; nNumUni: Double; sNSerie: WideString; sLote: WideString; sUbicacion: WideString; sFecCad: WideString);
Sirve la línea con los parámetros indicados.

Parámetros:	nNumGrupo	Obsoleto, poner un cero.
	nNumLin	Nº de línea. NUMLINOFE en ofertas y NUMLINPED en pedidos
	nNumBul	Nº de bultos de la línea que desean servirse.
	nNumPaq	Nº de paquetes que desean servirse.
	nNumUni	Nº de unidades que desean servirse.
	sNSerie	Nº de serie.
	sLote	Lote.
	sUbicacion	Ubicación
	sFecCad	Fecha de caducidad.

procedure AnularLinea(nNumLin: Currency; nNumBul: Currency; nNumPaq: Currency; nNumUni: Double);		
Anula las unidades indicadas de la línea pasada por parámetro. Esas unidades no estarán disponibles para ser servidas posteriormente.		
Parámetros:	nNumLin	Nº de línea del documento origen (NUMLINOFE o NUMLINPED).
	nNumBul	Nº de bultos de la línea que desean servirse.
	nNumPaq	Nº de paquetes que desean servirse.
	nNumUni	Nº de unidades que desean servirse.

procedure FinServir;
Realiza las operaciones realizadas desde que se llamó a IniciarServir de forma definitiva. Libera los recursos necesarios.

Los procedimientos que se describen a continuación se utilizan para **especificar los lotes, nº de serie, fechas de caducidad y ubicaciones de una línea** a no ser que esta selección se realice de forma automática. Para consultar si ya se han detallado todos los lotes, nº de serie, fechas de caducidad y ubicaciones necesarias se debe llamar a la función ConsultarDetalle;

procedure AnadirDetalle(nUnidades: Double; sNumSerie: WideString; sLote: WideString; sUbicacion: WideString; sFecCadu: WideString);		
Indica las unidades necesarias y su nº serie, lote, fecha de caducidad y ubicación de la línea activa. Para conocer los nº serie, lote, fecha de caducidad y ubicaciones disponibles en stock hay que llamar a la función **Varios.DetalleStockDisponible**..		
Parámetros:	nNunidades	Nº de unidades de la línea de detalle.
	sNumSerie	Nº de serie.
	sLote	Lote.
	sUbicacion	Ubicación.
	sFecCadu	Fecha de caducidad.

procedure CambiarDetalle(nIdLine: Double; sNumSerie: WideString; sLote: WideString; sUbicacion: WideString; sFecCadu: WideString);		
Indica las unidades necesarias y su Nº serie, lote, fecha de caducidad y ubicación de la línea activa. Para conocer los nº serie, lote, fecha de caducidad y ubicaciones disponibles en stock hay que llamar a la función **Varios.DetalleStockDisponible**. Para consultar el detalle de la línea activa hay que llamar a la función **ConsultarDetalle**.		
Parámetros:	nIdLine	Identificador de la línea. Se corresponde con el campo IDLIN de la tabla LINEALBA.
	nUnidades	Nº de unidades de la línea de detalle.
	sNumSerie	Nº de serie.
	sLote	Lote.
	sUbicacion	Ubicación.
	sFecCadu	Fecha de caducidad.

procedure BorrarDetalle(nIdLine: Double);		
Elimina la línea de detalle indicada en el parámetro.		
Parámetros:	nIdLine	Identificador de la línea. Se corresponde con el campo IDLIN de la tabla LINEALBA.

procedure AnadirTalla(sCodFamTallaH, sCodFamTallaV, sCodTallaH, sCodTallaV: WideString; nUnidades, nPrcMoneda:Double);		
Indica las unidades y el precio necesarios, junto con sus tallas y colores de la línea activa.		
Parámetros:	sCodFamTallaH	Familia de talla horizontal
	sCodFamTallaV	Familia de talla vertical
	sCodTallaH	Código de talla horizontal
	sCodTallaV	Código de talla vertical
	nUnidades	Nº unidades de la línea de talla
	nPrcMoneda	Precio de la línea de talla.

procedure CambiarTalla(nIdLine, nUnidades, nPrcMoneda: Double);		
Indica las unidades y el precio de la línea activa.		
Parámetros:	nIdLine	Identificador de la línea. Se corresponde con el campo IDLIN de la tabla LINEALBA.
	nUnidades	Nº de unidades de la línea de talla.
	nPrcMoneda	Precio de la línea de talla.

procedure BorrarTalla(nIdLine: Double);		
Elimina la línea de talla indicada en el parámetro.		
Parámetros:	nIdLine	Identificador de la línea. Se corresponde con el campo IDLIN de la tabla LINEALBA.

Procedure NuevoComponente;
Crea una línea con los valores por defecto. A partir de ese momento podremos editar los valores de la línea componente

Procedure AnadirComponente;
Almacena los valores de los campos de la línea componente en la base de datos.

Procedure CancelaComponente;
Cancela la edición de la línea componente activa.

procedure EditarComponente(nIdLinea: Currency);		
Pone la línea componente indicada en el parámetro nIdLinea en edición, preparada para editar el contenido de sus campos. Al final de las modificaciones deberá llamarse a AnadirComponente.		
Parámetros:	nIdLinea	Identificador de la línea componente. Se corresponde con el campo NUMLINALB del fichero LINEALBA.

procedure BorrarComponente(nIdLinea: Currency);		
Borra la línea componente con el identificador indicado en el parámetro nIdLinea.		
Parámetros:	nIdLinea	Identificador de la línea componente. Se corresponde con el campo NUMLINALB del fichero LINEALBA.

procedure AnadirTallaColor(sCodTallaH, sCodTallaV: WideString; nUnidades:Double);		
Indica las unidades, junto con su talla y color de la línea activa.		
Parámetros:	sCodTallaH	Código de talla horizontal
	sCodTallaV	Código de talla vertical
	nUnidades	Nº unidades de la línea de talla

procedure CalcularImpuestosyTotales;
Realiza el cálculo de los impuestos y los totales del documento.

procedure AnadirSubdetalle(const ParametrosDetalle: ILineaDetalleParametros);		
Permite añadir líneas de detalle (lotes, num serie…) de la línea del componente de un kit		
Parámetros:	ParametrosDetalle	Clase que implementa la interfaz ILineaDetalleParametros. La que nos permite informar del detalle a añadir

procedure BorrarSubdetalle(IdLinea: Currency);		
Premite borrar la línea de detalle (lotes, num serie..) de la línea del componente de un kit		
Parámetros:	IdLinea	Identificador de la línea (campo IDLIN) para localizar el detalle y borrarlo

procedure CambiarSubdetalle(const ParametrosDetalle: ILineaDetalleParametros);		
Permite cambiar la línea de detalle (lotes, num serie…) de la línea del componente de un kit.		
Parámetros:	ParametrosDetalle	Clase que implementa la interfaz ILineaDetalleParametros. La que nos permite informar del detalle a modificar. Si el valor de la línea es -1 cambiará el registro posicionado sin tener necesidad de buscarlo. Pensado para cuando sólo hay un detalle

function ILineasDocumento.ObtenerComponentesLinea: OleVariant
Permite obtener todas las líneas de componentes del artículo kit posicionado Devuelve una array de arrays con la siguiente estructura: A[0] => Numero líneas de componentes A[1..N][0] => Número de campos. Ejemplo 150 campos A[1..n][1..n][0] => Nombre del campo. Ejemplo 'CodArt' A[1..n][1..n][1] => Valor del campo. Ejemplo ' 1'

function ILineasDocumento.ObtenerDetalleComponentesLinea: OleVariant;
FunciónPermite obtener todas las líneas de detalle del componente del kit posicionado Devuelve una array de arrays con la siguiente estructura: A[0] => Numero líneas de detalle A[1..N][0] => Número de campos. Ejemplo 150 campos A[1..n][1..n][0] => Nombre del campo. Ejemplo 'CodArt' A[1..n][1..n][1] => Valor del campo. Ejemplo 'LOTE100'

function ILineasDocumento.ObtenerDetallesLinea: OleVariant;
Permite obtener todas las líneas de detalle del artículo padre posicionado Devuelve una array de arrays con la siguiente estructura: A[0] => Numero líneas de detalle A[1..N][0] => Número de campos. Ejemplo 150 campos A[1..n][1..n][0] => Nombre del campo. Ejemplo 'CodArt' A[1..n][1..n][1] => Valor del campo. Ejemplo 'LOTE100'

function ILineasDocumento.ObtenerLineas: OleVariant;
Permite obtener todas las líneas sin detalle/componentes del documento Devuelve una array de arrays con la siguiente estructura: A[0] => Numero líneas del documento A[1..N][0] => Número de campos. Ejemplo 150 campos A[1..n][1..n][0] => Nombre del campo. Ejemplo 'CodArt' A[1..n][1..n][1] => Valor del campo. Ejemplo 'B10

11.- OBJETO ALBARAN

Extiende Albaran140501 y añade nuevos métodos.

Interface Albaran
procedure Clonar(const DocumentoOrigenTipo: WideString; DocumentoOrigenID: Currency; BorrarLineasDelDocumentoDestino: WordBool; ClonarVariablesDeDocumentos: WordBool); [Nuevo 14.05.02] **procedure Copiar(const DocumentoOrigenTipo: WideString; DocumentoOrigenID: Currency; BorrarLineasDelDocumentoDestino: WordBool; CopiarCondicionesFinancieras: WordBool; CopiarLosPortes: WordBool; CopiarPreciosYDescuentos: WordBool; CopiarDescripcionesYObservacionesLineas: WordBool);** [Nuevo 14.05.02]

procedure Clonar(const DocumentoOrigenTipo: WideString; DocumentoOrigenID: Currency; BorrarLineasDelDocumentoDestino: WordBool; ClonarVariablesDeDocumentos: WordBool);
Mediante este procedimiento se permite clonar un documento sobre otro.
Parámetros:

DocumentoOrigenTipo	'OV' Oferta venta 'PV' Pedido de venta 'AV' Albarán de venta 'DV' Depósito de venta 'FV' Factura de venta	'OC' Oferta compra 'PC' Pedido de compra 'AC' Albarán de compra 'DC' Depósito de compra 'FC' Factura de compra
DocumentoOrigenID	Identificador del documento de origen	
BorrarLineasDelDocumentoDestino	Antes de realizar la operación, borra las lineas del documento destino	
ClonarVariablesDeDocumentos	Se clonarán las variables de los documentos	

procedure Copiar(const DocumentoOrigenTipo: WideString; DocumentoOrigenID: Currency; BorrarLineasDelDocumentoDestino: WordBool; CopiarCondicionesFinancieras: WordBool; CopiarLosPortes: WordBool; CopiarPreciosYDescuentos: WordBool; CopiarDescripcionesYObservacionesLineas: WordBool);		
Mediante este procedimiento se permite copiar un documento sobre otro.		
Parámetros:		
DocumentoOrigenTipo	'OV' Oferta venta 'PV' Pedido de venta 'AV' Albarán de venta 'DV' Depósito de venta 'FV' Factura de venta	'OC' Oferta compra 'PC' Pedido de compra 'AC' Albarán de compra 'DC' Depósito de compra 'FC' Factura de compra
DocumentoOrigenID	Identificador del documento de origen	
BorrarLineasDelDocumentoDestino	Antes de realizar la operación, borra las lineas del documento destino	
CopiarCondicionesFinancieras	Copiar condiciones financieras del documento origen. Se copiarán del documento origen la forma de pago, documento de pago, descuento de pronto pago o recargo financiero y si éste afecta a la base imponible.	
CopiarLosPortes	Copiar los portes al documento destino	
CopiarPreciosYDescuentos	Copiar precios y descuentos del documento origen	
CopiarDescripcionesYObservacionesLineas	Copiar descripciones y observaciones de las líneas del documento origen	

12.- OBJETO PEDIDO140501

Objeto que permite la creación, modificación y borrado de depósitos.

Interface Pedido
property Estado: EstadoMaestro readonly; **property** AsStringCab[const sCampo: WideString]: WideString; **property** AsFloatCab[const sCampo: WideString]: Double; **property** AsIntegerCab[const sCampo: WideString]: Integer; **property** AsBooleanCab[const sCampo: WideString]: WordBool; **property** AsCurrencyCab[const sCampo: WideString]: Currency; **property** AsVariantCab[const sCampo: WideString]: OleVariant; **property** AsStringLin[const sCampo: WideString]: WideString; **property** AsFloatLin[const sCampo: WideString]: Double; **property** AsIntegerLin[const sCampo: WideString]: Integer; **property** AsBooleanLin[const sCampo: WideString]: WordBool; **property** AsCurrencyLin[const sCampo: WideString]: Currency; **property** AsVariantLin[const sCampo: WideString]: OleVariant; **property** AsBooleanComp[const sCampo: WideString]: WordBool; **property** AsStringComp[const sCampo: WideString]: WideString; **property** AsFloatComp[const sCampo: WideString]: Double; **property** AsIntegerComp[const sCampo: WideString]: Integer; **property** AsCurrencyComp[const sCampo: WideString]: Currency; **property** AsVariantComp[const sCampo: WideString]: OleVariant; **property** AsBooleanDes[const sCampo: WideString]: WordBool; [SOLO LECTURA] **property** AsStringDes[const sCampo: WideString]: WideString; [SOLO LECTURA] **property** AsFloatDes[const sCampo: WideString]: Double; [SOLO LECTURA] **property** AsIntegerDes[const sCampo: WideString]: Integer; [SOLO LECTURA] **property** AsCurrencyDes[const sCampo: WideString]: Currency; [SOLO LECTURA] **property** AsVariantDes[const sCampo: WideString]: OleVariant; [SOLO LECTURA] **property** OmitirMensajes:WordBool; **property** ActivarAlarmaCab : WordBool; **property** ActivarAlarmaLin: WordBool; **property** ValidarPrecios: WordBool; **property** ValidarArtBloqueado: WordBool; **property** AvisarRiesgo: WordBool; **property** AvisarCentroCosteCabecera: WordBool; **procedure** Iniciar; **procedure** Acabar; **procedure** Cancela; **procedure** Nuevo(sFecha: WideString; sCodCliPro: WideString; bEsDeCompra: WordBool); **procedure** Borra(IdDocu: Currency; bEsDeCompra: WordBool); **procedure** Modifica(IdDocu: Currency; bEsDeCompra: WordBool); **procedure** ModificaParcial(IdDocu: Currency; NumLin: Integer, EsDeCompra: WordBool);

```
function Anade: Currency;
procedure NuevaLinea;
procedure AnadirLinea;
procedure CancelaLin;
procedure NuevaLineaArt(const sCodArt: WideString; nUnidades: Currency);
procedure EditarLinea(nIdLinea: Currency);
procedure BorrarLinea(nIdLinea: Currency);
procedure IniciarServir(const sIdDocu: WideString; nIdDocu: Currency; bConMensaje:
WordBool);
procedure ServirDocumento;
procedure FinServir;
procedure ServirLinea( nNumGru: Currency; nNumLin: Currency; nNumBul: Currency;
                       nNumPaq: Currency; nNumUni: Double; nPrcMed: Double;
                       sNSerie: WideString; sLote: WideString; sUbicac: WideString;
                       sFecCad: WideString);
procedure AnularLinea( nNumLin: Currency; nNumBul: Currency; nNumPaq: Currency;
                        nNumUni: Double);
procedure AnadirDetalle( nUnidades: Double; sNumSerie: WideString; sLote: WideString;
                         sUbicacion: WideString; sFecCadu: WideString);
procedure CambiarDetalle( nIdLine: Double; nUnidades: Double; sNumSerie: WideString;
                          sLote: WideString; sUbicacion: WideString;  sFecCadu: WideString);
procedure BorrarDetalle(nIdLine: Double);
procedure AnadirTalla(sCodFamTallaH, sCodFamTallaV, sCodTallaH,
    sCodTallaV: WideString; nUnidades, nPrcMoneda: Double);
 procedure CambiarTalla(nIdLine, nUnidades, nPrcMoneda: Double);
 procedure BorrarTalla(nIdLine: Double);
procedure NuevoComponente;
procedure EditarComponente(nIdLine:Currency);
procedure BorrarComponente(nIdLine:Currency);
procedure AnadirComponente;
procedure CancelarComponente;
procedure AnadirTallaColor(const sCodTallaH: WideString; const sCodTallaV: WideString;
nUnidades: Double);
function AnularUnidades(IdLin: Currency; Bultos: Double; Paquetes: Double; Unidades:
Double; Fecha: TDateTime; const Motivo: WideString): Currency;
procedure DeshacerAnularUnidades(IdLin: Currency; IdA3ERPACTIVEX: Currency);
procedure CalcularImpuestosyTotales;
function ObtenerComponentesLinea: OleVariant;
function ObtenerDetalleComponentesLinea: OleVariant;
function ObtenerDetallesLinea: OleVariant;
```

Interface ILineasDocumento (*cast* Pedido a ILineasDocumento)
function ObtenerLineas: OleVariant; **procedure AnadirSubdetalle(const ParametrosDetalle: ILineaDetalleParametros);** **procedure BorrarSubdetalle(IdLinea: Currency);** **procedure CambiarSubdetalle(const ParametrosDetalle: ILineaDetalleParametros);**

Propiedades	Tipo	Descripción
AsStringCab	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsIntegerCab	Tabla(Integer)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsFloatCab	Tabla(Float)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsBooleanCab	Tabla(Lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsCurrencyCab	Tabla(Currency)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsStringLin	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsIntegerLin	Tabla(Integer)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsFloatLin	Tabla(Float)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsBooleanLin	Tabla(Lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsCurrencyLin	Tabla(Currency)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsStringDes	Tabla(string)	Lee valores de los campos del registro nuevo/actual.
AsIntegerDes	Tabla(Integer)	Lee valores de los campos del registro nuevo/actual.
AsFloatDes	Tabla(Float)	Lee valores de los campos del registro nuevo/actual.
AsBooleanDes	Tabla(Lógico)	Lee valores de los campos del registro nuevo/actual.
AsCurrencyDes	Tabla(Currency)	Lee valores de los campos del registro nuevo/actual.
AsVariantDes	Tabla(Variant)	Lee valores de los campos del registro nuevo/actual.
Estado	EstadoMaestro	Devuelve el estado en que se encuentra el objeto
AsStringComp	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsIntegerComp	Tabla(Integer)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsFloatComp	Tabla(Float)	Asigna/lee valores a/de los campos del registro nuevo/actual.

AsBooleanComp	Tabla(Lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsCurrencyComp	Tabla(Currency)	Asigna/lee valores a/de los campos del registro nuevo/actual.
OmitirMensajes	Lógico	Asignar valor para ocultar mensajes de información
ActivarAlarmaCab	Lógico	Asignar valor para activar o no alarmas en documentos
ActivarAlarmaLin	Lógico	Asignar valor para activar o no alarmas en documentos
ValidarPrecios	Lógico	Asignar valor para activar o no los mensajes de validación de precios
ValidarArtBloqueado	Lógico	Asignar valor para activar o no los mensajes de los artículos bloqueados
AvisarRiesgo	Lógico	Asignar valor para activar o no los mensajes de aviso de riesgo
AvisarCentroCosteCabecera	Lógico	Asignar valor para activar o no el mensaje que aparece al cambiar algún nivel de analítica y pregunta si se desa cambiar en las líneas.
CalcularImpuestosyTotales	Lógico	Realiza el cálculo de los impuestos y los totales del documento.

*EstadoMaestro: Ver objeto Maestro.

Método	**Tipo**	**Descripción**
Iniciar	Procedimiento	Inicia el objeto
Acabar	Procedimiento	Cierra el objeto
Nuevo	Procedimiento	Inicia un nuevo documento
Borra	Procedimiento	Borra el documento
Modifica	Procedimiento	Prepara el documento para poderlo modificar.
ModificaParcial	Procedimiento	Prepara el documento para poder cambiar solamente el detalle de una línea
Cancela	Procedimiento	Cancela la edición del documento.
Anade	Función	Añade los datos editados del documento a la base de datos y devuelve el identificador con el que se almacenará.
NuevaLinea	Procedimiento	Inicia una nueva línea
NuevaLineaArt	Procedimiento	Inicia una nueva línea aplicando las políticas de nexus, dados el código del artículo y las unidades pasadas como parámetro.
AnadirLinea	Procedimiento	Añade la línea
CancelaLin	Procedimiento	Cancela la línea en edición

EditarLinea	Procedimiento	Permite editar una línea ya existente del documento.
BorrarLinea	Procedimiento	Permite borrar una línea del documento.
IniciarServir	Procedimiento	Inicia el proceso de servir líneas de otro documento origen.
ServirDocumento	Procedimiento	Sirve el documento origen entero.
ServirLinea	Procedimiento	Sirve una línea del documento origen.
AnularLinea	Procedimiento	Anula una línea del documento origen conforme no podrán servirse las unidades indicadas.
FinServir	Procedimiento	Finaliza el proceso de servir.
AnadirDetalle	Procedimiento	Identifica los nº de serie, lotes, fechas de caducidad y/o ubicaciones de una línea del documento.
CambiarDetalle	Procedimiento	Cambia los nº de serie, lotes, fechas de caducidad y/o ubicaciones de una línea del documento.
BorrarDetalle	Procedimiento	Borra el detalle los nº de serie, lotes, fechas de caducidad y/o ubicaciones de una línea del documento.
AnadirTalla	Procedimiento	Identifica las tallas de una línea del documento.
CambiarTalla	Procedimiento	Cambia las tallas de una línea del documento.
BorrarTalla	Procedimiento	Borra el detalle de las tallas de una línea del documento.
NuevoComponente	Procedimiento	Inicia una nueva línea componente de un kit.
EditarComponente	Procedimiento	Permite editar una línea componente de un kit.
BorrarComponente	Procedimiento	Borra una línea componente de un kit.
AnadirComponente	Procedimiento	Añade una línea componente de un kit.
CancelarComponente	Procedimiento	Cancela una línea componente de un kit.
AnadirSubdetalle	Procedimiento	Permite añadir líneas de detalle (lotes, num serie…) de la línea del componente de un kit
CambiarSubdetalle	Procedimiento	Permite cambiar la línea de detalle (lotes, num serie…) de la línea del componente de un kit
BorrarSubdetalle	Procedimiento	Premite borrar la línea de detalle (lotes, num serie..) de la línea del componente de un kit

Funciones que se obtienen al convertir (*cast*) a ILineasDocumento		
ObtenerComponentesLinea	Función	Permite obtener todas las líneas de componentes del artículo kit posicionado
ObtenerDetalleComponentesLinea	Función	Permite obtener todas las líneas de detalle del componente del kit posicionado
ObtenerDetallesLinea	Función	Permite obtener todas las líneas de detalle del artículo padre posicionado

ObtenerLineas	Función	Permite obtener todas las líneas sin detalle/componentes del documento

Las primeras funciones de la lista que se presentan a continuación se utilizan **para asignar / leer valores de la cabecera o de las líneas del documento.**

Deben tenerse en cuenta algunas normas importantes.

- Si el nombre del campo no existe se producirá un error.
- Si el valor asignado al campo especificado no es correcto, no se producirá un error. Este se producirá al intentar guardar el documento o la línea.

Hay que tener en cuenta lo siguiente:

- Los campos fecha se han de asignar como cadenas de texto como '01/01/2003'
- Uso en Delphi: a3ERPACTIVEXDocumento.AsStringCab['CodCli']
- Uso en Visual Basic: a3ERPACTIVEXDocumento. AsStringCab ('CodCli')

property AsStringCab[sCampo:String]: String		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerCab[sCampo:String]: Integer		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatCab[sCampo:String]: Double		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanCab[sCampo:String]: Boolean		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es lógico.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsCurrencyCab[sCampo:String]: Currency		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es currency.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsVariantCab[sCampo:String]: OleVariant		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es variant.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Variant	Valor del campo

property AsStringLin[sCampo:String]: String		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerLin[sCampo:String]: Integer		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatLin[sCampo:String]: Double		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanLin[sCampo:String]: Boolean
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es lógico.

Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsCurrencyLin[sCampo:String]: Currency		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es currency.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsVariantLin[sCampo:String]: Currency		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es variant.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsStringComp[sCampo:String]: String		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerComp[sCampo:String]: Integer		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatComp[sCampo:String]: Double		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanComp[sCampo:String]: Boolean		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es lógico.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsCurrencyComp[sCampo:String]: Currency		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es currency.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsVariantComp[sCampo:String]: Currency		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es variant.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsStringDes[sCampo:String]: String [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerDes[sCampo:String]: Integer [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatDes[sCampo:String]: Double [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanDes[sCampo:String]: Boolean [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es lógico.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsCurrencyDes[sCampo:String]: Currency [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es currency.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsVariantDes[sCampo:String]: OleVariant [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es variant.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Variant	Valor del campo

property ValidarPrecios: WordBool		
A través de esta propiedad podemos asignar valor si queremos mostrar o ocultar los avisos de precios por debajo del precio mínimo		
Valor retornado	Lógico	Valor del campo

property ValidarArtBloqueado: WordBool
A través de esta propiedad podemos asignar valor si queremos mostrar o ocultar los avisos de artículo bloqueado

Valor retornado	Lógico	Valor del campo

property AvisarStock: WordBool		
A través de esta propiedad podemos asignar valor si queremos mostrar o ocultar los avisos del stock		
Valor retornado	Lógico	Valor del campo

property AvisarRiesgo: WordBool		
A través de esta propiedad podemos asignar valor si queremos mostrar o ocultar los avisos de riesgo		
Valor retornado	Lógico	Valor del campo

property AvisarCentroCosteCabecera: WordBool		
A través de esta propiedad podemos asignar valor si queremos mostrar o ocultar el aviso para cambiar el centro de coste en las líneas cuando se cambia desde la cabecera		
Valor retornado	Lógico	Valor del campo

A continuación, se presentan las propiedades y métodos necesarios para crear, borrar y modificar pedidos.

property Estado:EstadoMaestro		
Esta propiedad nos indica el estado del objeto. Posibles estados: ESTM_NOACTIVO: Documento cerrado. No se puede usar. ESTM_ACTIVO: Documento activo. Podemos añadir, borrar, modificar. ESTM_NUEVO: Documento en estado de inserción de un nuevo registro. ESTM_EDICION: Documento en estado modificación, podemos asignar valores a los campos.		
Valor retornado	EstadoMaestro	Retorna el estado del documento.

procedure Iniciar;
Este procedimiento es el que reserva los recursos necesarios para que el documento pueda ser usado. Al iniciar el documento el estado cambia de ESTM_NOACTIVO a ESTM_ACTIVO.

procedure Acabar;
Este procedimiento es el que libera los recursos utilizados por el objeto. Al acabar el documento el estado cambia de ESTM_ACTIVO a ESTM_NOACTIVO.

procedure Nuevo(sFecha: String; sCodCliPro: String; bEsDeCompra: Boolean);		
Inserta un nuevo registro y los prepara para ser editado. Pasa de estado ESTM_ACTIVO a ESTM_NUEVO.		
Parámetros:	sFecha	Fecha del documento
	sCodCliPro	Código del cliente o proveedor
	bEsDeCompra	True → Compra False → Venta

procedure Modifica(nIdDocu: Currency; bEsDeCompra: WordBool);		
Pone el documento que tiene el identificador indicado en edición.		
Parámetros:	nIdDocu	Identificador del documento
	bEsDeCompra	True → Compra False → Venta

procedure ModificaParcial(IdDocu: Currency; NumLin: Integer; EsDeCompra: WordBool);		
Este método se utiliza únicamente para cambiar el detalle de una línea de un documento. Se puede utilizar en documentos con muchas líneas, mejorando la velocidad, ya que solo carga el detalle de la línea que se indica. Despues de llamar a este método solo se pueden utilizar los métodos AnadirDetalle, CambiarDetalle, BorrarDetalle, AnadirLinea, CancelaLin, Anade y Cancela. Pone el documento que tiene el identificador indicado en edición y la línea que tiene el Numlin indicado en edición		
Parámetros:	IdDocu	Identificador del documento
	NumLin	Numero de línea
	EsDeCompra	True → Compra False → Venta

Procedure Anade;
Almacena los valores de los campos en la base de dato con los cambios introducidos por el programador. Genera las repercusiones del documento como pueden ser: actualización de stocks y actualización de estadísticas. Pasa de los estados ESTM_NUEVO o EST_EDICION a ESTM_ACTIVO.

Procedure Cancela;
Cancela la edición del documento. Pasa de los estados ESTM_NUEVO o EST_EDICION a ESTM_ACTIVO.

procedure Borra(nIdDocu: Currency; bEsDeCompra: Boolean);		
Elimina el documento con el identificador indicado eliminando también sus repercusiones en stocks, contabilidad, cartera, estadísticas e IVA.		
Parámetros:	nIdDocu	Identificador del documento
	bEsDeCompra	True → Compra False → Venta

Procedure NuevaLinea;
Crea una línea con los valores por defecto. A partir de ese momento podremos editar los valores de la línea.

Procedure NuevaLineaArt(sCodArt: string; nUnidades: Currency);		
Crea una línea con el artículo y las unidades indicadas realizando las búsquedas necesarias de los valores correspondientes a cuentas, precios y descuentos, etc... A partir de ese momento podremos editar los valores de la línea.		
Parámetros:	sCodArt	Código del artículo
	nUnidades	Unidades

Procedure AnadirLinea;
Almacena los valores de los campos de la línea en la base de datos.

Procedure CancelaLin;
Cancela la edición de la línea activa.

procedure EditarLinea(nIdLinea: Currency);		
Pone la línea indicada en el parámetro nIdLinea en edición, preparada para editar el contenido de sus campos. Al final de las modificaciones deberá llamarse a AnadirLinea.		
Parámetros:	nIdLinea	Identificador de la línea.. Se corresponde con el campo NUMLINPED del fichero LINEPEDI. Para localizarlo debe realizarse una consulta de este campo (directamente a la base de datos) en los registros con IdPedV = a3ERPACTIVEXPedido.AsStringCab['IdPedV'] en ventas o IdPedC = a3ERPACTIVEXPedido.AsStringCab['IdPedC'] en compras.

procedure BorrarLinea(nIdLinea: Currency);		
Borra la línea con el identificador indicado en el parámetro nIdLinea.		
Parámetros:	nIdLinea	Identificador de la línea. Se corresponde con el campo NUMLINPED del fichero LINEPEDI. Para localizarlo debe realizarse una consulta de este campo (directamente a la base de datos) en los registros con IdPedV = a3ERPACTIVEXPedido.AsStringCab['IdPedV'] en ventas o IdPedC = a3ERPACTIVEXPedido.AsStringCab['IdPedC'] en compras.

En la siguiente colección de procedimientos se encapsula el funcionamiento de las utilidades para **servir cualquier documento sobre los pedidos**. En primer lugar, debe llamarse al procedimiento IniciarServir. A continuación, pueden utilizarse los procedimientos ServirDocumento, ServirLinea o AnularLinea y funalmente debe llamarse a FinServir.

procedure IniciarServir(sIdDocu: string; nIdDocu: Currency; bConMensaje: Boolean);		
Prepara el documento indicado para servirse. Reserva los recursos necesarios.		
Parámetros:	sIdDocu	'O' → Oferta
	nIdDocu	Representa el identificador de la oferta (IdOfeV o IdOfeC).
	bConMensaje	En caso en que se produzcan diferencias entre las condiciones del documento origen y el destino (Por ejemplo, la forma de pago, el documento de pago, el transportista, etc....), el programa puede actuar de dos formas distintas: - bConMesnaje = True: Preguntar al usuario si se desean aplicar las condiciones del documento destino o respetar las del documento origen. - bConMensaje = False: Aplicar las condiciones del documento destino.

procedure ServirDocumento;
Sirve todo el material pendiente de todo del documento origen indicado en la función IniciarServir.

procedure ServirLinea(nNumGru, nNumLin: Currency; nNumBul: Currency; nNumPaq: Currency; nNumUni: Double; sNSerie: WideString; sLote: WideString; sUbicac: WideString; sFecCad: WideString);
Sirve la línea con los parámetros indicados.

Parámetros:	nNumGru	Obsoleto, poner un cero.
	nNumLin	Nº de línea. NUMLINOFE de ofertas.
	nNumBul	Nº de bultos de la línea que desean servirse.
	nNumPaq	Nº de paquetes que desean servirse.
	nNumUni	Nº de unidades que desean servirse.
	sNSerie	Nº de serie.
	sLote	Lote.
	sUbicac	Ubicación
	sFecCad	Fecha de caducidad.

procedure AnularLinea(nNumLin: Currency; nNumBul: Currency; nNumPaq: Currency; nNumUni: Double);		
Anula las unidades indicadas de la línea pasada por parámetro. Esas unidades no estarán disponibles para ser servidas posteriormente.		
Parámetros:	nNumLin	Nº de línea del documento origen (NUMLINOFE).
	nNumBul	Nº de bultos de la línea que desean servirse.
	nNumPaq	Nº de paquetes que desean servirse.
	nNumUni	Nº de unidades que desean servirse.

procedure FinServir;
Realiza las operaciones realizadas desde que se llamó a IniciarServir de forma definitiva. Libera los recursos necesarios.

Los procedimientos que se describen a continuación se utilizan para **especificar los lotes, nº de serie, fechas de caducidad y ubicaciones de una línea** a no ser que esta selección se realice de forma automática. Para consultar si ya se han detallado todos los lotes, nº de serie, fechas de caducidad y ubicaciones necesarias se debe llamar a la función ConsultarDetalle;

procedure AnadirDetalle(nUnidades: Double; sNumSerie: WideString; sLote: WideString; sUbicacion: WideString; sFecCadu: WideString);		
Indica las unidades necesarias y su nº serie, lote, fecha de caducidad y ubicación de la línea activa. Para conocer los nº serie, lote, fecha de caducidad y ubicaciones disponibles en stock hay que llamar a la función **Varios.DetalleStockDisponible**..		
Parámetros:	nNunidades	Nº de unidades de la línea de detalle.
	sNumSerie	Nº de serie.
	sLote	Lote.
	sUbicacion	Ubicación.
	sFecCadu	Fecha de caducidad.

procedure CambiarDetalle(nIdLine: Double; sNumSerie: WideString; sLote: WideString; sUbicacion: WideString; sFecCadu: WideString);		
Indica las unidades necesarias y su Nº serie, lote, fecha de caducidad y ubicación de la línea activa. Para conocer los nº serie, lote, fecha de caducidad y ubicaciones disponibles en stock hay que llamar a la función **Varios.DetalleStockDisponible**. Para consultar el detalle de la línea activa hay que llamar a la función **ConsultarDetalle**.		
Parámetros:	nIdLine	Identificador de la línea. Se corresponde con el campo IDLIN de la tabla LINEPEDI.
	nUnidades	Nº de unidades de la línea de detalle.
	sNumSerie	Nº de serie.
	sLote	Lote.
	sUbicacion	Ubicación.
	sFecCadu	Fecha de caducidad.

procedure BorrarDetalle(nIdLine: Double);		
Elimina la línea de detalle indicada en el parámetro.		
Parámetros:	nIdLine	Identificador de la línea. Se corresponde con el campo IDLIN de la tabla LINEPEDI.

procedure AnadirTalla(sCodFamTallaH, sCodFamTallaV, sCodTallaH, sCodTallaV: WideString; nUnidades, nPrcMoneda:Double);		
Indica las unidades y el precio necesarios, junto con sus tallas y colores de la línea activa.		
Parámetros:	sCodFamTallaH	Familia de talla horizontal
	sCodFamTallaV	Familia de talla vertical
	sCodTallaH	Código de talla horizontal
	sCodTallaV	Código de talla vertical
	nUnidades	Nº unidades de la línea de talla
	nPrcMoneda	Precio de la línea de talla.

procedure CambiarTalla(nIdLine, nUnidades, nPrcMoneda: Double);		
Indica las unidades y el precio de la línea activa.		
Parámetros:	nIdLine	Identificador de la línea. Se corresponde con el campo IDLIN de la tabla LINEPEDI.
	nUnidades	Nº de unidades de la línea de talla.
	nPrcMoneda	Precio de la línea de talla.

procedure BorrarTalla(nIdLine: Double);		
Elimina la línea de talla indicada en el parámetro.		
Parámetros:	nIdLine	Identificador de la línea. Se corresponde con el campo IDLIN de la tabla LINEPEDI.

Procedure NuevoComponente;
Crea una línea con los valores por defecto. A partir de ese momento podremos editar los valores de la línea componente

Procedure AnadirComponente;
Almacena los valores de los campos de la línea componente en la base de datos.

Procedure CancelaComponente;
Cancela la edición de la línea componente activa.

procedure EditarComponente(nIdLinea: Currency);		
Pone la línea componente indicada en el parámetro nIdLinea en edición, preparada para editar el contenido de sus campos. Al final de las modificaciones deberá llamarse a AnadirComponente.		
Parámetros:	nIdLinea	Identificador de la línea componente. Se corresponde con el campo NUMLINPED del fichero LINEPEDI.

procedure AnadirTallaColor(sCodTallaH, sCodTallaV: WideString; nUnidades:Double);		
Indica las unidades, junto con su talla y color de la línea activa.		
	sCodTallaH	Código de talla horizontal
	sCodTallaV	Código de talla vertical

function AnularUnidades(IdLin: Currency; Bultos: Double; Paquetes: Double; Unidades: Double; Fecha: TDateTime; const Motivo: WideString): Currency;
Anula las unidades, bultos, paquetes en la fecha indicada para el identificador que se indica en el parámetro IdLin.

Parámetros:	IdLin	Identificador de la línea. Se corresponde del LINEPEDI.	nUnidades	Nº unidades de la línea de talla
	Bultos	Número de bultos.		
	Paquetes	Número de paquetes.		
	Unidades	Número de unidades a anular.		
	Fecha	Fecha de la anulación.		
	Motivo	Motivo por el que se anulan las unidad		
Valor retornado:	Currency	Devuelve el valor del campo IDA3ERPA __ANULACIONLINEPEDI.		
Parámetros:				

procedure DeshacerAnularUnidades(IdLin: Currency; IdA3ERPACTIVEX: Currency);		
Deshace una anulación de unidades realizada previamente.		
Parámetros:	IdLin	Identificador de la línea. Se corresponde con el campo IDLIN del LINEPEDI.
	IdA3ERPACTIVEX	Identificador de la anulación realizada previamente. Se corresponde con el campo IDA3ERPACTIVEX de la tabla __ANULACIONLINEPEDI.

procedure CalcularImpuestosyTotales;
Realiza el cálculo de los impuestos y los totales del documento.

procedure AnadirSubdetalle(const ParametrosDetalle: ILineaDetalleParametros);		
Permite añadir líneas de detalle (lotes, num serie…) de la línea del componente de un kit		
Parámetros:	ParametrosDetalle	Clase que implementa la interfaz ILineaDetalleParametros. La que nos permite informar del detalle a añadir

procedure BorrarSubdetalle(IdLinea: Currency);		
Premite borrar la línea de detalle (lotes, num serie..) de la línea del componente de un kit		
Parámetros:	IdLinea	Identificador de la línea (campo IDLIN) para localizar el detalle y borrarlo

procedure CambiarSubdetalle(const ParametrosDetalle: ILineaDetalleParametros);
Permite cambiar la línea de detalle (lotes, num serie…) de la línea del componente de un kit

Parámetros:	ParametrosDetalle	Clase que implementa la interfaz ILineaDetalleParametros. La que nos permite informar del detalle a modificar. Si el valor de la línea es -1 cambiará el registro posicionado sin tener necesidad de buscarlo. Pensado para cuando sólo hay un detalle

function ILineasDocumento.ObtenerComponentesLinea: OleVariant
Permite obtener todas las líneas de componentes del artículo kit posicionado Devuelve una array de arrays con la siguiente estructura: A[0] => Numero líneas de componentes A[1..N][0] => Número de campos. Ejemplo 150 campos A[1..n][1..n][0] => Nombre del campo. Ejemplo 'CodArt' A[1..n][1..n][1] => Valor del campo. Ejemplo ' 1'

function ILineasDocumento.ObtenerDetalleComponentesLinea: OleVariant;
FunciónPermite obtener todas las líneas de detalle del componente del kit posicionado Devuelve una array de arrays con la siguiente estructura: A[0] => Numero líneas de detalle A[1..N][0] => Número de campos. Ejemplo 150 campos A[1..n][1..n][0] => Nombre del campo. Ejemplo 'CodArt' A[1..n][1..n][1] => Valor del campo. Ejemplo 'LOTE100'

function ILineasDocumento.ObtenerDetallesLinea: OleVariant;
Permite obtener todas las líneas de detalle del artículo padre posicionado Devuelve una array de arrays con la siguiente estructura: A[0] => Numero líneas de detalle A[1..N][0] => Número de campos. Ejemplo 150 campos A[1..n][1..n][0] => Nombre del campo. Ejemplo 'CodArt' A[1..n][1..n][1] => Valor del campo. Ejemplo 'LOTE100'

function ILineasDocumento.ObtenerLineas: OleVariant;
Permite obtener todas las líneas sin detalle/componentes del documento Devuelve una array de arrays con la siguiente estructura: A[0] => Numero líneas del documento A[1..N][0] => Número de campos. Ejemplo 150 campos A[1..n][1..n][0] => Nombre del campo. Ejemplo 'CodArt' A[1..n][1..n][1] => Valor del campo. Ejemplo 'B10

13.- OBJETO PEDIDO

Extiende Pedido140501 y añade nuevos métodos.

Interface Pedido
procedure Clonar(const DocumentoOrigenTipo: WideString; DocumentoOrigenID: Currency; BorrarLineasDelDocumentoDestino: WordBool; ClonarVariablesDeDocumentos: WordBool); [Nuevo 14.05.02]
procedure Copiar(const DocumentoOrigenTipo: WideString; DocumentoOrigenID: Currency; BorrarLineasDelDocumentoDestino: WordBool; CopiarCondicionesFinancieras: WordBool; CopiarLosPortes: WordBool; CopiarPreciosYDescuentos: WordBool; CopiarDescripcionesYObservacionesLineas: WordBool); [Nuevo 14.05.02]

procedure Clonar(const DocumentoOrigenTipo: WideString; DocumentoOrigenID: Currency; BorrarLineasDelDocumentoDestino: WordBool; ClonarVariablesDeDocumentos: WordBool);		
Mediante este procedimiento se permite clonar un documento sobre otro.		
Parámetros:		
DocumentoOrigenTipo	'OV' Oferta venta 'PV' Pedido de venta 'AV' Albarán de venta 'DV' Depósito de venta 'FV' Factura de venta	'OC' Oferta compra 'PC' Pedido de compra 'AC' Albarán de compra 'DC' Depósito de compra 'FC' Factura de compra
DocumentoOrigenID	Identificador del documento de origen	
BorrarLineasDelDocumentoDestino	Antes de realizar la operación, borra las lineas del documento destino	
ClonarVariablesDeDocumentos	Se clonarán las variables de los documentos	

procedure Copiar(const DocumentoOrigenTipo: WideString; DocumentoOrigenID: Currency; BorrarLineasDelDocumentoDestino: WordBool; CopiarCondicionesFinancieras: WordBool; CopiarLosPortes: WordBool; CopiarPreciosYDescuentos: WordBool; CopiarDescripcionesYObservacionesLineas: WordBool);		
Mediante este procedimiento se permite copiar un documento sobre otro.		
Parámetros:		
DocumentoOrigenTipo	'OV' Oferta venta 'PV' Pedido de venta 'AV' Albarán de venta 'DV' Depósito de venta 'FV' Factura de venta	'OC' Oferta compra 'PC' Pedido de compra 'AC' Albarán de compra 'DC' Depósito de compra 'FC' Factura de compra
DocumentoOrigenID	Identificador del documento de origen	
BorrarLineasDelDocumentoDestino	Antes de realizar la operación, borra las lineas del documento destino	
CopiarCondicionesFinancieras	Copiar condiciones financieras del documento origen. Se copiarán del documento origen la forma de pago, documento de pago, descuento de pronto pago o recargo financiero y si éste afecta a la base imponible.	
CopiarLosPortes	Copiar los portes al documento destino	
CopiarPreciosYDescuentos	Copiar precios y descuentos del documento origen	
CopiarDescripcionesYObservacionesLineas	Copiar descripciones y observaciones de las líneas del documento origen	

14.- OBJETO DEPÓSITO140501

Objeto que permite la creación, modificación y borrado de depósitos.

Interface Deposito
property Estado: EstadoMaestro readonly; **property** AsStringCab[const sCampo: WideString]: WideString; **property** AsFloatCab[const sCampo: WideString]: Double; **property** AsIntegerCab[const sCampo: WideString]: Integer; **property** AsBooleanCab[const sCampo: WideString]: WordBool; **property** AsCurrencyCab[const sCampo: WideString]: Currency;

```
property AsVariantCab[const sCampo: WideString]: Currency;
property AsStringLin[const sCampo: WideString]: WideString;
property AsFloatLin[const sCampo: WideString]: Double;
property AsIntegerLin[const sCampo: WideString]: Integer;
property AsBooleanLin[const sCampo: WideString]: WordBool;
property AsCurrencyLin[const sCampo: WideString]: Currency;
property AsVariantLin[const sCampo: WideString]: Currency;
property AsBooleanComp[const sCampo: WideString]: WordBool;
property AsStringComp[const sCampo: WideString]: WideString;
property AsFloatComp[const sCampo: WideString]: Double;
property AsIntegerComp[const sCampo: WideString]: Integer;
property AsCurrencyComp[const sCampo: WideString]: Currency;
property AsVariantComp[const sCampo: WideString]: Currency;
property OmitirMensajes:WordBool;
property ActivarAlarmaCab: WordBool;
property ValidarPrecios: WordBool;
property ValidarArtBloqueado: WordBool;
property AvisarStock: WordBool;
property AvisarRiesgo: WordBool;
property AvisarCentroCosteCabecera: WordBool;
procedure Iniciar;
procedure Acabar;
procedure Cancela;
procedure Nuevo(  sFecha: WideString; sCodCliPro: WideString; bEsDeCompra: WordBool);
procedure Borra(IdDocu: Currency; bEsDeCompra: WordBool);
procedure Modifica(IdDocu: Currency; bEsDeCompra: WordBool);
procedure ModificaParcial(IdDocu: Currency; NumLin: Integer, EsDeCompra: WordBool);
function Anade: Currency;
procedure NuevaLinea;
procedure AnadirLinea;
procedure CancelaLin;
procedure NuevaLineaArt(const sCodArt: WideString; nUnidades: Currency);
procedure EditarLinea(nIdLinea: Currency);
procedure BorrarLinea(nIdLinea: Currency);
procedure IniciarServir(const sIdDocu: WideString; nIdDocu: Currency; bConMensaje:
WordBool);
procedure ServirDocumento;
procedure FinServir;
procedure ServirLinea( nNumGru: Currency; nNumLin: Currency; nNumBul: Currency;
                       nNumPaq: Currency; nNumUni: Double; nPrcMed: Double;
                       sNSerie: WideString; sLote: WideString; sUbicac: WideString;
                       sFecCad: WideString);
procedure AnularLinea( nNumLin: Currency; nNumBul: Currency; nNumPaq: Currency;
                        nNumUni: Double);
```

```
procedure AnadirDetalle( nUnidades: Double; sNumSerie: WideString; sLote: WideString;
                         sUbicacion: WideString; sFecCadu: WideString);
procedure CambiarDetalle( nIdLine: Double; nUnidades: Double; sNumSerie: WideString;
                          sLote: WideString; sUbicacion: WideString;  sFecCadu: WideString);
procedure BorrarDetalle(nIdLine: Double);
procedure AnadirTalla(sCodFamTallaH, sCodFamTallaV, sCodTallaH,
    sCodTallaV: WideString; nUnidades, nPrcMoneda: Double);
procedure CambiarTalla(nIdLine, nUnidades, nPrcMoneda: Double);
procedure BorrarTalla(nIdLine: Double);
procedure AnadirTallaColor(const sCodTallaH: WideString; const sCodTallaV: WideString;
nUnidades: Double); safecall;
procedure NuevoComponente;
procedure EditarComponente(nIdLine:Currency);
procedure BorrarComponente(nIdLine:Currency);
procedure AnadirComponente;
procedure CancelarComponente;
function AnularUnidades(IdLin: Currency; Bultos: Double; Paquetes: Double; Unidades:
Double; Fecha: TDateTime; const Motivo: WideString): Currency;
procedure DeshacerAnularUnidades(IdLin: Double; IdA3ERPACTIVEX: Double);
procedure CalcularImpuestosyTotales;
function ObtenerComponentesLinea: OleVariant;
function ObtenerDetalleComponentesLinea: OleVariant;
function ObtenerDetallesLinea: OleVariant;
```

Interface ILineasDocumento (*cast* Deposito a ILineasDocumento)
function ObtenerLineas: OleVariant; **procedure AnadirSubdetalle(const ParametrosDetalle: ILineaDetalleParametros);** **procedure BorrarSubdetalle(IdLinea: Currency);** **procedure CambiarSubdetalle(const ParametrosDetalle: ILineaDetalleParametros);**

Propiedades	**Tipo**	**Descripción**
AsStringCab	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsIntegerCab	Tabla(Integer)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsFloatCab	Tabla(Float)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsBooleanCab	Tabla(Lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsCurrencyCab	Tabla(Currency)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsStringLin	Tabla(string)	Asigna/lee valores a/de los campos del registro

		nuevo/actual.
AsIntegerLin	Tabla(Integer)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsFloatLin	Tabla(Float)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsBooleanLin	Tabla(Lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsCurrencyLin	Tabla(Currency)	Asigna/lee valores a/de los campos del registro nuevo/actual.
Estado	EstadoMaestro	Devuelve el estado en que se encuentra el objeto
AsStringComp	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsIntegerComp	Tabla(Integer)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsFloatComp	Tabla(Float)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsBooleanComp	Tabla(Lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsCurrencyComp	Tabla(Currency)	Asigna/lee valores a/de los campos del registro nuevo/actual.
OmitirMensajes	Lógico	Asignar valor para ocultar mensajes de información
ActivarAlarmaCab	Lógico	Asignar valor para activar o no alarmas en documentos
ActivarAlarmaLin	Lógico	Asignar valor para activar o no alarmas en documentos
ValidarPrecios	Lógico	Asignar valor para activar o no los mensajes de validación de precios
ValidarArtBloqueado	Lógico	Asignar valor para activar o no los mensajes de los artículos bloqueados
AvisarStock	Lógico	Asignar valor para activar o no los mensajes de aviso de stock
AvisarRiesgo	Lógico	Asignar valor para activar o no los mensajes de aviso de riesgo
AvisarCentroCosteCabecera	Lógico	Asignar valor para activar o no el mensaje que aparece al cambiar algún nivel de analítica y pregunta si se desa cambiar en las líneas.
CalcularImpuestosyTotales	Lógico	Realiza el cálculo de los impuestos y los totales del documento.

*EstadoMaestro: Ver objeto Maestro.

Método	Tipo	Descripción
Iniciar	Procedimiento	Inicia el objeto
Acabar	Procedimiento	Cierra el objeto
Nuevo	Procedimiento	Inicia un nuevo documento
Borra	Procedimiento	Borra el documento
Modifica	Procedimiento	Prepara el documento para poderlo modificar.
ModificaParcial	Procedimiento	Prepara el documento para poder cambiar solamente el detalle de una línea
Cancela	Procedimiento	Cancela la edición del documento.
Anade	Función	Añade los datos editados del documento a la base de datos y devuelve el identificador con el que se almacenará.
NuevaLinea	Procedimiento	Inicia una nueva línea
NuevaLineaArt	Procedimiento	Inicia una nueva línea aplicando las políticas de nexus, dados el código del artículo y las unidades pasadas como parámetro.
AnadirLinea	Procedimiento	Añade la línea
CancelaLin	Procedimiento	Cancela la línea en edición
EditarLinea	Procedimiento	Permite editar una línea ya existente del documento.
BorrarLinea	Procedimiento	Permite borrar una línea del documento.
IniciarServir	Procedimiento	Inicia el proceso de servir líneas de otro documento origen.
ServirDocumento	Procedimiento	Sirve el documento origen entero.
ServirLinea	Procedimiento	Sirve una línea del documento origen.
AnularLinea	Procedimiento	Anula una línea del documento origen conforme no podrán servirse las unidades indicadas.
FinServir	Procedimiento	Finaliza el proceso de servir.
AnadirDetalle	Procedimiento	Identifica los nº de serie, lotes, fechas de caducidad y/o ubicaciones de una línea del documento.
CambiarDetalle	Procedimiento	Cambia los nº de serie, lotes, fechas de caducidad y/o ubicaciones de una línea del documento.
BorrarDetalle	Procedimiento	Borra el detalle de los nºs de serie, lotes, fechas de caducidad y/o ubicaciones de una línea del documento.
AnadirTalla	Procedimiento	Identifica las tallas de una línea del documento.
CambiarTalla	Procedimiento	Cambia las tallas de una línea del documento.
BorrarTalla	Procedimiento	Borra el detalle de las tallas de una línea del documento.

NuevoComponente	Procedimiento	Inicia una nueva línea componente de un kit.
EditarComponente	Procedimiento	Permite editar una línea componente de un kit.
BorrarComponente	Procedimiento	Borra una línea componente de un kit.
AnadirComponente	Procedimiento	Añade una línea componente de un kit.
CancelarComponente	Procedimiento	Cancela una línea componente de un kit.
AnadirSubdetalle	Procedimiento	Permite añadir líneas de detalle (lotes, num serie…) de la línea del componente de un kit
CambiarSubdetalle	Procedimiento	Permite cambiar la línea de detalle (lotes, num serie…) de la línea del componente de un kit
BorrarSubdetalle	Procedimiento	Premite borrar la línea de detalle (lotes, num serie..) de la línea del componente de un kit
Funciones que se obtienen al convertir (*cast*) a IL**ineasDocumento**		
ObtenerComponentesLinea	Función	Permite obtener todas las líneas de componentes del artículo kit posicionado
ObtenerDetalleComponentesLinea	Función	Permite obtener todas las líneas de detalle del componente del kit posicionado
ObtenerDetallesLinea	Función	Permite obtener todas las líneas de detalle del artículo padre posicionado
ObtenerLineas	Función	Permite obtener todas las líneas sin detalle/componentes del documento

Las primeras funciones de la lista que se presentan a continuación se utilizan para asignar / leer valores de la cabecera o de las líneas del documento.

Deben tenerse en cuenta algunas normas importantes:

- Si el nombre del campo no existe se producirá un error.
- Si el valor asignado al campo especificado no es correcto, no se producirá un error. Este se producirá al intentar guardar el documento o la línea.

Hay que tener en cuenta lo siguiente:

- Los campos fecha se han de asignar como cadenas de texto como ‘01/01/2003’
- Uso en Delphi: a3ERPACTIVEXDocumento.AsStringCab[‘CodCli’]
- Uso en Visual Basic: a3ERPACTIVEXDocumento. AsStringCab (‘CodCli’)

property AsStringCab[sCampo:String]: String		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerCab[sCampo:String]: Integer		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatCab[sCampo:String]: Double		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanCab[sCampo:String]: Boolean		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es lógico.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsCurrencyCab[sCampo:String]: Currency		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es currency.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsVariantCab[sCampo:String]: Currency		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es variant.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsStringLin[sCampo:String]: String		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerLin[sCampo:String]: Integer		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatLin[sCampo:String]: Double		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanLin[sCampo:String]: Boolean		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es lógico.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsCurrencyLin[sCampo:String]: Currency		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es currency.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsVariantLin[sCampo:String]: Currency		
A través de esta propiedad podemos asignar valor a los campos de la linea especificando que su tipo es variant.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsStringComp[sCampo:String]: String		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo

Valor retornado	String	Valor del campo

property AsIntegerComp[sCampo:String]: Integer		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatComp[sCampo:String]: Double		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanComp[sCampo:String]: Boolean		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es lógico.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsCurrencyComp[sCampo:String]: Currency		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es currency.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsVariantComp[sCampo:String]: Currency		
A través de esta propiedad podemos asignar valor a los campos de la linea especificando que su tipo es variant.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property ValidarPrecios: WordBool		
A través de esta propiedad podemos asignar valor si queremos mostrar o ocultar los avisos de precios por debajo del precio mínimo		
Valor retornado	Lógico	Valor del campo

property ValidarArtBloqueado: WordBool		
A través de esta propiedad podemos asignar valor si queremos mostrar o ocultar los avisos de artículo bloqueado		
Valor retornado	Lógico	Valor del campo

property AvisarStock: WordBool		
A través de esta propiedad podemos asignar valor si queremos mostrar o ocultar los avisos del stock		
Valor retornado	Lógico	Valor del campo

property AvisarRiesgo: WordBool		
A través de esta propiedad podemos asignar valor si queremos mostrar o ocultar los avisos de riesgo		
Valor retornado	Lógico	Valor del campo

property AvisarCentroCosteCabecera: WordBool		
A través de esta propiedad podemos asignar valor si queremos mostrar o ocultar el aviso para cambiar el centro de coste en las líneas cuando se cambia desde la cabecera		
Valor retornado	Lógico	Valor del campo

A continuación, se presentan las propiedades y métodos necesarios para **crear, borrar y modificar depósitos.**

property Estado:EstadoMaestro		
Esta propiedad nos indica el estado del objeto. Posibles estados: ESTM_NOACTIVO: Documento cerrado. No se puede usar. ESTM_ACTIVO: Documento activo. Podemos añadir, borrar, modificar. ESTM_NUEVO: Documento en estado de inserción de un nuevo registro. ESTM_EDICION: Documento en estado modificación, podemos asignar valores a los campos.		
Valor retornado	EstadoMaestro	Retorna el estado del documento.

procedure Iniciar;
Este procedimiento es el que reserva los recursos necesarios para que el documento pueda ser usado. Al iniciar el documento el estado cambia de ESTM_NOACTIVO a ESTM_ACTIVO.

procedure Acabar;
Este procedimiento es el que libera los recursos utilizados por el objeto. Al acabar el documento el estado cambia de ESTM_ACTIVO a ESTM_NOACTIVO.

procedure Nuevo(sFecha: String; sCodCliPro: String; bEsDeCompra: Boolean);		
Inserta un nuevo registro y los prepara para ser editado. Pasa de estado ESTM_ACTIVO a ESTM_NUEVO.		
Parámetros:	sFecha	Fecha del documento
	sCodCliPro	Código del cliente o proveedor
	bEsDeCompra	True → Compra False → Venta

procedure Modifica(nIdDocu: Currency; bEsDeCompra: WordBool);		
Pone el documento que tiene el identificador indicado en edición.		
Parámetros:	nIdDocu	Identificador del documento
	bEsDeCompra	True → Compra False → Venta

procedure ModificaParcial(IdDocu: Currency; NumLin: Integer; EsDeCompra: WordBool);		
Este método se utiliza únicamente para cambiar el detalle de una línea de un documento. Se puede utilizar en documentos con muchas líneas, mejorando la velocidad, ya que solo carga el detalle de la línea que se indica. Despues de llamar a este método solo se pueden utilizar los métodos AnadirDetalle, CambiarDetalle, BorrarDetalle, AnadirLinea, CancelaLin, Anade y Cancela. Pone el documento que tiene el identificador indicado en edición y la línea que tiene el Numlin indicado en edición		
Parámetros:	IdDocu	Identificador del documento
	NumLin	Numero de línea
	EsDeCompra	True → Compra False → Venta

Procedure Anade;
Almacena los valores de los campos en la base de dato con los cambios introducidos por el programador. Genera las repercusiones del documento como pueden ser: actualización de stocks y actualización de estadísticas. Pasa de los estados ESTM_NUEVO o EST_EDICION a ESTM_ACTIVO.

Procedure Cancela;
Cancela la edición del documento. Pasa de los estados ESTM_NUEVO o EST_EDICION a ESTM_ACTIVO.

procedure Borra(nIdDocu: Currency; bEsDeCompra: Boolean);		
Elimina el documento con el identificador indicado eliminando también sus repercusiones en stocks, contabilidad, cartera, estadísticas e IVA.		
Parámetros:	nIdDocu	Identificador del documento
	bEsDeCompra	True → Compra False → Venta

Procedure NuevaLinea;
Crea una línea con los valores por defecto. A partir de ese momento podremos editar los valores de la línea.

Procedure NuevaLineaArt(sCodArt: string; nUnidades: Currency);
Crea una línea con el artículo y las unidades indicadas realizando las búsquedas necesarias de los valores correspondientes a cuentas, precios y descuentos, etc... A partir de ese momento podremos editar los valores de la línea.

Parámetros:	sCodArt	Código del artículo
	nUnidades	Unidades

Procedure AnadirLinea;
Almacena los valores de los campos de la línea en la base de datos.

Procedure CancelaLin;
Cancela la edición de la línea activa.

procedure EditarLinea(nIdLinea: Currency);		
Pone la línea indicada en el parámetro nIdLinea en edición, preparada para editar el contenido de sus campos. Al final de las modificaciones deberá llamarse a AnadirLinea.		
Parámetros:	nIdLinea	Identificador de la línea.. Se corresponde con el campo NUMLINDEP del fichero LINEDEPO. Para localizarlo debe realizarse una consulta de este campo (directamente a la base de datos) en los registros con IdDepV = a3ERPACTIVEXDeposito.AsStringCab['IdDepV'] en ventas o IdDepC = a3ERPACTIVEXDeposito.AsStringCab['IdDepC'] en compras.

procedure BorrarLinea(nIdLinea: Currency);		
Borra la línea con el identificador indicado en el parámetro nIdLinea.		
Parámetros:	nIdLinea	Identificador de la línea. Se corresponde con el campo NUMLINDEP del fichero LINEDEPO. Para localizarlo debe realizarse una consulta de este campo (directamente a la base de datos) en los registros con IdDepV = a3ERPACTIVEXDeposito.AsStringCab['IdDepV'] en ventas o IdDepC = a3ERPACTIVEXDeposito.AsStringCab['IdDepC'] en compras.

En la siguiente colección de procedimientos se encapsula el funcionamiento de las utilidades para **servir cualquier documento sobre los depósitos**. En primer lugar, debe llamarse al procedimiento IniciarServir. A continuación, pueden utilizarse los procedimientos ServirDocumento, ServirLinea o AnularLinea y funalmente debe llamarse a FinServir.

procedure IniciarServir(sIdDocu: string; nIdDocu: Currency; bConMensaje: Boolean);		
Prepara el documento indicado para servirse. Reserva los recursos necesarios.		
Parámetros:	sIdDocu	'O' → Oferta 'P'->Pedido
	nIdDocu	Representa el identificador de la oferta (IdOfeV, IdOfeC) o del pedido(IdPedV o IdPedC).

	bConMensaje	En caso en que se produzcan diferencias entre las condiciones del documento origen y el destino (Por ejemplo, la forma de pago, el documento de pago, el transportista, etc....), el programa puede actuar de dos formas distintas: - bConMesnaje = True: Preguntar al usuario si se desean aplicar las condiciones del documento destino o respetar las del documento origen. - bConMensaje = False: Aplicar las condiciones del documento destino.

procedure ServirDocumento;
Sirve todo el material pendiente de todo del documento origen indicado en la función IniciarServir.

procedure ServirLinea(nNumGru, nNumLin: Currency; nNumBul: Currency; nNumPaq: Currency; nNumUni: Double; sNSerie: WideString; sLote: WideString; sUbicac: WideString; sFecCad: WideString);		
Sirve la línea con los parámetros indicados.		
Parámetros:	nNumGru	Obsoleto, poner un cero.
	nNumLin	Nº de línea. NUMLINOFE de ofertas y NUMLINPED en pedidos
	nNumBul	Nº de bultos de la línea que desean servirse.
	nNumPaq	Nº de paquetes que desean servirse.
	nNumUni	Nº de unidades que desean servirse.
	sNSerie	Nº de serie.
	sLote	Lote.
	sUbicac	Ubicación
	sFecCad	Fecha de caducidad.

procedure AnularLinea(nNumLin: Currency; nNumBul: Currency; nNumPaq: Currency; nNumUni: Double);		
Anula las unidades indicadas de la línea pasada por parámetro. Esas unidades no estarán disponibles para ser servidas posteriormente.		
Parámetros:	nNumLin	Nº de línea del documento origen (NUMLINOFE o NUMLINPED).
	nNumBul	Nº de bultos de la línea que desean servirse.
	nNumPaq	Nº de paquetes que desean servirse.
	nNumUni	Nº de unidades que desean servirse.

procedure FinServir;
Realiza las operaciones realizadas desde que se llamó a IniciarServir de forma definitiva. Libera los recursos necesarios.

Los procedimientos que se describen a continuación se utilizan para **especificar los lotes, nº de serie, fechas de caducidad y ubicaciones de una línea** a no ser que esta selección se realice de forma automática. Para consultar si ya se han detallado todos los lotes, nº de serie, fechas de caducidad y ubicaciones necesarias se debe llamar a la función ConsultarDetalle;

procedure AnadirDetalle(nUnidades: Double; sNumSerie: WideString; sLote: WideString; sUbicacion: WideString; sFecCadu: WideString);		
Indica las unidades necesarias y su nº serie, lote, fecha de caducidad y ubicación de la línea activa. Para conocer los nº serie, lote, fecha de caducidad y ubicaciones disponibles en stock hay que llamar a la función **Varios.DetalleStockDisponible**..		
Parámetros:	nNunidades	Nº de unidades de la línea de detalle.
	sNumSerie	Nº de serie.
	sLote	Lote.
	sUbicacion	Ubicación.
	sFecCadu	Fecha de caducidad.

procedure CambiarDetalle(nIdLine: Double; sNumSerie: WideString; sLote: WideString; sUbicacion: WideString; sFecCadu: WideString);		
Indica las unidades necesarias y su Nº serie, lote, fecha de caducidad y ubicación de la línea activa. Para conocer los nº serie, lote, fecha de caducidad y ubicaciones disponibles en stock hay que llamar a la función **Varios.DetalleStockDisponible**. Para consultar el detalle de la línea activa hay que llamar a la función **ConsultarDetalle**.		
Parámetros:	nIdLine	Identificador de la línea. Se corresponde con el campo IDLIN de la tabla LINEDEPO.
	nUnidades	Nº de unidades de la línea de detalle.
	sNumSerie	Nº de serie.
	sLote	Lote.
	sUbicacion	Ubicación.
	sFecCadu	Fecha de caducidad.

procedure BorrarDetalle(nIdLine: Double);		
Elimina la línea de detalle indicada en el parámetro.		
Parámetros:	nIdLine	Identificador de la línea. Se corresponde con el campo IDLIN de la tabla LINEDEPO.

procedure AnadirTalla(sCodFamTallaH, sCodFamTallaV, sCodTallaH, sCodTallaV: WideString; nUnidades, nPrcMoneda:Double);		
Indica las unidades y el precio necesarios, junto con sus tallas y colores de la línea activa.		
Parámetros:	sCodFamTallaH	Familia de talla horizontal
	sCodFamTallaV	Familia de talla vertical
	sCodTallaH	Código de talla horizontal
	sCodTallaV	Código de talla vertical
	nUnidades	Nº unidades de la línea de talla
	nPrcMoneda	Precio de la línea de talla.

procedure CambiarTalla(nIdLine, nUnidades, nPrcMoneda: Double);		
Indica las unidades y el precio de la línea activa.		
Parámetros:	nIdLine	Identificador de la línea. Se corresponde con el campo IDLIN de la tabla LINEDEPO.
	nUnidades	Nº de unidades de la línea de talla.
	nPrcMoneda	Precio de la línea de talla.

procedure BorrarTalla(nIdLine: Double);		
Elimina la línea de talla indicada en el parámetro.		
Parámetros:	nIdLine	Identificador de la línea. Se corresponde con el campo IDLIN de la tabla LINEDEPO.

procedure AnadirTallaColor(sCodTallaH, sCodTallaV: WideString; nUnidades:Double);		
Indica las unidades, junto con su talla y color de la línea activa.		
Parámetros:	sCodTallaH	Código de talla horizontal
	sCodTallaV	Código de talla vertical
	nUnidades	Nº unidades de la línea de talla

Procedure NuevoComponente;
Crea una línea con los valores por defecto. A partir de ese momento podremos editar los valores de la línea componente

Procedure AnadirComponente;
Almacena los valores de los campos de la línea componente en la base de datos.

Procedure CancelaComponente;
Cancela la edición de la línea componente activa.

procedure EditarComponente(nIdLinea: Currency);		
Pone la línea componente indicada en el parámetro nIdLinea en edición, preparada para editar el contenido de sus campos. Al final de las modificaciones deberá llamarse a AnadirComponente.		
Parámetros:	nIdLinea	Identificador de la línea componente. Se corresponde con el campo NUMLINDEP del fichero LINEDEPO.

procedure BorrarComponente(nIdLinea: Currency);		
Borra la línea componente con el identificador indicado en el parámetro nIdLinea.		
Parámetros:	nIdLinea	Identificador de la línea componente. Se corresponde con el campo NUMLINDEP del fichero LINEDEPO.

function AnularUnidades(IdLin: Currency; Bultos: Double; Paquetes: Double; Unidades: Double; Fecha: TDateTime; const Motivo: WideString): Currency;		
Anula las unidades, bultos, paquetes en la fecha indicada para el identificador que se indica en el parámetro IdLin.		
Parámetros:	IdLin	Identificador de la línea. Se corresponde con el campo IDLIN del __LINEDEPO.
	Bultos	Número de bultos.
	Paquetes	Número de paquetes.
	Unidades	Número de unidades a anular.
	Fecha	Fecha de la anulación.
	Motivo	Motivo por el que se anulan las unidades.
Valor retornado:	Currency	Devuelve el valor del campo IDA3ERPACTIVEX de la tabla __ANULACION__LINEDEPO.

procedure DeshacerAnularUnidades(IdLin: Currency; IdA3ERPACTIVEX: Currency);		
Deshace una anulación de unidades realizada previamente.		
Parámetros:	IdLin	Identificador de la línea. Se corresponde con el campo IDLIN del __LINEDEPO.
	IdA3ERPACTIVEX	Identificador de la anulación realizada previamente. Se corresponde con el campo IDA3ERPACTIVEX de la tabla __ANULACION__LINEDEPO.

procedure CalcularImpuestosyTotales;
Realiza el cálculo de los impuestos y los totales del documento.

procedure AnadirSubdetalle(const ParametrosDetalle: ILineaDetalleParametros);		
Permite añadir líneas de detalle (lotes, num serie…) de la línea del componente de un kit		
Parámetros:	ParametrosDetalle	Clase que implementa la interfaz ILineaDetalleParametros. La que nos permite informar del detalle a añadir

procedure BorrarSubdetalle(IdLinea: Currency);		
Premite borrar la línea de detalle (lotes, num serie..) de la línea del componente de un kit		
Parámetros:	IdLinea	Identificador de la línea (campo IDLIN) para localizar el detalle y borrarlo

procedure CambiarSubdetalle(const ParametrosDetalle: ILineaDetalleParametros);		
Permite cambiar la línea de detalle (lotes, num serie…) de la línea del componente de un kit		
Parámetros:	ParametrosDetalle	Clase que implementa la interfaz ILineaDetalleParametros. La que nos permite informar del detalle a modificar. Si el valor de la línea es -1 cambiará el registro posicionado sin tener necesidad de buscarlo. Pensado para cuando sólo hay un detalle

function ILineasDocumento.ObtenerComponentesLinea: OleVariant
Permite obtener todas las líneas de componentes del artículo kit posicionado Devuelve una array de arrays con la siguiente estructura: A[0] => Numero líneas de componentes A[1..N][0] => Número de campos. Ejemplo 150 campos A[1..n][1..n][0] => Nombre del campo. Ejemplo 'CodArt' A[1..n][1..n][1] => Valor del campo. Ejemplo ' 1'

function ILineasDocumento.ObtenerDetalleComponentesLinea: OleVariant;
FunciónPermite obtener todas las líneas de detalle del componente del kit posicionado Devuelve una array de arrays con la siguiente estructura: A[0] => Numero líneas de detalle A[1..N][0] => Número de campos. Ejemplo 150 campos A[1..n][1..n][0] => Nombre del campo. Ejemplo 'CodArt' A[1..n][1..n][1] => Valor del campo. Ejemplo 'LOTE100'

function ILineasDocumento.ObtenerDetallesLinea: OleVariant;
Permite obtener todas las líneas de detalle del artículo padre posicionado Devuelve una array de arrays con la siguiente estructura: A[0] => Numero líneas de detalle A[1..N][0] => Número de campos. Ejemplo 150 campos A[1..n][1..n][0] => Nombre del campo. Ejemplo 'CodArt' A[1..n][1..n][1] => Valor del campo. Ejemplo 'LOTE100'

function ILineasDocumento.ObtenerLineas: OleVariant;
Permite obtener todas las líneas sin detalle/componentes del documento Devuelve una array de arrays con la siguiente estructura: A[0] => Numero líneas del documento A[1..N][0] => Número de campos. Ejemplo 150 campos A[1..n][1..n][0] => Nombre del campo. Ejemplo 'CodArt' A[1..n][1..n][1] => Valor del campo. Ejemplo 'B10

15.- OBJETO DEPÓSITO

Extiende Deposito140501 y añade nuevos métodos.

Interface Deposito
procedure Clonar(const DocumentoOrigenTipo: WideString; DocumentoOrigenID: Currency; BorrarLineasDelDocumentoDestino: WordBool; ClonarVariablesDeDocumentos: WordBool); [Nuevo 14.05.02] procedure Copiar(const DocumentoOrigenTipo: WideString; DocumentoOrigenID: Currency; BorrarLineasDelDocumentoDestino: WordBool; CopiarCondicionesFinancieras: WordBool; CopiarLosPortes: WordBool; CopiarPreciosYDescuentos: WordBool; CopiarDescripcionesYObservacionesLineas: WordBool); [Nuevo 14.05.02]

procedure Clonar(const DocumentoOrigenTipo: WideString; DocumentoOrigenID: Currency; BorrarLineasDelDocumentoDestino: WordBool; ClonarVariablesDeDocumentos: WordBool);		
Mediante este procedimiento se permite clonar un documento sobre otro.		
Parámetros:		
DocumentoOrigenTipo	'OV' Oferta venta 'PV' Pedido de venta 'AV' Albarán de venta 'DV' Depósito de venta 'FV' Factura de venta	'OC' Oferta compra 'PC' Pedido de compra 'AC' Albarán de compra 'DC' Depósito de compra 'FC' Factura de compra
DocumentoOrigenID	Identificador del documento de origen	
BorrarLineasDelDocumentoDestino	Antes de realizar la operación, borra las lineas del documento destino	
ClonarVariablesDeDocumentos	Se clonarán las variables de los documentos	

procedure Copiar(const DocumentoOrigenTipo: WideString; DocumentoOrigenID: Currency; BorrarLineasDelDocumentoDestino: WordBool; CopiarCondicionesFinancieras: WordBool; CopiarLosPortes: WordBool; CopiarPreciosYDescuentos: WordBool; CopiarDescripcionesYObservacionesLineas: WordBool);		
Mediante este procedimiento se permite copiar un documento sobre otro.		
Parámetros:		
DocumentoOrigenTipo	'OV' Oferta venta 'PV' Pedido de venta 'AV' Albarán de venta 'DV' Depósito de venta 'FV' Factura de venta	'OC' Oferta compra 'PC' Pedido de compra 'AC' Albarán de compra 'DC' Depósito de compra 'FC' Factura de compra
DocumentoOrigenID	Identificador del documento de origen	
BorrarLineasDelDocumentoDestino	Antes de realizar la operación, borra las lineas del documento destino	
CopiarCondicionesFinancieras	Copiar condiciones financieras del documento origen. Se copiarán del documento origen la forma de pago, documento de pago, descuento de pronto pago o recargo financiero y si éste afecta a la base imponible.	

CopiarLosPortes	Copiar los portes al documento destino
CopiarPreciosYDescuentos	Copiar precios y descuentos del documento origen
CopiarDescripcionesYObservacionesLineas	Copiar descripciones y observaciones de las líneas del documento origen

16.- OBJETO OFERTA140501

Objeto que permite la creación, modificación y borrado de ofertas.

Interface Oferta
property Estado: EstadoMaestro readonly; **property** AsStringCab[const sCampo: WideString]: WideString; **property** AsFloatCab[const sCampo: WideString]: Double; **property** AsIntegerCab[const sCampo: WideString]: Integer; **property** AsBooleanCab[const sCampo: WideString]: WordBool; **property** AsCurrencyCab[const sCampo: WideString]: Currency; **property** AsVariantCab[const sCampo: WideString]: Currency; **property** AsStringLin[const sCampo: WideString]: WideString; **property** AsFloatLin[const sCampo: WideString]: Double; **property** AsIntegerLin[const sCampo: WideString]: Integer; **property** AsBooleanLin[const sCampo: WideString]: WordBool; **property** AsCurrencyLin[const sCampo: WideString]: Currency; **property** AsVariantLin[const sCampo: WideString]: Currency; **property** AsBooleanComp[const sCampo: WideString]: WordBool; **property** AsStringComp[const sCampo: WideString]: WideString; **property** AsFloatComp[const sCampo: WideString]: Double; **property** AsIntegerComp[const sCampo: WideString]: Integer; **property** AsCurrencyComp[const sCampo: WideString]: Currency; **property** AsVariantComp[const sCampo: WideString]: Currency; **property** AsBooleanDes[const sCampo: WideString]: WordBool; [SOLO LECTURA] **property** AsStringDes[const sCampo: WideString]: WideString; [SOLO LECTURA] **property** AsFloatDes[const sCampo: WideString]: Double; [SOLO LECTURA] **property** AsIntegerDes[const sCampo: WideString]: Integer; [SOLO LECTURA] **property** AsCurrencyDes[const sCampo: WideString]: Currency; [SOLO LECTURA] **property** AsVariantDes[const sCampo: WideString]: OleVariant; [SOLO LECTURA] **property** OmitirMensajes:WordBool; **property** ActivarAlarmaCab : WordBool; **property** ActivarAlarmaLin: WordBool; **property** ValidarPrecios: WordBool; **property** ValidarArtBloqueado: WordBool; **property** AvisarCentroCosteCabecera: WordBool;

```
procedure Iniciar;
procedure Acabar;
procedure Cancela;
procedure Nuevo(  sFecha: WideString; sCodCliPro: WideString; bEsDeCompra: WordBool);
procedure Borra(IdDocu: Currency; bEsDeCompra: WordBool);
procedure Modifica(IdDocu: Currency; bEsDeCompra: WordBool);
procedure ModificaParcial(IdDocu: Currency; NumLin: Integer, EsDeCompra: WordBool);
function Anade: Currency;
procedure NuevaLinea;
procedure AnadirLinea;
procedure CancelaLin;
procedure NuevaLineaArt(const sCodArt: WideString; nUnidades: Currency);
procedure EditarLinea(nIdLinea: Currency);
procedure BorrarLinea(nIdLinea: Currency);
procedure AnadirDetalle( nUnidades: Double; sNumSerie: WideString; sLote: WideString;
                         sUbicacion: WideString; sFecCadu: WideString);
procedure CambiarDetalle( nIdLine: Double; nUnidades: Double; sNumSerie: WideString;
                          sLote: WideString; sUbicacion: WideString;  sFecCadu:
WideString);
procedure BorrarDetalle(nIdLine: Double);
procedure AnadirTalla(sCodFamTallaH, sCodFamTallaV, sCodTallaH,
    sCodTallaV: WideString; nUnidades, nPrcMoneda: Double);
procedure CambiarTalla(nIdLine, nUnidades, nPrcMoneda: Double);
procedure BorrarTalla(nIdLine: Double);
procedure AnadirTallaColor(const sCodTallaH: WideString; const sCodTallaV: WideString;
                  nUnidades: Double);
procedure NuevoComponente;
procedure EditarComponente(nIdLine:Currency);
procedure BorrarComponente(nIdLine:Currency);
procedure AnadirComponente;
procedure CancelarComponente;
function AnularUnidades(IdLin: Currency; Bultos: Double; Paquetes: Double; Unidades:
Double;
                Fecha: TDateTime; const Motivo: WideString): Currency;
procedure DeshacerAnularUnidades(IdLin: Currency; IdA3ERPACTIVEX: Currency);
procedure CalcularImpuestosyTotales;
function ObtenerComponentesLinea: OleVariant;
function ObtenerDetalleComponentesLinea: OleVariant;
function ObtenerDetallesLinea: OleVariant;
```

Interface ILineasDocumento (*cast* Oferta a ILineasDocumento)
function ObtenerLineas: OleVariant; **procedure AnadirSubdetalle(const ParametrosDetalle: ILineaDetalleParametros);** **procedure BorrarSubdetalle(IdLinea: Currency);** **procedure CambiarSubdetalle(const ParametrosDetalle: ILineaDetalleParametros);**

Propiedades	Tipo	Descripción
AsStringCab	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsIntegerCab	Tabla(Integer)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsFloatCab	Tabla(Float)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsBooleanCab	Tabla(Lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsCurrencyCab	Tabla(Currency)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsStringLin	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsIntegerLin	Tabla(Integer)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsFloatLin	Tabla(Float)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsBooleanLin	Tabla(Lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsCurrencyLin	Tabla(Currency)	Asigna/lee valores a/de los campos del registro nuevo/actual.
Estado	EstadoMaestro	Devuelve el estado en que se encuentra el objeto
AsStringComp	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsIntegerComp	Tabla(Integer)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsFloatComp	Tabla(Float)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsBooleanComp	Tabla(Lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsCurrencyComp	Tabla(Currency)	Asigna/lee valores a/de los campos del registro nuevo/actual.

AsStringDes	Tabla(string)	Lee valores de los campos del registro nuevo/actual.
AsIntegerDes	Tabla(Integer)	Lee valores de los campos del registro nuevo/actual.
AsFloatDes	Tabla(Float)	Lee valores de los campos del registro nuevo/actual.
AsBooleanDes	Tabla(Lógico)	Lee valores de los campos del registro nuevo/actual.
AsCurrencyDes	Tabla(Currency)	Lee valores de los campos del registro nuevo/actual.
AsVariantDes	Tabla(Variant)	Lee valores de los campos del registro nuevo/actual.
OmitirMensajes	Lógico	Asignar valor para ocultar mensajes de información
ActivarAlarmaCab	Lógico	Asignar valor para activar o no alarmas en documentos
ActivarAlarmaLin	Lógico	Asignar valor para activar o no alarmas en documentos
ValidarPrecios	Lógico	Asignar valor para activar o no los mensajes de validación de precios
ValidarArtBloqueado	Lógico	Asignar valor para activar o no los mensajes de los artículos bloqueados
AvisarCentroCosteCabecera	Lógico	Asignar valor para activar o no el mensaje que aparece al cambiar algún nivel de analítica y pregunta si se desa cambiar en las líneas.
CalcularImpuestosyTotales	Lógico	Realiza el cálculo de los impuestos y los totales del documento.

*EstadoMaestro: Ver objeto Maestro.

Método	**Tipo**	**Descripción**
Iniciar	Procedimiento	Inicia el objeto
Acabar	Procedimiento	Cierra el objeto
Nuevo	Procedimiento	Inicia un nuevo documento
Borra	Procedimiento	Borra el documento
Modifica	Procedimiento	Prepara el documento para poderlo modificar.
ModificaParcial	Procedimiento	Prepara el documento para poder cambiar solamente el detalle de una línea
Cancela	Procedimiento	Cancela la edición del documento.
Anade	Función	Añade los datos editados del documento a la base de datos y devuelve el identificador con el que se almacenará.
NuevaLinea	Procedimiento	Inicia una nueva línea
NuevaLineaArt	Procedimiento	Inicia una nueva línea aplicando las políticas de

		nexus, dados el código del artículo y las unidades pasadas como parámetro.
AnadirLinea	Procedimiento	Añade la línea
CancelaLin	Procedimiento	Cancela la línea en edición
EditarLinea	Procedimiento	Permite editar una línea ya existente del documento.
BorrarLinea	Procedimiento	Permite borrar una línea del documento.
AnadirDetalle	Procedimiento	Identifica los nº de serie, lotes, fechas de caducidad y/o ubicaciones de una línea del documento.
CambiarDetalle	Procedimiento	Cambia los nº de serie, lotes, fechas de caducidad y/o ubicaciones de una línea del documento.
BorrarDetalle	Procedimiento	Borra el detalle los nº de serie, lotes, fechas de caducidad y/o ubicaciones de una línea del documento.
AnadirTalla	Procedimiento	Identifica las tallas de una línea del documento.
CambiarTalla	Procedimiento	Cambia las tallas de una línea del documento.
BorrarTalla	Procedimiento	Borra el detalle de las tallas de una línea del documento.
NuevoComponente	Procedimiento	Inicia una nueva línea componente de un kit.
EditarComponente	Procedimiento	Permite editar una línea componente de un kit.
BorrarComponente	Procedimiento	Borra una línea componente de un kit.
AnadirComponente	Procedimiento	Añade una línea componente de un kit.
CancelarComponente	Procedimiento	Cancela una línea componente de un kit.
AnadirSubdetalle	Procedimiento	Permite añadir líneas de detalle (lotes, num serie…) de la línea del componente de un kit
CambiarSubdetalle	Procedimiento	Permite cambiar la línea de detalle (lotes, num serie…) de la línea del componente de un kit
BorrarSubdetalle	Procedimiento	Premite borrar la línea de detalle (lotes, num serie..) de la línea del componente de un kit
Funciones que se obtienen al convertir (*cast*) a IL**ineasDocumento**		
ObtenerComponentesLinea	Función	Permite obtener todas las líneas de componentes del artículo kit posicionado
ObtenerDetalleComponentesLinea	Función	Permite obtener todas las líneas de detalle del componente del kit posicionado
ObtenerDetallesLinea	Función	Permite obtener todas las líneas de detalle del artículo padre posicionado
ObtenerLineas	Función	Permite obtener todas las líneas sin detalle/componentes del documento

Las primeras funciones de la lista que se presentan a continuación se utilizan para **asignar / leer valores de la cabecera o de las líneas del documento.**

Deben tenerse en cuenta algunas normas importantes:

- Si el nombre del campo no existe se producirá un error.
- Si el valor asignado al campo especificado no es correcto, no se producirá un error. Este se producirá al intentar guardar el documento o la línea.

Hay que tener en cuenta lo siguiente:

- Los campos fecha se han de asignar como cadenas de texto como '01/01/2003'
- Uso en Delphi: a3ERPACTIVEXDocumento.AsStringCab['CodCli']
- Uso en Visual Basic: a3ERPACTIVEXDocumento. AsStringCab ('CodCli')

property AsStringCab[sCampo:String]: String		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerCab[sCampo:String]: Integer		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatCab[sCampo:String]: Double		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanCab[sCampo:String]: Boolean		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es lógico.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsCurrencyCab[sCampo:String]: Currency		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es currency.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsVariantCab[sCampo:String]: Currency		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es variant.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsStringLin[sCampo:String]: String		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerLin[sCampo:String]: Integer		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatLin[sCampo:String]: Double		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanLin[sCampo:String]: Boolean		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es lógico.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsCurrencyLin[sCampo:String]: Currency		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es currency.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsVariantLin[sCampo:String]: Currency		
A través de esta propiedad podemos asignar valor a los campos de la linea especificando que su tipo es variant.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsStringComp[sCampo:String]: String		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerComp[sCampo:String]: Integer		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatComp[sCampo:String]: Double		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanComp[sCampo:String]: Boolean		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es lógico.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsCurrencyComp[sCampo:String]: Currency		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es currency.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsVariantComp[sCampo:String]: Currency		
A través de esta propiedad podemos asignar valor a los campos de la linea especificando que su tipo es variant.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsStringDes[sCampo:String]: String [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerDes[sCampo:String]: Integer [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatDes[sCampo:String]: Double [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanDes[sCampo:String]: Boolean [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es lógico.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsCurrencyDes[sCampo:String]: Currency [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es currency.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsVariantDes[sCampo:String]: OleVariant [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es variant.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Variant	Valor del campo

property ValidarPrecios: WordBool		
A través de esta propiedad podemos asignar valor si queremos mostrar o ocultar los avisos de precios por debajo del precio mínimo		
Valor retornado	Lógico	Valor del campo

property ValidarArtBloqueado: WordBool		
A través de esta propiedad podemos asignar valor si queremos mostrar o ocultar los avisos de artículo bloqueado		
Valor retornado	Lógico	Valor del campo

property AvisarCentroCosteCabecera: WordBool		
A través de esta propiedad podemos asignar valor si queremos mostrar o ocultar el aviso para cambiar el centro de coste en las líneas cuando se cambia desde la cabecera		
Valor retornado	Lógico	Valor del campo

A continuación, se presentan las propiedades y métodos necesarios para crear, **borrar y modificar ofertas.**

property Estado:EstadoMaestro		
Esta propiedad nos indica el estado del objeto. Posibles estados: ESTM_NOACTIVO: Documento cerrado. No se puede usar. ESTM_ACTIVO: Documento activo. Podemos añadir, borrar, modificar. ESTM_NUEVO: Documento en estado de inserción de un nuevo registro. ESTM_EDICION: Documento en estado modificación, podemos asignar valores a los campos.		
Valor retornado	EstadoMaestro	Retorna el estado del documento.

procedure Iniciar;
Este procedimiento es el que reserva los recursos necesarios para que el documento pueda ser usado. Al iniciar el documento el estado cambia de ESTM_NOACTIVO a ESTM_ACTIVO.

procedure Acabar;
Este procedimiento es el que libera los recursos utilizados por el objeto. Al acabar el documento el estado cambia de ESTM_ACTIVO a ESTM_NOACTIVO.

procedure Nuevo(sFecha: String; sCodCliPro: String; bEsDeCompra: Boolean);		
Inserta un nuevo registro y los prepara para ser editado. Pasa de estado ESTM_ACTIVO a ESTM_NUEVO.		
Parámetros:	sFecha	Fecha del documento
	sCodCliPro	Código del cliente o proveedor
	bEsDeCompra	True → Compra False → Venta

procedure Modifica(nIdDocu: Currency; bEsDeCompra: WordBool);		
Pone el documento que tiene el identificador indicado en edición.		
Parámetros:	nIdDocu	Identificador del documento
	bEsDeCompra	True → Compra False → Venta

procedure ModificaParcial(IdDocu: Currency; NumLin: Integer; EsDeCompra: WordBool);		
Este método se utiliza únicamente para cambiar el detalle de una línea de un documento. Se puede utilizar en documentos con muchas líneas, mejorando la velocidad, ya que solo carga el detalle de la línea que se indica. Despues de llamar a este método solo se pueden utilizar los métodos AnadirDetalle, CambiarDetalle, BorrarDetalle, AnadirLinea, CancelaLin, Anade y Cancela. Pone el documento que tiene el identificador indicado en edición y la línea que tiene el Numlin indicado en edición		
Parámetros:	IdDocu	Identificador del documento
	NumLin	Numero de línea
	EsDeCompra	True → Compra False → Venta

Procedure Anade;
Almacena los valores de los campos en la base de dato con los cambios introducidos por el programador. Genera las repercusiones del documento como pueden ser: actualización de stocks y actualización de estadísticas. Pasa de los estados ESTM_NUEVO o EST_EDICION a ESTM_ACTIVO.

Procedure Cancela;
Cancela la edición del documento. Pasa de los estados ESTM_NUEVO o EST_EDICION a ESTM_ACTIVO.

procedure Borra(nIdDocu: Currency; bEsDeCompra: Boolean);		
Elimina el documento con el identificador indicado eliminando también sus repercusiones en stocks, contabilidad, cartera, estadísticas e IVA.		
Parámetros:	nIdDocu	Identificador del documento
	bEsDeCompra	True → Compra False → Venta

Procedure NuevaLinea;
Crea una línea con los valores por defecto. A partir de ese momento podremos editar los valores de la línea.

Procedure NuevaLineaArt(sCodArt: string; nUnidades: Currency);		
Crea una línea con el artículo y las unidades indicadas realizando las búsquedas necesarias de los valores correspondientes a cuentas, precios y descuentos, etc... A partir de ese momento podremos editar los valores de la línea.		
Parámetros:	sCodArt	Código del artículo
	nUnidades	Unidades

Procedure AnadirLinea;
Almacena los valores de los campos de la línea en la base de datos.

Procedure CancelaLin;
Cancela la edición de la línea activa.

procedure EditarLinea(nIdLinea: Currency);		
Pone la línea indicada en el parámetro nIdLinea en edición, preparada para editar el contenido de sus campos. Al final de las modificaciones deberá llamarse a AnadirLinea.		
Parámetros:	nIdLinea	Identificador de la línea.. Se corresponde con el campo NUMLINOFE del fichero LINEOFER. Para localizarlo debe realizarse una consulta de este campo (directamente a la base de datos) en los registros con IdOfeV = a3ERPACTIVEXOferta.AsStringCab['IdOfeV'] en ventas o IdOfeC = a3ERPACTIVEXOferta.AsStringCab['IdOfeC'] en compras.

procedure BorrarLinea(nIdLinea: Currency);		
Borra la línea con el identificador indicado en el parámetro nIdLinea.		
Parámetros:	nIdLinea	Identificador de la línea. Se corresponde con el campo NUMLINOFE del fichero LINEOFER. Para localizarlo debe realizarse una consulta de este campo (directamente a la base de datos) en los registros con IdOfeV = a3ERPACTIVEXOferta.AsStringCab['IdOfeV'] en ventas o IdOfeC = a3ERPACTIVEXOferta.AsStringCab['IdOfeC'] en compras.

Los procedimientos que se describen a continuación se utilizan **para especificar los lotes, nº de serie, fechas de caducidad y ubicaciones de una línea** a no ser que esta selección se realice de forma automática. Para consultar si ya se han detallado todos los lotes, nº de serie, fechas de caducidad y ubicaciones necesarias se debe llamar a la función ConsultarDetalle;

procedure AnadirDetalle(nUnidades: Double; sNumSerie: WideString; sLote: WideString; sUbicacion: WideString; sFecCadu: WideString);		
Indica las unidades necesarias y su nº serie, lote, fecha de caducidad y ubicación de la línea activa. Para conocer los nº serie, lote, fecha de caducidad y ubicaciones disponibles en stock hay que llamar a la función **Varios.DetalleStockDisponible**..		
Parámetros:	nNunidades	Nº de unidades de la línea de detalle.
	sNumSerie	Nº de serie.
	sLote	Lote.
	sUbicacion	Ubicación.
	sFecCadu	Fecha de caducidad.

procedure CambiarDetalle(nIdLine: Double; sNumSerie: WideString; sLote: WideString; sUbicacion: WideString; sFecCadu: WideString);		
Indica las unidades necesarias y su Nº serie, lote, fecha de caducidad y ubicación de la línea activa. Para conocer los nº serie, lote, fecha de caducidad y ubicaciones disponibles en stock hay que llamar a la función **Varios.DetalleStockDisponible**. Para consultar el detalle de la línea activa hay que llamar a la función **ConsultarDetalle**.		
Parámetros:	nIdLine	Identificador de la línea. Se corresponde con el campo IDLIN de la tabla LINEOFER.
	nUnidades	Nº de unidades de la línea de detalle.
	sNumSerie	Nº de serie.
	sLote	Lote.

	sUbicacion	Ubicación.
	sFecCadu	Fecha de caducidad.

procedure BorrarDetalle(nIdLine: Double);		
Elimina la línea de detalle indicada en el parámetro.		
Parámetros:	nIdLine	Identificador de la línea. Se corresponde con el campo IDLIN de la tabla LINEOFER.

procedure AnadirTalla(sCodFamTallaH, sCodFamTallaV, sCodTallaH, sCodTallaV: WideString; nUnidades, nPrcMoneda:Double);		
Indica las unidades y el precio necesarios, junto con sus tallas y colores de la línea activa.		
Parámetros:	sCodFamTallaH	Familia de talla horizontal
	sCodFamTallaV	Familia de talla vertical
	sCodTallaH	Código de talla horizontal
	sCodTallaV	Código de talla vertical
	nUnidades	Nº unidades de la línea de talla
	nPrcMoneda	Precio de la línea de talla.

procedure CambiarTalla(nIdLine, nUnidades, nPrcMoneda: Double);		
Indica las unidades y el precio de la línea activa.		
Parámetros:	nIdLine	Identificador de la línea. Se corresponde con el campo IDLIN de la tabla LINEOFER.
	nUnidades	Nº de unidades de la línea de talla.
	nPrcMoneda	Precio de la línea de talla.

procedure BorrarTalla(nIdLine: Double);		
Elimina la línea de talla indicada en el parámetro.		
Parámetros:	nIdLine	Identificador de la línea. Se corresponde con el campo IDLIN de la tabla LINEOFER.

procedure AnadirTallaColor(sCodTallaH, sCodTallaV: WideString; nUnidades:Double);		
Indica las unidades, junto con su talla y color de la línea activa.		
Parámetros:	sCodTallaH	Código de talla horizontal
	sCodTallaV	Código de talla vertical
	nUnidades	Nº unidades de la línea de talla

Procedure NuevoComponente;
Crea una línea con los valores por defecto. A partir de ese momento podremos editar los valores de la línea componente

Procedure AnadirComponente;
Almacena los valores de los campos de la línea componente en la base de datos.

Procedure CancelaComponente;
Cancela la edición de la línea componente activa.

procedure EditarComponente(nIdLinea: Currency);		
Pone la línea componente indicada en el parámetro nIdLinea en edición, preparada para editar el contenido de sus campos. Al final de las modificaciones deberá llamarse a AnadirComponente.		
Parámetros:	nIdLinea	Identificador de la línea componente. Se corresponde con el campo NUMLINOFE del fichero LINEOFER.

procedure BorrarComponente(nIdLinea: Currency);		
Borra la línea componente con el identificador indicado en el parámetro nIdLinea.		
Parámetros:	nIdLinea	Identificador de la línea componente. Se corresponde con el campo NUMLINOFE del fichero LINEOFER.

function AnularUnidades(IdLin: Currency; Bultos: Double; Paquetes: Double; Unidades: Double; Fecha: TDateTime; const Motivo: WideString): Currency;		
Anula las unidades, bultos, paquetes en la fecha indicada para el identificador que se indica en el parámetro IdLin.		
Parámetros:	IdLin	Identificador de la línea. Se corresponde con el campo IDLIN del LINEOFER.
	Bultos	Número de bultos.
	Paquetes	Número de paquetes.
	Unidades	Número de unidades a anular.
	Fecha	Fecha de la anulación.
	Motivo	Motivo por el que se anulan las unidades.
Valor retornado:	Currency	Devuelve el valor del campo IDA3ERPACTIVEX de la tabla __ANULACIONLINEOFER.

procedure DeshacerAnularUnidades(IdLin: Currency; IdA3ERPACTIVEX: Currency);		
Deshace una anulación de unidades realizada previamente.		
Parámetros:	IdLin	Identificador de la línea. Se corresponde con el campo IDLIN del LINEOFER.
	IdA3ERPACTIVEX	Identificador de la anulación realizada previamente. Se corresponde con el campo IDA3ERPACTIVEX de la tabla __ANULACIONLINEOFER.

procedure CalcularImpuestosyTotales;
Realiza el cálculo de los impuestos y los totales del documento.

procedure AnadirSubdetalle(const ParametrosDetalle: ILineaDetalleParametros);		
Permite añadir líneas de detalle (lotes, num serie…) de la línea del componente de un kit		
Parámetros:	ParametrosDetalle	Clase que implementa la interfaz ILineaDetalleParametros. La que nos permite informar del detalle a añadir

procedure BorrarSubdetalle(IdLinea: Currency);		
Premite borrar la línea de detalle (lotes, num serie..) de la línea del componente de un kit		
Parámetros:	IdLinea	Identificador de la línea (campo IDLIN) para localizar el detalle y borrarlo

procedure CambiarSubdetalle(const ParametrosDetalle: ILineaDetalleParametros);		
Permite cambiar la línea de detalle (lotes, num serie…) de la línea del componente de un kit		
Parámetros:	ParametrosDetalle	Clase que implementa la interfaz ILineaDetalleParametros. La que nos permite informar del detalle a modificar. Si el valor de la línea es -1 cambiará el registro posicionado sin tener necesidad de buscarlo. Pensado para cuando sólo hay un detalle

function ILineasDocumento.ObtenerComponentesLinea: OleVariant
Permite obtener todas las líneas de componentes del artículo kit posicionado Devuelve una array de arrays con la siguiente estructura: A[0] => Numero líneas de componentes A[1..N][0] => Número de campos. Ejemplo 150 campos A[1..n][1..n][0] => Nombre del campo. Ejemplo 'CodArt' A[1..n][1..n][1] => Valor del campo. Ejemplo ' 1'

function ILineasDocumento.ObtenerDetalleComponentesLinea: OleVariant;
FunciónPermite obtener todas las líneas de detalle del componente del kit posicionado Devuelve una array de arrays con la siguiente estructura: A[0] => Numero líneas de detalle A[1..N][0] => Número de campos. Ejemplo 150 campos A[1..n][1..n][0] => Nombre del campo. Ejemplo 'CodArt' A[1..n][1..n][1] => Valor del campo. Ejemplo 'LOTE100'

function ILineasDocumento.ObtenerDetallesLinea: OleVariant;
Permite obtener todas las líneas de detalle del artículo padre posicionado Devuelve una array de arrays con la siguiente estructura: A[0] => Numero líneas de detalle A[1..N][0] => Número de campos. Ejemplo 150 campos A[1..n][1..n][0] => Nombre del campo. Ejemplo 'CodArt' A[1..n][1..n][1] => Valor del campo. Ejemplo 'LOTE100'

function ILineasDocumento.ObtenerLineas: OleVariant;
Permite obtener todas las líneas sin detalle/componentes del documento Devuelve una array de arrays con la siguiente estructura: A[0] => Numero líneas del documento A[1..N][0] => Número de campos. Ejemplo 150 campos A[1..n][1..n][0] => Nombre del campo. Ejemplo 'CodArt' A[1..n][1..n][1] => Valor del campo. Ejemplo 'B10

17.- OBJETO OFERTA

Extiende Oferta140501 y añade nuevos métodos.

Interface Oferta
procedure Clonar(const DocumentoOrigenTipo: WideString; DocumentoOrigenID: Currency; BorrarLineasDelDocumentoDestino: WordBool; ClonarVariablesDeDocumentos: WordBool); [Nuevo 14.05.02] **procedure Copiar(const DocumentoOrigenTipo: WideString; DocumentoOrigenID: Currency; BorrarLineasDelDocumentoDestino: WordBool; CopiarCondicionesFinancieras: WordBool; CopiarLosPortes: WordBool; CopiarPreciosYDescuentos: WordBool; CopiarDescripcionesYObservacionesLineas: WordBool);** [Nuevo 14.05.02]

procedure Clonar(const DocumentoOrigenTipo: WideString; DocumentoOrigenID: Currency; BorrarLineasDelDocumentoDestino: WordBool; ClonarVariablesDeDocumentos: WordBool);		
Mediante este procedimiento se permite clonar un documento sobre otro.		
Parámetros:		
DocumentoOrigenTipo	'OV' Oferta venta 'PV' Pedido de venta 'AV' Albarán de venta 'DV' Depósito de venta 'FV' Factura de venta	'OC' Oferta compra 'PC' Pedido de compra 'AC' Albarán de compra 'DC' Depósito de compra 'FC' Factura de compra
DocumentoOrigenID	Identificador del documento de origen	
BorrarLineasDelDocumentoDestino	Antes de realizar la operación, borra las lineas del documento destino	
ClonarVariablesDeDocumentos	Se clonarán las variables de los documentos	

procedure Copiar(const DocumentoOrigenTipo: WideString; DocumentoOrigenID: Currency; BorrarLineasDelDocumentoDestino: WordBool; CopiarCondicionesFinancieras: WordBool; CopiarLosPortes: WordBool; CopiarPreciosYDescuentos: WordBool; CopiarDescripcionesYObservacionesLineas: WordBool);		
Mediante este procedimiento se permite copiar un documento sobre otro.		
Parámetros:		
DocumentoOrigenTipo	'OV' Oferta venta 'PV' Pedido de venta 'AV' Albarán de venta 'DV' Depósito de venta 'FV' Factura de venta	'OC' Oferta compra 'PC' Pedido de compra 'AC' Albarán de compra 'DC' Depósito de compra 'FC' Factura de compra
DocumentoOrigenID	Identificador del documento de origen	
BorrarLineasDelDocumentoDestino	Antes de realizar la operación, borra las lineas del documento destino	
CopiarCondicionesFinancieras	Copiar condiciones financieras del documento origen. Se copiarán del documento origen la forma de pago, documento de pago, descuento de pronto pago o recargo financiero y si éste afecta a la base imponible.	

CopiarLosPortes	Copiar los portes al documento destino
CopiarPreciosYDescuentos	Copiar precios y descuentos del documento origen
CopiarDescripcionesYObservacionesLineas	Copiar descripciones y observaciones de las líneas del documento origen

18.- OBJETO REGULARIZACION

Objeto que permite la creación, **modificación y borrado de regularizaciones**.

Interface Regularizacion
property Estado: EstadoMaestro readonly; **property** AsStringCab[const sCampo: WideString]: WideString; **property** AsFloatCab[const sCampo: WideString]: Double; **property** AsIntegerCab[const sCampo: WideString]: Integer; **property** AsBooleanCab[const sCampo: WideString]: WordBool; **property** AsCurrencyCab[const sCampo: WideString]: Currency; **property** AsVariantCab[const sCampo: WideString]: OleVariant; **property** AsStringLin[const sCampo: WideString]: WideString; **property** AsFloatLin[const sCampo: WideString]: Double; **property** AsIntegerLin[const sCampo: WideString]: Integer; **property** AsBooleanLin[const sCampo: WideString]: WordBool; **property** AsCurrencyLin[const sCampo: WideString]: Currency; **property** AsVariantLin[const sCampo: WideString]: OleVariant; **property** AsBooleanDes[const sCampo: WideString]: WordBool; [SOLO LECTURA] **property** AsStringDes[const sCampo: WideString]: WideString; [SOLO LECTURA] **property** AsFloatDes[const sCampo: WideString]: Double; [SOLO LECTURA] **property** AsIntegerDes[const sCampo: WideString]: Integer; [SOLO LECTURA] **property** AsCurrencyDes[const sCampo: WideString]: Currency; [SOLO LECTURA] **property** AsVariantDes[const sCampo: WideString]: OleVariant; [SOLO LECTURA] **property** AvisarStock: WordBool; **procedure** Iniciar; **procedure** Acabar; **procedure** Cancela; **procedure** Nuevo(sFecha: WideString; sCodAlm: WideString); **procedure** Borra(IdDocu: Currency); **procedure** Modifica(IdDocu: Currency); **function** Anade: Currency; **procedure** NuevaLinea; **procedure** AnadirLinea; **procedure** CancelaLin; **procedure** NuevaLineaArt(const sCodArt: WideString; nUnidades: Currency);

```
procedure EditarLinea(nIdLinea: Currency);
procedure BorrarLinea(nIdLinea: Currency);
procedure AnadirDetalle( nUnidades: Double; sNumSerie: WideString; sLote: WideString;
                         sUbicacion: WideString; sFecCadu: WideString);
procedure CambiarDetalle( nIdLine: Double; nUnidades: Double; sNumSerie: WideString;
                          sLote: WideString; sUbicacion: WideString;  sFecCadu: WideString);
procedure BorrarDetalle(nIdLine: Double);
procedure AnadirTalla(sCodFamTallaH, sCodFamTallaV, sCodTallaH,
    sCodTallaV: WideString; nUnidades: Double);
procedure CambiarTalla(nIdLine, nUnidades: Double);
procedure BorrarTalla(nIdLine: Double);
property OmitirMensajes: WordBool;
```

Propiedades	Tipo	Descripción
AsStringCab	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsIntegerCab	Tabla(Integer)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsFloatCab	Tabla(Float)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsBooleanCab	Tabla(Lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsCurrencyCab	Tabla(Currency)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsVariantCab	Tabla(Variant)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsStringLin	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsIntegerLin	Tabla(Integer)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsFloatLin	Tabla(Float)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsBooleanLin	Tabla(Lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsCurrencyLin	Tabla(Currency)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsStringDes	Tabla(string)	Lee valores de los campos del registro nuevo/actual.
AsIntegerDes	Tabla(Integer)	Lee valores de los campos del registro nuevo/actual.
AsFloatDes	Tabla(Float)	Lee valores de los campos del registro nuevo/actual.
AsBooleanDes	Tabla(Lógico)	Lee valores de los campos del registro nuevo/actual.

AsCurrencyDes	Tabla(Currency)	Lee valores de los campos del registro nuevo/actual.
AsVariantDes	Tabla(Variant)	Lee valores de los campos del registro nuevo/actual.
Estado	EstadoMaestro	Devuelve el estado en que se encuentra el objeto
AsVariantLin	Tabla(Variant)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AvisarStock	Lógico	Asignar valor para activar o no los mensajes de aviso de stock

*EstadoMaestro: Ver objeto Maestro.

Método	Tipo	Descripción
Iniciar	Procedimiento	Inicia el objeto
Acabar	Procedimiento	Cierra el objeto
Nuevo	Procedimiento	Inicia un nuevo documento
Borra	Procedimiento	Borra el documento
Modifica	Procedimiento	Prepara el documento para poderlo modificar.
Cancela	Procedimiento	Cancela la edición del documento.
Anade	Función	Añade los datos editados del documento a la base de datos y devuelve el identificador con el que se almacenará.
NuevaLinea	Procedimiento	Inicia una nueva línea
NuevaLineaArt	Procedimiento	Inicia una nueva línea aplicando las políticas de nexus, dados el código del artículo y las unidades pasadas como parámetro.
AnadirLinea	Procedimiento	Añade la línea
CancelaLin	Procedimiento	Cancela la línea en edición
EditarLinea	Procedimiento	Permite editar una línea ya existente del documento.
BorrarLinea	Procedimiento	Permite borrar una línea del documento.
AnadirDetalle	Procedimiento	Identifica los nº de serie, lotes, fechas de caducidad y/o ubicaciones de una línea del documento.
CambiarDetalle	Procedimiento	Cambia los nº de serie, lotes, fechas de caducidad y/o ubicaciones de una línea del documento.
BorrarDetalle	Procedimiento	Borra el detalle los nº de serie, lotes, fechas de caducidad y/o ubicaciones de una línea del documento.
AnadirTalla	Procedimiento	Identifica las tallas de una línea del documento.
CambiarTalla	Procedimiento	Cambia las tallas de una línea del documento.

BorrarTalla	Procedimiento	Borra el detalle de las tallas de una línea del documento.
OmitirMensajes	Lógico	Asignar valor para ocultar mensajes de información

Las primeras funciones de la lista que se presentan a continuación se utilizan para asignar / leer valores de la cabecera o de las líneas del documento.

Deben tenerse en cuenta algunas normas importantes.

- Si el nombre del campo no existe se producirá un error.
- Si el valor asignado al campo especificado no es correcto, no se producirá un error. Este se producirá al intentar guardar el documento o la línea.

Hay que tener en cuenta lo siguiente:

- Los campos fecha se han de asignar como cadenas de texto como '01/01/2003'
- Uso en Delphi: a3ERPACTIVEXDocumento.AsStringLin['CodArt']
- Uso en Visual Basic: a3ERPACTIVEXDocumento.AsStringLin ('CodArt')

property AsStringCab[sCampo:String]: String		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerCab[sCampo:String]: Integer		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatCab[sCampo:String]: Double		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanCab[sCampo:String]: Boolean		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es lógico.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsCurrencyCab[sCampo:String]: Currency		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es currency.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsVariantCab[sCampo:String]: OleVariant		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es variant.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Variant	Valor del campo

property AsStringLin[sCampo:String]: String		
Con esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerLin[sCampo:String]: Integer		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatLin[sCampo:String]: Double		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanLin[sCampo:String]: Boolean		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es lógico.		
Parámetros:	sCampo	Nombre del campo

Valor retornado	Lógico	Valor del campo

property AsCurrencyLin[sCampo:String]: Currency		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es currency.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsVariantLin[sCampo:String]: OleVariant		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es variant.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Variant	Valor del campo

property AsStringDes[sCampo:String]: String [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerDes[sCampo:String]: Integer [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatDes[sCampo:String]: Double [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanDes[sCampo:String]: Boolean [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es lógico.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsCurrencyDes[sCampo:String]: Currency [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es currency.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsVariantDes[sCampo:String]: OleVariant [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es variant.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Variant	Valor del campo

property AvisarStock: WordBool		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es lógico.		
Valor retornado	Currency	Valor del campo

A continuación. se presentan las **propiedades y métodos necesarios para crear, borrar y modificar regularizaciones.**

property Estado:EstadoMaestro		
Esta propiedad nos indica el estado del objeto. Posibles estados: ESTM_NOACTIVO: Documento cerrado. No se puede usar. ESTM_ACTIVO: Documento activo. Podemos añadir, borrar, modificar. ESTM_NUEVO: Documento en estado de inserción de un nuevo registro. ESTM_EDICION: Documento en estado modificación, podemos asignar valores a los campos.		
Valor retornado	EstadoMaestro	Retorna el estado del documento.

procedure Iniciar;
Este procedimiento es el que reserva los recursos necesarios para que el documento pueda ser usado. Al iniciar el documento el estado cambia de ESTM_NOACTIVO a ESTM_ACTIVO.

procedure Acabar;
Este procedimiento es el que libera los recursos utilizados por el objeto. Al acabar el documento el estado cambia de ESTM_ACTIVO a ESTM_NOACTIVO.

procedure Nuevo(sFecha: String; sCodAlm: String);		
Inserta un nuevo registro y los prepara para ser editado. Pasa de estado ESTM_ACTIVO a ESTM_NUEVO.		
Parámetros:	sFecha	Fecha del documento
	sCodAlm	Código del almacén

procedure Modifica(nIdDocu: Currency);		
Pone el documento que tiene el identificador indicado en edición.		
Parámetros:	nIdDocu	Identificador del documento

Procedure Anade;
Almacena los valores de los campos en la base de dato con los cambios introducidos por el programador. Genera las repercusiones del documento como pueden ser: actualización de stocks y actualización de estadísticas. Pasa de los estados ESTM_NUEVO o EST_EDICION a ESTM_ACTIVO.

Procedure Cancela;
Cancela la edición del documento. Pasa de los estados ESTM_NUEVO o EST_EDICION a ESTM_ACTIVO.

procedure Borra(nIdDocu: Currency);		
Elimina el documento con el identificador indicado eliminando también sus repercusiones en stocks, contabilidad, cartera, estadísticas e IVA.		
Parámetros:	nIdDocu	Identificador del documento

Procedure NuevaLinea;
Crea una línea con los valores por defecto. A partir de ese momento podremos editar los valores de la línea.

Procedure NuevaLineaArt(sCodArt: string; nUnidades: Currency);		
Crea una línea con el artículo y las unidades indicadas. A partir de ese momento podremos editar los valores de la línea.		
Parámetros:	sCodArt	Código del artículo
	nUnidades	Unidades

Procedure AnadirLinea;
Almacena los valores de los campos de la línea en la base de datos.

Procedure CancelaLin;
Cancela la edición de la línea activa.

procedure EditarLinea(nIdLinea: Currency);		
Pone la línea indicada en el parámetro nIdLinea en edición, preparada para editar el contenido de sus campos. Al final de las modificaciones deberá llamarse a AnadirLinea.		
Parámetros:	nIdLinea	Identificador de la línea.. Se corresponde con el campo NUMLINREG del fichero LINEREGU. Para localizarlo debe realizarse una consulta de este campo (directamente a la base de datos) en los registros con IdReg = a3ERPACTIVEXRegu.AsStringCab['IdReg'].

procedure BorrarLinea(nIdLinea: Currency);		
Borra la línea con el identificador indicado en el parámetro nIdLinea.		
Parámetros:	nIdLinea	Identificador de la línea.. Se corresponde con el campo NUMLINREG del fichero LINEREGU. Para localizarlo debe realizarse una consulta de este campo (directamente a la base de datos) en los registros con IdReg = a3ERPACTIVEXRegu.AsStringCab['IdReg'].

Los procedimientos que se describen a continuación se utilizan para **especificar los lotes, nº de serie, fechas de caducidad y ubicaciones de una línea** a no ser que esta selección se realice de forma automática. Para consultar si ya se han detallado todos los lotes, nº de serie, fechas de caducidad y ubicaciones necesarios se debe llamar a la función ConsultarDetalle;

procedure AnadirDetalle(nUnidades: Double; sNumSerie: WideString; sLote: WideString; sUbicacion: WideString; sFecCadu: WideString);		
Indica las unidades necesarias y su nº serie, lote, fecha de caducidad y ubicación de la línea activa. Para conocer los nº serie, lote, fecha de caducidad y ubicaciones disponibles en stock hay que llamar a la función **Varios.DetalleStockDisponible**..		
Parámetros:	nUnidades	Nº de unidades de la línea de detalle.

	sNumSerie	Nº de serie.
	sLote	Lote.
	sUbicacion	Ubicación.
	sFecCadu	Fecha de caducidad.

procedure CambiarDetalle(nIdLine: Double; sNumSerie: WideString; sLote: WideString; sUbicacion: WideString; sFecCadu: WideString);		
Indica las unidades necesarias y su Nº serie, lote, fecha de caducidad y ubicación de la línea activa. Para conocer los nº serie, lote, fecha de caducidad y ubicaciones disponibles en stock hay que llamar a la función **Varios.DetalleStockDisponible**. Para consultar el detalle de la línea activa hay que llamar a la función **ConsultarDetalle**.		
Parámetros:	nIdLine	Identificador de la línea. Se corresponde con el campo IDLIN de la tabla LINEREGU.
	nUnidades	Nº de unidades de la línea de detalle.
	sNumSerie	Nº de serie.
	sLote	Lote.
	sUbicacion	Ubicación.
	sFecCadu	Fecha de caducidad.

procedure BorrarDetalle(nIdLine: Double);		
Elimina la línea de detalle indicada en el parámetro.		
Parámetros:	nIdLine	Identificador de la línea. Se corresponde con el campo IDLIN de la tabla LINEREGU.

procedure AnadirTalla(sCodFamTallaH, sCodFamTallaV, sCodTallaH, sCodTallaV: WideString; nUnidades:Double);		
Indica las unidades y el precio necesarios, junto con sus tallas y colores de la línea activa.		
Parámetros:	sCodFamTallaH	Familia de talla horizontal
	sCodFamTallaV	Familia de talla vertical
	sCodTallaH	Código de talla horizontal
	sCodTallaV	Código de talla vertical
	nUnidades	Nº unidades de la línea de talla

procedure CambiarTalla(nIdLine, nUnidades: Double);		
Indica las unidades y el precio de la línea activa.		
Parámetros:	nIdLine	Identificador de la línea. Se corresponde con el campo IDLIN de la tabla LINEREGU.
	nUnidades	Nº de unidades de la línea de talla.

procedure BorrarTalla(nIdLine: Double);		
Elimina la línea de talla indicada en el parámetro.		
Parámetros:	nIdLine	Identificador de la línea. Se corresponde con el campo IDLIN de la tabla LINEREGU.

property OmitirMensajes:Boolean
Para activar o desactivar mensajes de información.

19.- OBJETO TRASPASO

Objeto que permite la creación, modificación y borrado de traspasos.

Interface Traspaso
property Estado: EstadoMaestro readonly; **property** AsStringCab[const sCampo: WideString]: WideString; **property** AsFloatCab[const sCampo: WideString]: Double; **property** AsIntegerCab[const sCampo: WideString]: Integer; **property** AsBooleanCab[const sCampo: WideString]: WordBool; **property** AsCurrencyCab[const sCampo: WideString]: Currency; **property** AsVariantCab[const sCampo: WideString]: Variant; **property** AsStringLin[const sCampo: WideString]: WideString; **property** AsFloatLin[const sCampo: WideString]: Double; **property** AsIntegerLin[const sCampo: WideString]: Integer; **property** AsBooleanLin[const sCampo: WideString]: WordBool; **property** AsCurrencyLin[const sCampo: WideString]: Currency; **property** AsVariantLin[const sCampo: WideString]: Variant; **property** AsBooleanDes[const sCampo: WideString]: WordBool; [SOLO LECTURA] **property** AsStringDes[const sCampo: WideString]: WideString; [SOLO LECTURA] **property** AsFloatDes[const sCampo: WideString]: Double; [SOLO LECTURA] **property** AsIntegerDes[const sCampo: WideString]: Integer; [SOLO LECTURA] **property** AsCurrencyDes[const sCampo: WideString]: Currency; [SOLO LECTURA] **property** AsVariantDes[const sCampo: WideString]: OleVariant; [SOLO LECTURA] **property** AvisarStock: WordBool; **procedure** Iniciar; **procedure** Acabar; **procedure** Cancela; **procedure** Nuevo(sFecha: WideString; sCodAlmSal, sCodalmEnt: WideString); **procedure** NuevoReubica (sFecha: WideString; sCodAlm: WideString); **procedure** Borra(IdDocu: Currency); **procedure** Modifica(IdDocu: Currency); **function** Anade: Currency; **procedure** NuevaLinea;

```
procedure AnadirLinea;
procedure CancelaLin;
procedure NuevaLineaArt(const sCodArt: WideString; nUnidades: Currency);
procedure EditarLinea(nIdLinea: Currency);
procedure BorrarLinea(nIdLinea: Currency);
procedure AnadirDetalle( nUnidades: Double; sNumSerie: WideString; sLote: WideString;
                         sUbicacion1, sUnicacion2: WideString; sFecCadu: WideString);
procedure CambiarDetalle( nIdLine: Double; nUnidades: Double; sNumSerie: WideString;
                          sLote: WideString; sUbicacion1, sUnicacion2: WideString;
                          sFecCadu: WideString);
procedure BorrarDetalle(nIdLine: Double);
procedure AnadirTalla(sCodFamTallaH, sCodFamTallaV, sCodTallaH,
    sCodTallaV: WideString; nUnidades: Double);
procedure CambiarTalla(nIdLine, nUnidades: Double);
procedure BorrarTalla(nIdLine: Double);
property OmitirMensajes: WordBool;
```

Propiedades	Tipo	Descripción
AsStringCab	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsIntegerCab	Tabla(Integer)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsFloatCab	Tabla(Float)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsBooleanCab	Tabla(Lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsCurrencyCab	Tabla(Currency)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsVariantCab	Tabla(Variant)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsStringLin	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsIntegerLin	Tabla(Integer)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsFloatLin	Tabla(Float)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsBooleanLin	Tabla(Lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsCurrencyLin	Tabla(Currency)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsVariantLin	Tabla(Variant)	Asigna/lee valores a/de los campos del registro nuevo/actual.

AsStringDes	Tabla(string)	Lee valores de los campos del registro nuevo/actual.
AsIntegerDes	Tabla(Integer)	Lee valores de los campos del registro nuevo/actual.
AsFloatDes	Tabla(Float)	Lee valores de los campos del registro nuevo/actual.
AsBooleanDes	Tabla(Lógico)	Lee valores de los campos del registro nuevo/actual.
AsCurrencyDes	Tabla(Currency)	Lee valores de los campos del registro nuevo/actual.
AsVariantDes	Tabla(Variant)	Lee valores de los campos del registro nuevo/actual.
Estado	EstadoMaestro	Devuelve el estado en que se encuentra el objeto
AvisarStock	Lógico	Asignar valor para activar o no los mensajes de aviso de stock

*EstadoMaestro: Ver objeto Maestro.

Método	**Tipo**	**Descripción**
Iniciar	Procedimiento	Inicia el objeto
Acabar	Procedimiento	Cierra el objeto
Nuevo	Procedimiento	Inicia un nuevo documento
NuevoReubica	Procedimiento	Inicia un nuevo traspaso de reubicación. Sólo para almacenes con ubicaciones.
Borra	Procedimiento	Borra el documento
Modifica	Procedimiento	Prepara el documento para poderlo modificar.
Cancela	Procedimiento	Cancela la edición del documento.
Anade	Función	Añade los datos editados del documento a la base de datos y devuelve el identificador con el que se almacenará.
NuevaLinea	Procedimiento	Inicia una nueva línea
NuevaLineaArt	Procedimiento	Inicia una nueva línea aplicando las políticas de nexus, dados el código del artículo y las unidades pasadas como parámetro.
AnadirLinea	Procedimiento	Añade la línea
CancelaLin	Procedimiento	Cancela la línea en edición
EditarLinea	Procedimiento	Permite editar una línea ya existente del documento.
BorrarLinea	Procedimiento	Permite borrar una línea del documento.
AnadirDetalle	Procedimiento	Identifica los nº de serie, lotes, fechas de caducidad y/o ubicaciones de una línea del documento.

CambiarDetalle	Procedimiento	Cambia los nº de serie, lotes, fechas de caducidad y/o ubicaciones de una línea del documento.
BorrarDetalle	Procedimiento	Borra el detalle los nº de serie, lotes, fechas de caducidad y/o ubicaciones de una línea del documento.
AnadirTalla	Procedimiento	Identifica las tallas de una línea del documento.
CambiarTalla	Procedimiento	Cambia las tallas de una línea del documento.
BorrarTalla	Procedimiento	Borra el detalle de las tallas de una línea del documento.
OmitirMensajes	Lógico	Asignar valor para ocultar mensajes de información

Las primeras funciones de la lista que se presentan a continuación se utilizan para asignar / leer valores de la cabecera o de las líneas del documento.

Deben tenerse en cuenta algunas normas importantes.

- Si el nombre del campo no existe se producirá un error.
- Si el valor asignado al campo especificado no es correcto, no se producirá un error. Este se producirá al intentar guardar el documento o la línea.

Hay que tener en cuenta lo siguiente:

- Los campos fecha se han de asignar como cadenas de texto como '01/01/2003'
- Uso en Delphi: a3ERPACTIVEXDocumento.AsStringLin['CodArt']
- Uso en Visual Basic: a3ERPACTIVEXDocumento. AsStringLin ('CodArt')

property AsStringCab[sCampo:String]: String		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerCab[sCampo:String]: Integer		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatCab[sCampo:String]: Double		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanCab[sCampo:String]: Boolean		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es lógico.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsCurrencyCab[sCampo:String]: Currency		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es currency.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsStringLin[sCampo:String]: String		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerLin[sCampo:String]: Integer		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatLin[sCampo:String]: Double		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo

Valor retornado	Double	Valor del campo

property AsBooleanLin[sCampo:String]: Boolean		
A esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es lógico.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsCurrencyLin[sCampo:String]: Currency		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es currency.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsStringDes[sCampo:String]: String [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerDes[sCampo:String]: Integer [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatDes[sCampo:String]: Double [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanDes[sCampo:String]: Boolean [SOLO LECTURA]
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es lógico.

Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsCurrencyDes[sCampo:String]: Currency [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es currency.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsVariantDes[sCampo:String]: OleVariant [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es variant.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Variant	Valor del campo

property AvisarStock: WordBool		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es lógico.		
Valor retornado	Currency	Valor del campo

A continuación, se presentan las **propiedades y métodos necesarios para crear, borrar y modificar traspasos.**

property Estado:EstadoMaestro		
Esta propiedad nos indica el estado del objeto. Posibles estados: ESTM_NOACTIVO: Documento cerrado. No se puede usar. ESTM_ACTIVO: Documento activo. Podemos añadir, borrar, modificar. ESTM_NUEVO: Documento en estado de inserción de un nuevo registro. ESTM_EDICION: Documento en estado modificación, podemos asignar valores a los campos.		
Valor retornado	EstadoMaestro	Retorna el estado del documento.

procedure Iniciar;
Este procedimiento es el que reserva los recursos necesarios para que el documento pueda ser usado. Al iniciar el documento el estado cambia de ESTM_NOACTIVO a ESTM_ACTIVO.

procedure Acabar;
Este procedimiento es el que libera los recursos utilizados por el objeto. Al acabar el documento el estado cambia de ESTM_ACTIVO a ESTM_NOACTIVO.

procedure Nuevo(sFecha: String; sCodAlmSal, sCodAlmEnt: String);		
Inserta un nuevo registro y los prepara para ser editado. Pasa de estado ESTM_ACTIVO a ESTM_NUEVO.		
Parámetros:	sFecha	Fecha del documento
	sCodAlmSal	Código del almacén de salida.
	sCodAlmEnt	Código del almacén de entrada.

procedure NuevoReubica(sFecha: String; sCodAlm: String);		
Inserta un nuevo traspaso de reubicación y lo prepara para ser editado. Un traspaso de reubicación es aquel documento donde registramos los cambios de ubicación de artículos dentro de un mismo almacén. Sólo para almacenes con ubicaciones. La multiubicación se activa en datos generales ->Stock -> Configuración de lotes -> "Se usan múltiples ubicaciones" y se necesita gama plus o superior. Pasa de estado ESTM_ACTIVO a ESTM_NUEVO.		
Parámetros:	sFecha	Fecha del documento
	sCodAlm	Código del almacén.

procedure Modifica(nIdDocu: Currency);		
Pone el documento que tiene el identificador indicado en edición.		
Parámetros:	nIdDocu	Identificador del documento

Procedure Anade;
Almacena los valores de los campos en la base de dato con los cambios introducidos por el programador. Genera las repercusiones del documento como pueden ser: actualización de stocks y actualización de estadísticas. Pasa de los estados ESTM_NUEVO o EST_EDICION a ESTM_ACTIVO.

Procedure Cancela;
Cancela la edición del documento. Pasa de los estados ESTM_NUEVO o EST_EDICION a ESTM_ACTIVO.

procedure Borra(nIdDocu: Currency);		
Elimina el documento con el identificador indicado eliminando también sus repercusiones en stocks, contabilidad, cartera, estadísticas e IVA.		
Parámetros:	nIdDocu	Identificador del documento

Procedure NuevaLinea;
Crea una línea con los valores por defecto. A partir de ese momento podremos editar los valores de la línea.

Procedure NuevaLineaArt(sCodArt: string; nUnidades: Currency);		
Crea una línea con el artículo y las unidades indicadas. A partir de ese momento podremos editar los valores de la línea.		
Parámetros:	sCodArt	Código del artículo
	nUnidades	Unidades

Procedure AnadirLinea;
Almacena los valores de los campos de la línea en la base de datos.

Procedure CancelaLin;
Cancela la edición de la línea activa.

procedure EditarLinea(nIdLinea: Currency);		
Pone la línea indicada en el parámetro nIdLinea en edición, preparada para editar el contenido de sus campos. Al final de las modificaciones deberá llamarse a AnadirLinea.		
Parámetros:	nIdLinea	Identificador de la línea.. Se corresponde con el campo NUMLINTRA del fichero LINETRAS. Para localizarlo debe realizarse una consulta de este campo (directamente a la base de datos) en los registros con IdTra = a3ERPACTIVEXTraspaso.AsStringCab['IdTra'].

procedure BorrarLinea(nIdLinea: Currency);		
Borra la línea con el identificador indicado en el parámetro nIdLinea.		
Parámetros:	nIdLinea	Identificador de la línea.. Se corresponde con el campo NUMLINTRA del fichero LINETRAS. Para localizarlo debe realizarse una consulta de este campo (directamente a la base de datos) en los registros con IdTra = a3ERPACTIVEXTraspaso.AsStringCab['IdTra'].

Los procedimientos que se describen a continuación se utilizan para **especificar los lotes, nº de serie, fechas de caducidad y ubicaciones de una línea** a no ser que esta selección se realice de forma automática. Para consultar si ya se han detallado todos los lotes, nº de serie, fechas de caducidad y ubicaciones necesarios se debe llamar a la función ConsultarDetalle;

procedure AnadirDetalle(nUnidades: Double; sNumSerie: WideString; sLote: WideString; sUbicacion1, sUbicacion2: WideString; sFecCadu: WideString);		
Indica las unidades necesarias y su nº serie, lote, fecha de caducidad y ubicación de la línea activa. Para conocer los nº serie, lote, fecha de caducidad y ubicaciones disponibles en stock hay que llamar a la función **Varios.DetalleStockDisponible**..		
Parámetros:	nUnidades	Nº de unidades de la línea de detalle.
	sNumSerie	Nº de serie.
	sLote	Lote.
	sUbicacion1	Ubicación en almacén origen.
	sUbicacion2	Ubicación en almacén destino
	sFecCadu	Fecha de caducidad.

procedure CambiarDetalle(nIdLine: Double; sNumSerie: WideString; sLote: WideString; sUbicacion1, sUbicacion2: WideString; sFecCadu: WideString);		
Indica las unidades necesarias y su Nº serie, lote, fecha de caducidad y ubicación de la línea activa. Para conocer los nº serie, lote, fecha de caducidad y ubicaciones disponibles en stock hay que llamar a la función **Varios.DetalleStockDisponible**. Para consultar el detalle de la línea activa hay que llamar a la función **ConsultarDetalle**.		
Parámetros:	nIdLine	Identificador de la línea. Se corresponde con el campo IDLIN de la tabla LINETRAS.
	nUnidades	Nº de unidades de la línea de detalle.
	sNumSerie	Nº de serie.
	sLote	Lote.
	sUbicacion1	Ubicación en almacén origen.
	sUbicacion2	Ubicación en almacén destino
	sFecCadu	Fecha de caducidad.

procedure BorrarDetalle(nIdLine: Double);		
Elimina la línea de detalle indicada en el parámetro.		
Parámetros:	nIdLine	Identificador de la línea. Se corresponde con el campo IDLIN de la tabla LINETRAS.

procedure AnadirTalla(sCodFamTallaH, sCodFamTallaV, sCodTallaH, sCodTallaV: WideString; nUnidades:Double);		
Indica las unidades y el precio necesarios, junto con sus tallas y colores de la línea activa.		
Parámetros:	sCodFamTallaH	Familia de talla horizontal
	sCodFamTallaV	Familia de talla vertical

	sCodTallaH	Código de talla horizontal
	sCodTallaV	Código de talla vertical
	nUnidades	Nº unidades de la línea de talla

procedure CambiarTalla(nIdLine, nUnidades: Double);		
Indica las unidades y el precio de la línea activa.		
Parámetros:	nIdLine	Identificador de la línea. Se corresponde con el campo IDLIN de la tabla LINETRAS.
	nUnidades	Nº de unidades de la línea de talla.

procedure BorrarTalla(nIdLine: Double);		
Elimina la línea de talla indicada en el parámetro.		
Parámetros:	nIdLine	Identificador de la línea. Se corresponde con el campo IDLIN de la tabla LINETRAS.

property OmitirMensajes:Boolean
Para activar o desactivar mensajes de información.

20.- OBJETO INVENTARIO

Objeto que permite la **creación, modificación y borrado de inventarios**.

Interface Inventario
property Estado: EstadoMaestro readonly; **property** AsStringCab[const sCampo: WideString]: WideString; **property** AsFloatCab[const sCampo: WideString]: Double; **property** AsIntegerCab[const sCampo: WideString]: Integer; **property** AsBooleanCab[const sCampo: WideString]: WordBool; **property** AsVariantCab[const sCampo: WideString]: OleVariant; **property** AsStringLin[const sCampo: WideString]: WideString; **property** AsFloatLin[const sCampo: WideString]: Double; **property** AsIntegerLin[const sCampo: WideString]: Integer; **property** AsBooleanLin[const sCampo: WideString]: WordBool; **property** AsCurrencyLin[const sCampo: WideString]: Currency; **property** AsVariantLin[const sCampo: WideString]: OleVariant; **property** AsBooleanDes[const sCampo: WideString]: WordBool; [SOLO LECTURA] **property** AsStringDes[const sCampo: WideString]: WideString; [SOLO LECTURA] **property** AsFloatDes[const sCampo: WideString]: Double; [SOLO LECTURA] **property** AsIntegerDes[const sCampo: WideString]: Integer; [SOLO LECTURA] **property** AsCurrencyDes[const sCampo: WideString]: Currency; [SOLO LECTURA]

```
property AsVariantDes[const sCampo: WideString]: OleVariant; [SOLO LECTURA]
property OmitirMensajes: WordBool;
property AvisarStock: WordBool;
procedure Iniciar;
procedure Acabar;
procedure Cancela;
procedure Nuevo(  sFecha: WideString; sCodAlm: WideString; bTodosArti: WrodBool);
procedure Borra(IdDocu: Currency);
procedure Editar(IdDocu: Currency);
function Anade: Currency;
procedure NuevaLinea(const sCodArt: WideString; nUnidades: Double; nPrcMed: Double);
procedure AnadirLinea;
procedure CancelaLin;
procedure NuevaLineaArt(const sCodArt: WideString; nUnidades: Currency);
procedure EditarLinea(nIdLinea: Currency);
procedure BorrarLinea(nIdLinea: Currency);
procedure AnadirDetalle( nUnidades: Double;  nPrcMedio: Double; sNumSerie: WideString;
                         sLote: WideString; sUbicacion: WideString; sFecCadu: WideString);
procedure CambiarDetalle( nIdLine: Double; nUnidades: Double; nPrcMedio: Double;
                          sNumSerie: WideString; sLote: WideString; sUbicacion: WideString;
                          sFecCadu: WideString);
procedure BorrarDetalle(nIdLine: Double);
procedure AnadirTalla(sCodFamTallaH, sCodFamTallaV, sCodTallaH,
                      sCodTallaV: WideString; nUnidades, nPrcMedio: Double);
procedure CambiarTalla(nIdLine, nUnidades, nPrcMedio: Double);
procedure BorrarTalla(nIdLine: Double);
procedure EditarRapido(nInventario: Currency; const nCodart: WideString)
procedure NuevoV2(sFecha: WideString; const sCodAlm: WideString; lTodosArti: WordBool;
                  Valoracion: Integer)
```

Propiedades	Tipo	Descripción
AsStringCab	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsIntegerCab	Tabla(Integer)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsFloatCab	Tabla(Float)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsBooleanCab	Tabla(Lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsVariantCab	Tabla(Variant)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsStringLin	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual.

AsIntegerLin	Tabla(Integer)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsFloatLin	Tabla(Float)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsBooleanLin	Tabla(Lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsCurrencyLin	Tabla(Currency)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsVariantCab	Tabla(Variant)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsStringDes	Tabla(string)	Lee valores de los campos del registro nuevo/actual.
AsIntegerDes	Tabla(Integer)	Lee valores de los campos del registro nuevo/actual.
AsFloatDes	Tabla(Float)	Lee valores de los campos del registro nuevo/actual.
AsBooleanDes	Tabla(Lógico)	Lee valores de los campos del registro nuevo/actual.
AsCurrencyDes	Tabla(Currency)	Lee valores de los campos del registro nuevo/actual.
AsVariantDes	Tabla(Variant)	Lee valores de los campos del registro nuevo/actual.
Estado	EstadoMaestro	Devuelve el estado en que se encuentra el objeto
AvisarStock	Lógico	Asignar valor para activar o no los mensajes de aviso de stock
OmitirMensajes	Lógico	Asignar valor para ocultar mensajes de información

*EstadoMaestro: Ver objeto Maestro.

Método	Tipo	Descripción
Iniciar	Procedimiento	Inicia el objeto
Acabar	Procedimiento	Cierra el objeto
Nuevo	Procedimiento	Inicia un nuevo documento
Borra	Procedimiento	Borra el documento
Modifica	Procedimiento	Prepara el documento para poderlo modificar.
Cancela	Procedimiento	Cancela la edición del documento.
Anade	Función	Añade los datos editados del documento a la base de datos y devuelve el identificador con el que se almacenará.
NuevaLinea	Procedimiento	Inicia una nueva línea
NuevaLineaArt	Procedimiento	Inicia una nueva línea aplicando las políticas de nexus, dados el código del artículo y las unidades pasadas como parámetro.
AnadirLinea	Procedimiento	Añade la línea
CancelaLin	Procedimiento	Cancela la línea en edición
EditarLinea	Procedimiento	Permite editar una línea ya existente del documento.
BorrarLinea	Procedimiento	Permite borrar una línea del documento.
AnadirDetalle	Procedimiento	Identifica los nº de serie, lotes, fechas de caducidad y/o

		ubicaciones de una línea del documento.
CambiarDetalle	Procedimiento	Cambia los nº de serie, lotes, fechas de caducidad y/o ubicaciones de una línea del documento.
BorrarDetalle	Procedimiento	Borra el detalle los nº de serie, lotes, fechas de caducidad y/o ubicaciones de una línea del documento.
AnadirTalla	Procedimiento	Identifica las tallas de una línea del documento.
CambiarTalla	Procedimiento	Cambia las tallas de una línea del documento.
BorrarTalla	Procedimiento	Borra el detalle de las tallas de una línea del documento.
EditarRapido	Procedimiento	Pone en edición un artículo concreto del inventario
NuevoV2	Procedimiento	Crea un Nuevo inventario utilizando el criterio de valoración que se le pasa por parámetro.

Las primeras funciones de la lista que se presentan a continuación se utilizan para **asignar / leer valores de la cabecera o de las líneas del documento.**

Deben tenerse en cuenta algunas normas importantes.

- Si el nombre del campo no existe se producirá un error.
- Si el valor asignado al campo especificado no es correcto, no se producirá un error. Este se producirá al intentar guardar el documento o la línea.

Hay que tener en cuenta lo siguiente:

- Los campos fecha se han de asignar como cadenas de texto como '01/01/2003'
- Uso en Delphi: a3ERPACTIVEXDocumento.AsStringLin['CodArt']
- Uso en Visual Basic: a3ERPACTIVEXDocumento. AsStringLin ('CodArt')

Cuando se crea un inventario nuevo, el programa necesita crear una línea en ese inventario para almacenar las unidades de stock que debería tener cada artículo en cada almacén. (Se almacena esa información en el campo UNICALC). Posteriormente el usuario puede cambiar esa "propuesta" por la realidad del almacén (llenando la columna que se corresponde con el campo UNIDADES).

Por tanto, al crear un inventario puede tardar un rato, en el que se estará realizando el cálculo del stock propuesto por la aplicación.

property AsStringCab[sCampo:String]: String		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerCab[sCampo:String]: Integer		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatCab[sCampo:String]: Double		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanCab[sCampo:String]: Boolean		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es lógico.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsStringLin[sCampo:String]: String		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerLin[sCampo:String]: Integer		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatLin[sCampo:String]: Double		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanLin[sCampo:String]: Boolean		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es lógico.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsCurrencyLin[sCampo:String]: Currency		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es currency.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsStringDes[sCampo:String]: String [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerDes[sCampo:String]: Integer [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatDes[sCampo:String]: Double [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanDes[sCampo:String]: Boolean [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es lógico.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsCurrencyDes[sCampo:String]: Currency [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es currency.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsVariantDes[sCampo:String]: OleVariant [SOLO LECTURA]		
A través de esta propiedad podemos leer el valor de los campos de la línea de desglose especificando que su tipo es variant.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Variant	Valor del campo

property AvisarStock: WordBool		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es lógico.		
Valor retornado	Currency	Valor del campo

A continuación, se presentan las propiedades y métodos necesarios para crear, borrar y modificar inventarios.

property Estado:EstadoMaestro		
Esta propiedad nos indica el estado del objeto. Posibles estados: ESTM_NOACTIVO: Documento cerrado. No se puede usar. ESTM_ACTIVO: Documento activo. Podemos añadir, borrar, modificar. ESTM_NUEVO: Documento en estado de inserción de un nuevo registro. ESTM_EDICION: Documento en estado modificación, podemos asignar valores a los campos.		
Valor retornado	EstadoMaestro	Retorna el estado del documento.

procedure Iniciar;
Este procedimiento es el que reserva los recursos necesarios para que el documento pueda ser usado. Al iniciar el documento el estado cambia de ESTM_NOACTIVO a ESTM_ACTIVO.

procedure Acabar;
Este procedimiento es el que libera los recursos utilizados por el objeto. Al acabar el documento el estado cambia de ESTM_ACTIVO a ESTM_NOACTIVO.

procedure Nuevo(sFecha, sCodAlm: String; bTodosArti: WordBool);		
Inserta un nuevo registro y los prepara para ser editado. Pasa de estado ESTM_ACTIVO a ESTM_NUEVO.		
Parámetros:	sFecha	Fecha del documento

	sCodAlm	Código del almacén
	bTodosArti	bTodosArti = True: Se incluyen los artículos con stock 0. bTodosArti = True: Sólo se incluyen los artículos cuyo stock es diferente de 0.

procedure Nuevo2(sFecha, sCodAlm: String; bTodosArti: WordBool; Valoracion: Integer);		
Inserta un nuevo registro y los prepara para ser editado. Pasa de estado ESTM_ACTIVO a ESTM_NUEVO.		
Parámetros:	sFecha	Fecha del documento
	sCodAlm	Código del almacén
	bTodosArti	bTodosArti = True: Se incluyen los artículos con stock 0. bTodosArti = True: Sólo se incluyen los artículos cuyo stock es diferente de 0.
	Valoracion	Criterio de valoración. Valores posibles: 1: Manual. 2: Precio medio. 3: Precio de coste. 4: Precio estándar. 5: Precio de última compra 6: Precio ponderado no continuo

procedure Modifica(nIdDocu: Currency);		
Pone el documento que tiene el identificador indicado en edición.		
Parámetros:	nIdDocu	Identificador del documento

Procedure Anade;
Almacena los valores de los campos en la base de dato con los cambios introducidos por el programador. Genera las repercusiones del documento como pueden ser: actualización de stocks y actualización de estadísticas. Pasa de los estados ESTM_NUEVO o EST_EDICION a ESTM_ACTIVO.

Procedure Cancela;
Cancela la edición del documento. Pasa de los estados ESTM_NUEVO o EST_EDICION a ESTM_ACTIVO.

procedure Borra(nIdDocu: Currency);
Elimina el documento con el identificador indicado eliminando también sus repercusiones en stocks, contabilidad, cartera, estadísticas e IVA.

Parámetros:	nIdDocu	Identificador del documento

Procedure NuevaLinea;
Crea una línea con los valores por defecto. A partir de ese momento podremos editar los valores de la línea.

Procedure NuevaLineaArt(sCodArt: string; nUnidades: Currency);		
Crea una línea con el artículo y las unidades indicadas. A partir de ese momento podremos editar los valores de la línea.		
Parámetros:	sCodArt	Código del artículo
	nUnidades	Unidades

Procedure AnadirLinea;
Almacena los valores de los campos de la línea en la base de datos.

Procedure CancelaLin;
Cancela la edición de la línea activa.

procedure EditarLinea(nIdLinea: Currency);		
Pone la línea indicada en el parámetro nIdLinea en edición, preparada para editar el contenido de sus campos. Al final de las modificaciones deberá llamarse a AnadirLinea.		
Parámetros:	nIdLinea	Identificador de la línea. Se corresponde con el campo IDINV del fichero LINEINVE. Para localizarlo debe realizarse una consulta de este campo (directamente a la base de datos) en los registros con IdInv = a3ERPACTIVEXInve.AsStringCab['IdInv'].

procedure BorrarLinea(nIdLinea: Currency);		
Borra la línea con el identificador indicado en el parámetro nIdLinea.		
Parámetros:	nIdLinea	Identificador de la línea. Se corresponde con el campo NUMLININV del fichero LINEINVE. Para localizarlo debe realizarse una consulta de este campo (directamente a la base de datos) en los registros con IdInv = a3ERPACTIVEXInve.AsStringCab['IdInv'].

procedure EditarRapido(nInventario: Currency; const nCodart: WideString);		
Pone en edición un artículo concreto del inventario, sin necesidad de cargar todo el inventario. No necesita hacer un 'Editar' del inventario previamente.		
Parámetros:	nInventario	Identificador del inventario. Se corresponde con el campo IDINV de la tabla CABEINVE.
	nCodart	Código de artículo a modificar.

Los procedimientos que se describen a continuación se utilizan **para especificar los lotes, nº de serie, fechas de caducidad y ubicaciones de una línea** a no ser que esta selección se realice de forma automática. Para consultar si ya se han detallado todos los lotes, nº de serie, fechas de caducidad y ubicaciones necesarias se debe llamar a la función ConsultarDetalle;

procedure AnadirDetalle(nUnidades: Double; nPrcMedio: Double; sNumSerie: WideString; sLote: WideString; sUbicacion: WideString; sFecCadu: WideString);		
Indica las unidades necesarias y su precio medio, nº serie, lote, fecha de caducidad y ubicación de la línea activa. Para conocer los nº serie, lote, fecha de caducidad y ubicaciones disponibles en stock hay que llamar a la función **Varios.DetalleStockDisponible**..		
Parámetros:	nNunidades	Nº de unidades de la línea de detalle.
	nPrcMedio	Precio medio.
	sNumSerie	Nº de serie.
	sLote	Lote.
	sUbicacion	Ubicación.
	sFecCadu	Fecha de caducidad.

procedure CambiarDetalle(nIdLine: Double; sNumSerie: WideString; sLote: WideString; sUbicacion: WideString; sFecCadu: WideString);		
Indica las unidades necesarias y su precio medio, Nº serie, lote, fecha de caducidad y ubicación de la línea activa. Para conocer los nº serie, lote, fecha de caducidad y ubicaciones disponibles en stock hay que llamar a la función **Varios.DetalleStockDisponible**. Para consultar el detalle de la línea activa hay que llamar a la función **ConsultarDetalle**.		
Parámetros:	nIdLine	Identificador de la línea. Se corresponde con el campo IDLIN de la tabla LINEINVE.
	nUnidades	Nº de unidades de la línea de detalle.
	nPrcMedio	Precio medio.
	sNumSerie	Nº de serie.
	sLote	Lote.
	sUbicacion	Ubicación.
	sFecCadu	Fecha de caducidad.

procedure BorrarDetalle(nIdLine: Double);		
Elimina la línea de detalle indicada en el parámetro.		
Parámetros:	nIdLine	Identificador de la línea. Se corresponde con el campo IDLIN de la tabla LINEINVE.

procedure AnadirTalla(sCodFamTallaH, sCodFamTallaV, sCodTallaH, sCodTallaV: WideString; nUnidades, nPrcMedio:Double);		
Indica las unidades y el precio necesarios, junto con sus tallas y colores de la línea activa.		
Parámetros:	sCodFamTallaH	Familia de talla horizontal
	sCodFamTallaV	Familia de talla vertical
	sCodTallaH	Código de talla horizontal
	sCodTallaV	Código de talla vertical
	nUnidades	Nº unidades de la línea de talla
	nPrcMedio	Precio medio de la línea de talla.

procedure CambiarTalla(nIdLine, nUnidades, nPrcMedio: Double);		
Indica las unidades y el precio de la línea activa.		
Parámetros:	nIdLine	Identificador de la línea. Se corresponde con el campo IDLIN de la tabla LINEINVE.
	nUnidades	Nº de unidades de la línea de talla.
	nPrcMedio	Precio medio de la línea de talla.

procedure BorrarTalla(nIdLine: Double);		
Elimina la línea de talla indicada en el parámetro.		
Parámetros:	nIdLine	Identificador de la línea. Se corresponde con el campo IDLIN de la tabla LINEINVE.

21.- OBJETO MOVIMIENTOSTOCK

Objeto que permite la **creación, modificación y borrado de conjuntos de movimientos de stock**.

Interface MovimientoStock
property Estado: EstadoMaestro readonly; **property** AsStringLin[const sCampo: WideString]: WideString; **property** AsFloatLin[const sCampo: WideString]: Double; **property** AsIntegerLin[const sCampo: WideString]: Integer; **property** AsBooleanLin[const sCampo: WideString]: WordBool; **property** AsCurrencyLin[const sCampo: WideString]: Currency; **property** AsVariantLin[const sCampo: WideString]: OleVariant dispid 203; **property** AvisarStock: WordBool; **property** ActivarAvisos: WordBool; **procedure** Iniciar; **procedure** Acabar; **procedure** Cancela;

```
procedure Nuevo(  sFecha: WideString; sCodAlm: WideString);
procedure Borra(IdDocu: Currency);
procedure Modifica(IdDocu: Currency);
function Anade: Currency;
procedure NuevaLinea;
procedure AnadirLinea;
procedure EditarLinea(nIdLinea: Currency);
procedure BorrarLinea(nIdLinea: Currency);
procedure CancelaLin;
```

Propiedades	Tipo	Descripción
AsStringLin	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsIntegerLin	Tabla(Integer)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsFloatLin	Tabla(Float)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsBooleanLin	Tabla(Lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsCurrencyLin	Tabla(Currency)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsVariantLin	Tabla(Variant)	Asigna/lee valores a/de los campos del registro nuevo/actual.
Estado	EstadoMaestro	Devuelve el estado en que se encuentra el objeto

*EstadoMaestro: Ver objeto Maestro.

Metodo	Tipo	Descripción
Iniciar	Procedimiento	Inicia el objeto
Acabar	Procedimiento	Cierra el objeto
Nuevo	Procedimiento	Inicia un nuevo documento
Borra	Procedimiento	Borra el docuemtno
Modifica	Procedimiento	Prepara el documento para poderlo modificar.
Cancela	Procedimiento	Cancela la edición del documento.
Anade	Función	Añade los datos editados del documento a la base de datos y devuelve el identificador con el que se almacenará.
NuevaLinea	Procedimiento	Inicia una nueva línea
AnadirLinea	Procedimiento	Añade la línea
CancelaLin	Procedimiento	Cancela la línea en edición
EditarLinea	Procedimiento	Permite editar una línea ya existente del documento.

BorrarLinea	Procedimiento	Permite borrar una línea del documento.
AvisarStock	Lógico	Asignar valor para activar o no los mensajes de aviso de stock
ActivarAvisos	Lógico	Asignar valor para activar o no otros mensajes de aviso

Las primeras funciones de la lista que se presentan a continuación se utilizan para asignar / leer valores de la cabecera o de las líneas del documento.

Deben tenerse en cuenta algunas normas importantes:

- Si el nombre del campo no existe se producirá un error.
- Si el valor asignado al campo especificado no es correcto, no se producirá un error. Este se producirá al intentar guardar el documento o la línea.

Hay que tener en cuenta lo siguiente:

- Los campos fecha se han de asignar como cadenas de texto como '01/01/2003'
- Uso en Delphi: a3ERPACTIVEXDocumento.AsStringLin['CodArt']
- Uso en Visual Basic: a3ERPACTIVEXDocumento. AsStringLin ('CodArt')

property AsStringLin[sCampo:String]: String		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerLin[sCampo:String]: Integer		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatLin[sCampo:String]: Double		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanLin[sCampo:String]: Boolean		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es lógico.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsCurrencyLin[sCampo:String]: Currency		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es currency.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsVariantLin[sCampo:String]: Boolean		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es variant.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Variant	Valor del campo

A continuación, se presentan las propiedades y métodos necesarios para **crear, borrar y modificar movimientos de stock.**

property Estado:EstadoMaestro		
Esta propiedad nos indica el estado del objeto. Posibles estados: ESTM_NOACTIVO: Documento cerrado. No se puede usar. ESTM_ACTIVO: Documento activo. Podemos añadir, borrar, modificar. ESTM_NUEVO: Documento en estado de inserción de un nuevo registro. ESTM_EDICION: Documento en estado modificación, podemos asignar valores a los campos.		
Valor retornado	EstadoMaestro	Retorna el estado del documento.

procedure Iniciar;
Este procedimiento es el que reserva los recursos necesarios para que el documento pueda ser usado. Al iniciar el documento el estado cambia de ESTM_NOACTIVO a ESTM_ACTIVO.

procedure Acabar;
Este procedimiento es el que libera los recursos utilizados por el objeto. Al acabar el documento el estado cambia de ESTM_ACTIVO a ESTM_NOACTIVO.

procedure Nuevo(sFecha: String; sCodAlm: String);		
Inserta un nuevo registro y los prepara para ser editado. Pasa de estado ESTM_ACTIVO a ESTM_NUEVO.		
Parámetros:	sFecha	Fecha del documento
	sCodAlm	Código del almacén.

procedure Modifica(nIdDocu: Currency);		
Pone el documento que tiene el identificador indicado en edición.		
Parámetros:	nIdDocu	Identificador del documento

Procedure Anade;
Almacena los valores de los campos en la base de dato con los cambios introducidos por el programador. Genera las repercusiones del documento como pueden ser: actualización de stocks y actualización de estadísticas. Pasa de los estados ESTM_NUEVO o EST_EDICION a ESTM_ACTIVO.

Procedure Cancela;
Cancela la edición del documento. Pasa de los estados ESTM_NUEVO o EST_EDICION a ESTM_ACTIVO.

procedure Borra(nIdDocu: Currency);		
Elimina el documento con el identificador indicado eliminando también sus repercusiones en stocks, contabilidad, cartera, estadísticas e IVA.		
Parámetros:	nIdDocu	Identificador del documento

Procedure NuevaLinea;
Crea una línea con los valores por defecto. A partir de ese momento podremos editar los valores de la línea.

Procedure AnadirLinea;
Almacena los valores de los campos de la línea en la base de datos.

Procedure CancelaLin;
Cancela la edición de la línea activa.

procedure EditarLinea(nIdLinea: Currency);		
Pone la línea indicada en el parámetro nIdLinea en edición, preparada para editar el contenido de sus campos. Al final de las modificaciones deberá llamarse a AnadirLinea.		
Parámetros:	nIdLinea	Identificador de la línea. Se corresponde con el campo NUMLINMOV del fichero LINEMOVI. Para localizarlo debe realizarse una consulta de este campo (directamente a la base de datos) en los registros con IdMov = a3ERPACTIVEXMovi.AsStringCab['IdMov'].

procedure BorrarLinea(nIdLinea: Currency);		
Borra la línea con el identificador indicado en el parámetro nIdLinea.		
Parámetros:	nIdLinea	Identificador de la línea. Se corresponde con el campo NUMLINMOV del fichero LINEMOVI. Para localizarlo debe realizarse una consulta de este campo (directamente a la base de datos) en los registros con IdMov = a3ERPACTIVEXMovi.AsStringCab['IdMov'].

22.- OBJETO RESERVASTOCK

Objeto que permite la **creación, modificación y borrado de conjuntos de reservas de stock**.

Interface ReservaStock
property Estado: EstadoMaestro readonly; **property** AsStringLin[const sCampo: WideString]: WideString; **property** AsFloatLin[const sCampo: WideString]: Double; **property** AsIntegerLin[const sCampo: WideString]: Integer; **property** AsBooleanLin[const sCampo: WideString]: WordBool; **property** AsCurrencyLin[const sCampo: WideString]: Currency; **property** AsVariantLin[const sCampo: WideString]: OleVariant; **property** ActivarAvisos: WordBool; **procedure** Iniciar; **procedure** Acabar; **procedure** Cancela; **procedure** Nuevo(sFecha: WideString; sFecEntrega: WideString; sCodAlm: WideString); **procedure** Borra(IdDocu: Currency); **procedure** Modifica(IdDocu: Currency); **function** Anade: Currency; **procedure** NuevaLinea;

```
procedure AnadirLinea;
procedure CancelaLin;
procedure EditarLinea(nIdLinea: Currency);
procedure BorrarLinea(nIdLinea: Currency);
```

Propiedades	Tipo	Descripción
AsStringLin	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsIntegerLin	Tabla(Integer)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsFloatLin	Tabla(Float)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsBooleanLin	Tabla(Lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsCurrencyLin	Tabla(Currency)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsVariantLin	Tabla(Variant)	Asigna/lee valores a/de los campos del registro nuevo/actual.
Estado	EstadoMaestro	Devuelve el estado en que se encuentra el objeto
ActivarAvisos	Lógico	Asignar valor para activar o no mensajes de aviso

*EstadoMaestro: Ver objeto Maestro.

Método	Tipo	Descripción
Iniciar	Procedimiento	Inicia el objeto
Acabar	Procedimiento	Cierra el objeto
Nuevo	Procedimiento	Inicia un nuevo documento
Borra	Procedimiento	Borra el documento
Modifica	Procedimiento	Prepara el documento para poderlo modificar.
Cancela	Procedimiento	Cancela la edición del documento.
Anade	Función	Añade los datos editados del documento a la base de datos y devuelve el identificador con el que se almacenará.
NuevaLinea	Procedimiento	Inicia una nueva línea
AnadirLinea	Procedimiento	Añade la línea
CancelaLin	Procedimiento	Cancela la línea en edición
EditarLinea	Procedimiento	Permite editar una línea ya existente del documento.
BorrarLinea	Procedimiento	Permite borrar una línea del documento.

Las primeras funciones de la lista que se presentan a continuación se utilizan para **asignar / leer valores de la cabecera o de las líneas del documento.**

Deben tenerse en cuenta algunas normas importantes:

- Si el nombre del campo no existe se producirá un error.
- Si el valor asignado al campo especificado no es correcto, no se producirá un error. Este se producirá al intentar guardar el documento o la línea.

Hay que tener en cuenta lo siguiente:

- Los campos fecha se han de asignar como cadenas de texto como '01/01/2003'
- Uso en Delphi: a3ERPACTIVEXDocumento.AsStringLin['CodArt']
- Uso en Visual Basic: a3ERPACTIVEXDocumento. AsStringLin ('CodArt')

property AsStringLin[sCampo:String]: String		
A esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerLin[sCampo:String]: Integer		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatLin[sCampo:String]: Double		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanLin[sCampo:String]: Boolean		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es lógico.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsCurrencyLin[sCampo:String]: Currency		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es currency.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsVariantLin[sCampo:String]: Variant		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es currency.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Variant	Valor del campo

A continuación, se presentan las propiedades y métodos necesarios para **crear, borrar y modificar reservas de stock.**

property Estado:EstadoMaestro		
Esta propiedad nos indica el estado del objeto. Posibles estados: ESTM_NOACTIVO: Documento cerrado. No se puede usar. ESTM_ACTIVO: Documento activo. Podemos añadir, borrar, modificar. ESTM_NUEVO: Documento en estado de inserción de un nuevo registro. ESTM_EDICION: Documento en estado modificación, podemos asignar valores a los campos.		
Valor retornado	EstadoMaestro	Retorna el estado del documento.

procedure Iniciar;
Este procedimiento es el que reserva los recursos necesarios para que el documento pueda ser usado. Al iniciar el documento el estado cambia de ESTM_NOACTIVO a ESTM_ACTIVO.

procedure Acabar;
Este procedimiento es el que libera los recursos utilizados por el objeto. Al acabar el documento el estado cambia de ESTM_ACTIVO a ESTM_NOACTIVO.

procedure Nuevo(sFecha: String; sFecEnt: String; sCodAlm: String);
Inserta un nuevo registro y los prepara para ser editado. Pasa de estado ESTM_ACTIVO a ESTM_NUEVO.

Parámetros:	sFecha	Fecha de la reserva
	sFecEnt	Fecha de entrega de la reserva
	sCodAlm	Código del almacén.

procedure Modifica(nIdDocu: Currency);		
Pone el documento que tiene el identificador indicado en edición.		
Parámetros:	nIdDocu	Identificador del documento

Procedure Anade;
Almacena los valores de los campos en la base de dato con los cambios introducidos por el programador. Genera las repercusiones del documento como pueden ser: actualización de stocks y actualización de estadísticas. Pasa de los estados ESTM_NUEVO o EST_EDICION a ESTM_ACTIVO.

Procedure Cancela;
Cancela la edición del documento. Pasa de los estados ESTM_NUEVO o EST_EDICION a ESTM_ACTIVO.

procedure Borra(nIdDocu: Currency);		
Elimina el documento con el identificador indicado eliminando también sus repercusiones en stocks, contabilidad, cartera, estadísticas e IVA.		
Parámetros:	nIdDocu	Identificador del documento

Procedure NuevaLinea;
Crea una línea con los valores por defecto. A partir de ese momento podremos editar los valores de la línea.

Procedure AnadirLinea;
Almacena los valores de los campos de la línea en la base de datos.

Procedure CancelaLin;
Cancela la edición de la línea activa.

procedure EditarLinea(nIdLinea: Currency);
Pone la línea indicada en el parámetro nIdLinea en edición, preparada para editar el contenido de sus campos. Al final de las modificaciones deberá llamarse a AnadirLinea.

Parámetros:	nIdLinea	Identificador de la línea.. Se corresponde con el campo NUMLINRES del fichero LINERESE. Para localizarlo debe realizarse una consulta de este campo (directamente a la base de datos) en los registros con IdRes = a3ERPACTIVEXReserva.AsStringCab['IdRes'].

procedure BorrarLinea(nIdLinea: Currency);		
Borra la línea con el identificador indicado en el parámetro nIdLinea.		
Parámetros:	nIdLinea	Identificador de la línea.. Se corresponde con el campo NUMLINRES del fichero LINERESE. Para localizarlo debe realizarse una consulta de este campo (directamente a la base de datos) en los registros con IdRes = a3ERPACTIVEXReserva.AsStringCab['IdRes'].

23.- OBJETO ASIENTO1404

Interface Asiento1404

```
property Estado: EstadoMaestro readonly;
property AsString[const sCampo: WideString]: WideString;
property AsFloat[const sCampo: WideString]: Double;
property AsInteger[const sCampo: WideString]: Integer;
property AsBoolean[const sCampo: WideString]: WordBool;
propery AsVariant[const sCampo: WideString]: OleVariant;
property TratarConcepto: WordBool;
property Importado: WordBool;
procedure Iniciar;
procedure Acabar;
procedure Nuevo(sFecha: WideString; sTipoContable: WideString; sTipo: WideString);
procedure Modifica( nAsiento: Currency);
procedure Borra( nAsiento: Currency);
function Anade: Currency;
procedure Cancela;
procedure AApunte( sCodMon: WideString; sCuenta: WideString; nDebe: Currency;
                   sDescripcion: WideString; nHaber: Currency; sCentroCoste:WideString;
                   sTexto: WideString);
procedure NuevoNAX( sTipApu: WideString; sTipoCont: WideString; sDiario: WideString;
                    sNumDoc: WideString; sFecha: WideString);
procedure AApunteNAX( sCodConce: WideString; sDescApu: WideString;
                      sCentroCoste: WideString; sCentroCoste2: WideString;
                      sCentroCoste3: WideString; sCodMon: WideString; sCuenta: WideString;
                      nDebeMoneda: Currency; nDebe: Currency; nHaberMoneda: Currency;
```

```
                    nHaber: Currency; sTexto: WideString; sFechaValor: WideString);
procedure Desligar (nAsiento: Currency);
procedure PuntearAsiento(NumApunte: Currency; const CampoPunteo: WideString; Valor:
WordBool);
```

Propiedades	Tipo	Descripción
Estado	EstadoMaestro	Devuelve el estado en que se encuentra el objeto Asiento.
AsString	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsFloat	Tabla(float)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsInteger	Tabla(integer)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsBoolean	Tabla(lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
TratarConcepto	Lógico	Para desactivar la petición visual de variables de conceptos
Importado	Lógico	Para desactivar los mensajes visuales de nº asiento o asiento cuadrado, etc

Métodos	Tipo	Descripción
Iniciar	Procedimiento	Inicia el objeto asiento.
Acabar	Procedimiento	Cierra el objeto asiento.
Nuevo	Procedimiento	Pasa al ESTM_NUEVO para comenzar a añadir apuntes.
Modifica	Procedimiento	Sustituye el asiento especificado por el actual.
Borra	Procedimiento	Borra el asiento especificado.
Anade	Función	Utiliza los apuntes introducidos para crear un nuevo asiento. Devuelve el número del asiento que se acaba de generar.
Cancela	Procedimiento	Cancela el nuevo, pasa a ESTM_ACTIVO y se pierden los apuntes introducidos.
AApunte	Procedimiento	Añade un apunte al asiento actual.
NuevoNAX	Procedimiento	Equivalente a Nuevo pero con más parámetros.
AApunteNAX	Procedimiento	Equivalente a Aapunte pero con más parámetros.
Desligar	Procedimiento	Para desvincular el asiento de su cartera. Se indica el nº de asiento a desligar
PuntearAsiento	Procedimiento	Permite marcar/desmarcar cualquiera de los campos de punteo (PUNTEADO1..5) de un asiento completo.

Las primeras funciones de la lista que se presentan a continuación se utilizan para **asignar / leer valores del asiento**. Deben tenerse en cuenta algunas normas importantes:

- Si el nombre del campo no existe se producirá un error.
- Si el valor asignado al campo especificado no es correcto, no se producirá un error. Este se producirá al intentar añadir la línea.

Hay que tener en cuenta lo siguiente:

- Los campos fecha se han de asignar como cadenas de texto como '1/1/2002'
- Uso en Delphi: a3ERPACTIVEXApunte.AsString['Cuenta']
- Uso en Visual Basic: a3ERPACTIVEXApunte.AsString ('Cuenta')

property AsString[sCampo:String]: String		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsInteger[sCampo:String]: Integer		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloat[sCampo:String]: Double		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBoolean[sCampo:String]: Boolean		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es lógico.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsVariant[sCampo:String]: OleVariant		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es Variant.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	OleVariant	Valor del campo

property Estado:EstadoMaestro		
Esta propiedad nos indica el estado del objeto. Posibles estados: ESTM_NOACTIVO: Asiento cerrado. No se puede usar. ESTM_ACTIVO: Asiento activo. Podemos añadir, borrar, modificar. ESTM_NUEVO: Asiento en estado de inserción de un nuevo registro. ESTM_EDICION: Asiento en estado modificación, podemos asignar valores a los campos.		
Parámetros:	(ninguno)	
Valor retornado	EstadoMaestro	Retorna el estado del asiento.

Property TratarConcepto: Boolean		
Para desactivar la petición visual de variables de conceptos		
Valor retornado	Lógico	Valor del campo

Property Importado: Boolean;		
Para desactivar los mensajes visuales de nº asiento o asiento cuadrado, etc		
Valor retornado	Lógico	Valor del campo

procedure Iniciar;
Este procedimiento es el que reserva los recursos necesarios para que el asiento pueda ser usado. Al iniciar el estado cambia de ESTM_NOACTIVO a ESTM_ACTIVO.

procedure Acabar;
Este procedimiento es el que libera los recursos utilizados por el objeto. Al acabar el estado cambia de ESTM_ACTIVO a ESTM_NOACTIVO.

procedure Nuevo(sFecha, sTipoContable, sTipo: String)		
Mediante este procedimiento se inicia la edición de un nuevo asiento. Como parámetros se pasan los valores que son comunes a todas las líneas del asiento. Pasa del estado ESTM_ACTIVO al estado ESTM_NUEVO.		
Parámetros:	sFecha	Fecha del asiento
	sTipoContable	Tipo contable del asiento.
	STipo	Tipo de asiento. Valores posibles: A → Apertura. N → Normal. P → Paso a explotación. T → Cierre.

procedure NuevoNAX(sTipApu: WideString; sTipoCont: WideString; sDiario: WideString; sNumDoc: WideString; sFecha: WideString);		
Mediante este procedimiento se inicia la edición de un nuevo asiento. Como parámetros se pasan los valores que son comunes a todas las líneas del asiento. Pasa del estado ESTM_ACTIVO al estado ESTM_NUEVO.		
Parámetros:	sTipApu	Tipo de asiento. Valores posibles: A → Apertura. N → Normal. P → Paso a explotación. T → Cierre.
	sTipoCont	Tipo contable del asiento.
	sDiario	Diario contable
	sNumDoc	Número de documento contable.
	sFecha	Fecha del asiento

procedure Modifica(nAsiento: Currency);		
Mediante este procedimiento se inicia la modificación de un asiento existente. Como parámetro se pasa el identificador del asiento. Pasa del estado ESTM_ACTIVO al estado ESTM_EDICION. Debe tenerse en cuenta que cuando se modifica un asiento que ya existe lo primero que hace A3ERPACTIVEX es borrar las líneas que ya tiene y a continuación el programador debe introducir todas las líneas que finalmente tendrá ese asiento.		
Parámetros:	nAsiento	Identificador del asiento. Se corresponde con el valor del campo NUMAPUNTE del fichero APUNTES.

procedure Borra(nAsiento: Currency);		
Mediante este procedimiento se borra un asiento existente. Como parámetro se pasa el identificador del asiento.		
Parámetros:	nAsiento	Identificador del asiento. Se corresponde con el valor del campo NUMAPUNTE del fichero APUNTES.

function Anade: Currency;		
Mediante este procedimiento se termina la edición de un asiento tanto nuevo como existente y se guarda su contenido en la base de datos. Cambia el estado de ESTM_NUEVO o ESTM_EDICION a ESTM_ACTIVO.		
Valor retornado:	Currency	Identificador del asiento. Se corresponde con el valor del campo NUMAPUNTE del fichero APUNTES.

Procedure Cancela;
Mediante este procedimiento se cancela la edición de un asiento tanto nuevo como existente. Cambia el estado de ESTM_NUEVO o ESTM_EDICION a ESTM_ACTIVO.

procedure AApunte(sCodMon, sCuenta: String; nDebe: Currency; sDescripcion: String; nHaber:Currency; sCentroCoste, sTexto: String);		
Mediante este procedimiento se crean las líneas (apuntes) de un asiento.		
Parámetros:	sCodMon	Código de la moneda.
	sCuenta	Cuenta contable.
	nDebe	Importe del debe en la moneda indicada.
	sDescripcion	Descripción del asiento.
	nHaber	Importe del haber en la moneda indicada.
	sCentroCoste	Centro de coste 1.
	sTexto	Texto.

procedure Desligar(nAsiento: Currency);		
Mediante este procedimiento podemos desvincular el asiento de su cartera		
Parámetros:	nAsiento	Identificador del asiento. Se corresponde con el valor del campo NUMAPUNTE del fichero APUNTES.

procedure AApunteNAX(sCodConce: WideString; sDescApu: WideString; sCentroCoste: WideString; sCentroCoste2: WideString; sCentroCoste3: WideString; sCodMon: WideString; sCuenta: WideString; nDebeMoneda: Currency; nDebe: Currency; nHaberMoneda: Currency; nHaber: Currency; sTexto: WideString; sFechaValor: WideString);		
Mediante este procedimiento se crean las líneas (apuntes) de un asiento.		
Parámetros:	sCodConce	Concepto del asiento. Únicamente sirve para rellenar la variable CodConce.
	sDescApu	Descripción del asiento.
	sCentroCoste	Centro de coste 1.
	sCentroCoste2	Centro de coste 2.
	sCentroCoste3	Centro de coste 3.
	sCodMon	Código de la moneda.
	sCuenta	Cuenta contable.
	nDebeMoneda	Importe del debe en la moneda indicada.
	nDebe	Importe del debe en la moneda principal. Si el valor es 0 se hace la conversión automáticamente.
	nHaberMoneda	Importe del haber en la moneda indicada.
	nHaber	Importe del haber en la moneda principal. Si el valor es 0 se hace la conversión automáticamente.
	sTexto	Texto.
	sFechaValor	Fecha valor del apunte.

procedure PunteArAsiento(NumApunte: Currency; CampoPunteo: WideString; Valor: WordBool);		
Mediante este procedimiento se puede marcar el asiento compelto como punteado.		
Parámetros:	NumApunte	Identificador del asiento. Se corresponde con el valor del campo NUMAPUNTE del fichero APUNTES.
	CampoPunteo	Campo que se quiere marcar/desmarcar. Puede ser PUNTEADO1, PUNTEADO2, PUNTEADO3, PUNTEADO4, PUNTEADO5.
	Valor	True = el campo se marca como punteado, False = el campo se queda como no punteado

24.- OBJETO ASIENTO

Interface Asiento(Asiento1404) <Nuevo en versión 14.05>
procedure ModificarTextoAsiento(nAsiento: Currency; const Texto: WideString); safecall;
procedure ModificarParametrizablesApunte(nAsiento: Currency; NumLinea: Integer; const Param1, Param2, Param3, Param4, Param5, Param6, Param7, Param8, Param9: WideString); safecall;

procedure ModificarTextoAsiento(nAsiento: Currency; const Texto: WideString)		
Mediante este procedimiento podemos modificar el texto del asiento		
Parámetros:	nAsiento	Identificador del asiento. Se corresponde con el valor del campo NUMAPUNTE del fichero APUNTES.
	Texto	Texto que se va a informar en la línea del primer apunte del asiento.

procedure ModificarParametrizablesApunte(nAsiento: Currency; NumLinea: Integer; const Param1, Param2, Param3, Param4, Param5, Param6, Param7, Param8, Param9: WideString); safecall;		
Mediante este procedimiento podemos modificar los parametrizables de un apunte de un asiento		
Parámetros:	nAsiento	Identificador del asiento. Se corresponde con el valor del campo NUMAPUNTE del fichero APUNTES.
	NumLinea	Número de la línea del apunte cuyos parametrizables queremos modificar. Corresponde con el valor del campo NUMLIN de la tabla de ASIENTOS
	Param1	Primer parametrizable
	Param2	Segundo parametrizable
	Param3	Tercer parametrizable
	Param4	Cuarto parametrizable
	Param5	Quinto parametrizable
	Param6	Sexto parametrizable
	Param7	Séptimo parametrizable
	Param8	Octavo parametrizable
	Param9	Noveno parametrizable

25.- OBJETO OPERACIONESINMOVILIZADO

Interface OperacionesInmovilizado
procedure Amortizar(sCodigo: WideString; sFecha: WideString); **procedure** Contabilizar(sCodigo: WideString; sFecha: WideString); **procedure** AnularAmortizacion(sCodigo: WideString; sFecha: WideString); **procedure** AnularContabilizacion(sCodigo: WideString; sFecha: WideString); **procedure** BajaElemento(sCodigo: WideString; sFecha: WideString; nValor: Currency; sMotivo: WideString); **procedure** Cancelar(sCodigo: WideString; sFecha: WideString; nValor: Currency; sMotivo: WideString); **procedure** AnularCancelar(sCodigo: WideString);

Métodos	Tipo	Descripción
Amortizar	Procedimiento	Amortiza una cuota determinada.
Contabilizar	Procedimiento	Genera las repercusiones contables de una cuota determinada.
AnularAmortizacion	Procedimiento	Anula la amortización ya realizada.
AnularContabilizacion	Procedimiento	Anula la contabilización ya realizada.
Cancelar	Procedimiento	Cancela la amortización de un inmovilizado. A partir del momento en que se ejecute este procedimiento no se amortizarán más cuotas.
AnularCancelar	Procedimiento	Anula la cancelación realizada mediante el procedimiento Cancelar.

procedure Amortizar(sCodigo: WideString; sFecha: WideString);		
Amortiza una cuota determinada.		
Parámetros:	sCodigo	Código del inmovilizado.
	sFecha	Fecha de la cuota a amortizar.

procedure Contabilizar(sCodigo: WideString; sFecha: WideString);		
Genera las repercusiones contables de una cuota determinada.		
Parámetros:	sCodigo	Código del inmovilizado.
	sFecha	Fecha de la cuota a contabilizar.

procedure AnularAmortizacion(sCodigo: WideString; sFecha: WideString);		
Anula la amortización ya realizada.		
Parámetros:	sCodigo	Código del inmovilizado.
	sFecha	Fecha de la cuota.

procedure AnularContabilizacion(sCodigo: WideString; sFecha: WideString);		
Anula la contabilización ya realizada.		
Parámetros:	sCodigo	Código del inmovilizado.
	sFecha	Fecha de la cuota.

procedure BajaElemento(sCodigo: WideString; sFecha: WideString; nValor: Currency; sMotivo: WideString);		
Cancela la amortización de un inmovilizado. A partir del momento en que se ejecute este procedimiento no se amortizarán más cuotas.		
Parámetros:	sCodigo	Código del inmovilizado.
	sFecha	Fecha de la cancelación
	nValor	Valor residual del inmovilizado.
	sMotivo	Motivo de la baja.

procedure AnularCancelar(sCodigo: WideString);		
Anula la cancelación del inmovilizado.		
Parámetros:	sCodigo	Código del inmovilizado.

26.- OBJETO CARTERA

Interface Cartera

```
property AsString[const sCampo: WideString]: WideString ;
property AsFloat[const sCampo: WideString]: Double;
property AsInteger[const sCampo: WideString]: Integer;
property AsBoolean[const sCampo: WideString]: WordBool;
property AsCurrency[const sCampo: WideString]: Currency;
property AsVariant[const sCampo: WideString]: Variant ;
 property VAsString[const sCampo: WideString]: WideString ;
 property VAsFloat[const sCampo: WideString]: Double;
 property VAsInteger[const sCampo: WideString]: Integer;
 property VAsBoolean[const sCampo: WideString]: WordBool;
 property VAsCurrency[const sCampo: WideString]: Currency;
property AsVariant[const sCampo: WideString]: Variant ;
property Estado: EstadoMaestro readonly;
property OmitirMensajes: Boolean;
procedure Iniciar;
procedure Nuevo( sFecha: WideString; sCodCliPro: WideString; sCodMon: WideString;
                 sTipoCont: WideString; sSerie: WideString; sNumDoc: WideString;
                 EsDeCompra: WordBool);
```

```
procedure NuevoNAX( sFecha: WideString; sCodCliPro: WideString; sCodMon: WideString;
                    sTipoCont: WideString; sSerie: WideString; sNumDoc: WideString;
                    sReferencia:WideString; EsDeCompra: WordBool);
function Anade: Currency;
procedure Cancela;
procedure NuevoVen(const sFecha: WideString; nImporteMon: Currency);
procedure Acabar;
procedure Modifica(nCartera: Currency);
procedure AnadirVen;
procedure Borra(nNumCart: Currency);
function ObtenerNumCartera(bComVen: WordBool; IdFac: Double): Double;
function PuedeModificarse(nNumCart: Currency): WordBool;
procedure Editar(nNumCart:Currency);
procedure EditarVencimiento(nNumVen:integer);
procedure BorrarVencimiento(nNumVen:integer);
procedure GuardarVencimiento;
```

Propiedades	Tipo	Descripción
Estado	EstadoMaestro	Devuelve el estado en que se encuentra el objeto Cartera.
AsString	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual. (cartera)
AsFloat	Tabla(float)	Asigna/lee valores a/de los campos del registro nuevo/actual. (cartera)
AsInteger	Tabla(integer)	Asigna/lee valores a/de los campos del registro nuevo/actual. (cartera)
AsBoolean	Tabla(lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual. (cartera)
AsCurrency	Tabla(Currency)	Asigna/lee valores a/de los campos del registro nuevo/actual. (cartera)
AsVariant	Tabla(Variant)	Asigna/lee valores a/de los campos del registro nuevo/actual. (cartera)
VAsString	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual. (vencimiento)
VAsFloat	Tabla(float)	Asigna/lee valores a/de los campos del registro nuevo/actual. (vencimiento)
VAsInteger	Tabla(integer)	Asigna/lee valores a/de los campos del registro nuevo/actual. (vencimiento)
VAsBoolean	Tabla(lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual. (vencimiento)
VAsCurrency	Tabla(Currency)	Asigna/lee valores a/de los campos del registro nuevo/actual. (vencimiento)
VAsVariant	Tabla(Variant)	Asigna/lee valores a/de los campos del registro nuevo/actual. (cartera)

OmitirMensajes	Lógico	Asignar valor para ocultar mensajes de información

property Estado:EstadoMaestro		
Esta propiedad nos indica el estado del objeto. Posibles estados: ESTM_NOACTIVO: Cartera (conjunto de vencimientos) cerrada. No se puede usar. ESTM_ACTIVO: Cartera activa. Podemos añadir, borrar, modificar. ESTM_NUEVO: Cartera en estado de inserción de un nuevo registro. ESTM_EDICION: Cartera en estado modificación, podemos asignar valores a los campos.		
Parámetros:	(ninguno)	
Valor retornado	EstadoMaestro	Retorna el estado de la cartera (conjunto de vencimientos).

procedure Iniciar;
Este procedimiento es el que reserva los recursos necesarios para que la certera pueda ser usada. Al iniciar el estado cambia de ESTM_NOACTIVO a ESTM_ACTIVO.

procedure Acabar;
Este procedimiento es el que libera los recursos utilizados por el objeto. Al acabar el estado cambia de ESTM_ACTIVO a ESTM_NOACTIVO.

Procedure Nuevo(sFechaFactura, sCodCliPro, sCodMon, sTipoCont, sSerie, sNumDoc: string; bEsDeCompra: boolean);		
Con esta función preparamos el objeto para insertar nuevos vencimientos de una factura. Los parámetros que se le pasan al procedimiento indican valores por defecto para todos los vencimientos y si se trata de una cartera asociada a un cliente (cobro) o a un proveedor (pago).		
Parámetros:	sFechaFactura	Fecha de la factura
	sCodCliPro	Código del cliente o proveedor
	sCodMon	Código de la moneda
	sTipoCont	Tipo contable
	sSerie	Serie de la factura
	sNumDoc	Nº de la factura
	bEsDeCompra	bEsDeCompra = True → Factura de compra bEsDeCompra = False → Factura de venta

Procedure NuevoNAX(sFechaFactura, sCodCliPro, sCodMon, sTipoCont, sSerie, SNumDoc, sReferencia : string; bEsDeCompra: boolean);
Con esta función preparamos el objeto para insertar nuevos vencimientos de una factura. Los parámetros que se le pasan al procedimiento indican valores por defecto para todos los vencimientos y si se trata de una cartera asociada a un cliente (cobro) o a un proveedor (pago).

Parámetros:	sFechaFactura	Fecha de la factura
	sCodCliPro	Código del cliente o proveedor
	sCodMon	Código de la moneda
	sTipoCont	Tipo contable
	sSerie	Serie de la factura
	SNumDoc	Nº de la factura
	SReferencia	Referencia de la factura
	BEsDeCompra	bEsDeCompra = True → Factura de compra bEsDeCompra = False → Factura de venta

Procedure Borra(nNumCartera: currency);		
Borra todos los efectos que pertenecen a un conjunto de cartera. Prerrequisito: No debe existir ningún vencimiento cobrado, pagado, recibido, bloqueado, etc... Para consultar si la cartera de una factura puede borrarse utilizar la función **PuedeModificarse**.		
Parámetros:	nNumCartera	Identificador del conjunto de cartera de una factura. Se corresponde con el campo NUMCARTERA de la tabla CARTERA.

Procedure Modifica(nNumCartera: currrency);		
Prepara un conjunto de cartera para modificar sus efectos. Lo primero que hace es borrar los que existen y el programador debe crearlos de nuevo. Prerrequisito: No debe existir ningún vencimiento cobrado, pagado, recibido, bloqueado, etc... Para consultar si la cartera de una factura puede borrarse utilizar la función **PuedeModificarse**.		
Parámetros:	nNumCartera	Identificador del conjunto de cartera de una factura. Se corresponde con el campo NUMCARTERA de la tabla CARTERA.

procedure Cancela;
Cancela la edición de vencimientos eliminando todas las operaciones realizadas desde que se ha llamado a Nuevo o Modifica. Pasa de un estado ESTM_NUEVO o ESTM_EDICION a un estado ESTM_ACTIVO.

function Anade: Currency;		
Mediante este procedimiento se termina la edición de un conjunto de cartera tanto nuevo como existente y se guarda su contenido en la base de datos. Cambia el estado de ESTM_NUEVO o ESTM_EDICION a ESTM_ACTIVO.		
Valor retornado:	Currency	Identificador del conjunto de cartera de una factura. Se corresponde con el campo NUMCARTERA de la tabla CARTERA.

function PuedeModificarse(nNumCart: Currency): WordBool;		
Función que indica si puede modificarse o no un conjunto de cartera. Para que un conjunto de cartera pueda ser modificado o borrado no deben haberse realizado operaciones sobre ninguno de sus vencimientos.		
Parámetros:	nNumCart	Nº del conjunto de cartera.
Valor retornado:	Boolean	Indica si puede modificarse o no.

procedure ObtenerNumCartera(bEsCobro: boolean; nIdFac: Currency): Currency;		
Función que permite obtener el identificador del conjunto de vencimientos (NUMCARTERA) de una factura.		
Parámetros:	bEsCobro	Indica si la factura es de compra (false) o de venta (true).
	nIdFac	bComVen = True → Identificador de la factura de venta (Se corresponde con el campo IDFACV de la tabla CABEFACV). bComVen = False → Identificador de la factura de compra (Se corresponde con el campo IDFACC de la tabla CABEFACC).
Valor retornado:	currency	Nº del conjunto de cartera.

procedure NuevoVen(sFecha: WideString; nImporteMon: Currency);		
Crea un nuevo efecto con la fecha (vencimiento) e importe indicados.		
Parámetros:	sFecha	Fecha del vencimiento.
	nImporteMon	Importe en la moneda indicada en el procedimiento **Nuevo**.

procedure AnadirVen;
Crea los valores del vencimiento que se acaba de crear en la base de datos.

procedure Editar(nNumCartera:Currency);		
Prepara los vencimientos de un NUMCARTERA para su posterior modificación o borrado		
Parámetros:	nNumCartera	Número de cartera

procedure EditarVencimiento(nNumVen:Currency);		
Prepara el vencimiento para su posterior modificación (NUMVEN).		
Parámetros:	nNumVen	Número de vencimiento

procedure GuardarVencimiento;
Guarda el vencimiento que está en modo edición

procedure BorrarVencimiento(nNumVen:Currency);		
Borrar un vencimiento indicado (NUMVEN)		
Parámetros:	nNumVen	Número de vencimiento

27.- OBJETO OPERACIONESCARTERA

Interface OperacionesCartera

```
property OmitirAvisoAsiento:WordBool;
property OmitirMensajes: WordBool;
procedure Cobrar( nNumCart: Currency; nNumVen: Integer; bGenRep: WordBool;
                  bDescue: WordBool;  sNumCta: WideString; sCtaRiesgo: WideString;
                  sFecha: WideString; sCodBan: WideString);
procedure Pagar( nNumCart: Currency; nNumVen: Integer; bGenRep: WordBool;
                 sNumCta: WideString; sFecha: WideString; sCodBan: WideString);
procedure Recibir( nNumCart: Currency; nNumVen: Integer; bGenRep: WordBool;
                 sFecha: WideString);
procedure CrearAnticipo (bEsCobro: WordBool; sCodCliPro: WideString; sCodBan:
                  WideString; sDocPag: WideString; sTipCon: WideString; sCodMon: WideString;
                  sFecVen: WideString; const sFecCon: WideString;  nImpMon: Currency;
                  Obs: WideString);
procedure ACuenta ( bEsCobro: WordBool; nIdFact: Currency; nNumAnti: Currency;
                    sCodMon: WideString; nImpACuentaMon: Currency); Obsoleto a partir de 9.6.6
procedure CrearAnticipoNAX ( bEsCobro: WordBool; sCodCliPro: WideString;
                  sCodBan: WideString; sDocPag: WideString; sTipCon: WideString;
                   sCodMon: WideString; sFecVen: WideString; sFecCon: WideString;
                    nImpMon: Currency; Obs: WideString; sNumDoc: WideString);
procedure Bloquear(nNumCart: Currency; nNumVen: Integer);
procedure Enviar( nNumCart: Currency; nNumVen: Integer; bGenRep: WordBool;
                 sFecha: WideString; sFechaValor: WideString; sNumDoc: WideString);
procedure ActualizarRiesgo( nNumCart: Currency; nNumVen: Integer; bGenRep:
                WordBool;  sFecha: WideString; sNumDoc: WideString);
procedure Devolver( nNumCart: Currency; nNumVen: Integer; bGenRep: WordBool;
                  sFecha: WideString; sFechaValor: WideString; nImpDev: currency,
                 nGastos: Currency;   sCtaGastos: WideString; sCentro1: WideString;
                  sCentro2: WideString;   sCentro3: WideString; sNumDoc: WideString);
procedure DevolverNAX (nNumCart: Currency; nNumVen: Integer; bGenRep: WordBool;
                    const sFecha: WideString; const sFechaValor: WideString;
```

```
                    nImpDev: Currency; nGastos: Currency; const sCtaGastos: WideString;
                    const sCtaComision: WideString; nComisiones: Currency;
                    const sCentro1: WideString; const sCentro2: WideString;
                    const sCentro3: WideString; const sNumDoc: WideString;
                    contabilizarGastos: WordBool);
procedure ImputarGastos( nNumCart: Currency; nNumVen: Integer; bGenRep: WordBool;
                        sFecha: WideString; sFechaValor: WideString; nGastos: Currency;
                        sCtaGastos: WideString; sCentro1: WideString; sCentro2: WideString;
                        sCentro3: WideString; sNumDoc: WideString);
procedure RecibirNAX ( nNumCart: Currency; nNumVen: Integer; bGenRep: WordBool;
                    sFecha: WideString; sFechaValor: WideString; sNumDoc: WideString);
procedure CobrarNAX ( nNumCart: Currency; nNumVen: Integer; bGenRep: WordBool;
                        bDescue: WordBool; nImporte: Currency; sNumCta: WideString;
                    sCtaRiesgo: WideString; sFecha: WideString; sFechaValor: WideString;
                    sCodBan: WideString;  nCambio: Double; sCtaGastos: WideString;
                    nGastos: Currency; sCentro1: WideString; sCentro2: WideString;
                    sCentro3: WideString; sNumDoc: WideString);
procedure PagarNAX ( nNumCart: Currency; nNumVen: Integer; bGenRep: WordBool;
                  nImporte: Currency; sNumCta: WideString; sFecha: WideString;
                  sFechaValor: WideString; sCodBan: WideString; nCambio: Double;
                  sCtaGastos: WideString; nGastos: Currency; sCentro1: WideString;
                  sCentro2: WideString; sCentro3: WideString; sNumDoc: WideString);
procedure AnularCobro(nNumCart: Currency; nNumVen: Integer);
procedure AnularPago(nNumCart: Currency; nNumVen: Integer);
procedure AnularRecepcion(nNumCart: Currency; nNumVen: Integer);
procedure AnularEnvio(nNumCart: Currency; nNumVen: Integer);
procedure AnularBloqueo(nNumCart: Currency; nNumVen: Integer);
procedure AnularImputacion(nNumCart: Currency; nNumVen: Integer);
procedure AnularDevolucion(nNumCart: Currency; nNumVen: Integer);
procedure AnularActualizacionRiesgo (nNumCart: Currency; nNumVen: Integer);
procedure Ligar(nNumCart: Currency; nIdFact: Currency; bEsVenta: WordBool);
procedure Desligar( nIdFact: Currency; bEsCobro: WordBool);
procedure CobrarRemesa( nRemesa: Currency; GenRep: WordBool;
                bAlDescuento: WordBool;   sCtaBanco: WideString; sCtaRiesgo: WideString;
                sFecha: WideString;    sFechaValor: WideString; sCodBan: WideString;
                nCambio: Double;    sCtaGastos: WideString; nGastos: Currency;
                sCentro1: WideString;    sCentro2: WideString; sCentro3: WideString;
                 sNumDoc: WideString);
procedure AnularCobroRemesa(nRemesa: Currency);
procedure PagarRemesa ( nRemesa: Currency; bGenRep: WordBool;
                  sCtaBanco: WideString;   sFecha: WideString; sFechaValor: WideString;
                  sCodBan: WideString;   nCambio: Double; sCtaGastos: WideString;
                 nGastos: Currency;   sCentro1: WideString; sCentro2: WideString;
                 sCentro3: WideString;   sNumDoc: WideString);
```

```
procedure AnularPagoRemesa( nRemesa: Currency);
procedure DevolverConComis( nNumCart: Currency; nNumVen: Integer;
                bGenRep:   WordBool;  sFecha: WideString; sFechaValor: WideString;
                nImpDev:  Currency; nGastos: Currency; const sCtaGastos: WideString;
                 sCtaComisiones:  WideString; nComisiones: Currency;  sCentro1: WideString;
                 sCentro2: WideString;   sCentro3: WideString; sNumDoc: WideString);
procedure CobrarNAXConComis(nNumCart: Currency; nNumVen: Integer;
                 bGenRep: WordBool;  bDescue: WordBool; nImporte: Currency;
                sNumCta: WideString; CtaRiesgo: WideString; sFecha: WideString;
                 sFechaValor: WideString; sCodBan: WideString;  nCambio: Double;
                 sCtaGastos: WideString; nGastos: Currency; sCtaComisiones: WideString;
                 nComisiones: Currency;sCentro1: WideString; const sCentro2: WideString;
                 sCentro3: WideString; sNumDoc: WideString);
procedure PagarNAXConComis(nNumCart: Currency; nNumVen: Integer;
                 bGenRep: WordBool; nImporte: Currency; sNumCta: WideString;
                 sFecha: WideString; sFechaValor: WideString;  sCodBan: WideString;
                nCambio: Double;sCtaGastos: WideString; nGastos: Currency;
                 sCtaComisiones: WideString; nComisiones: Currency; sCentro1: WideString;
                 sCentro2: WideString; sCentro3: WideString; const sNumDoc: WideString)
procedure PagarRemesaConComis(nRemesa: Currency; bGenRep: WordBool;
                sCtaBanco: WideString; sFecha: WideString;  sFechaValor: WideString;
                sCodBan: WideString;   nCambio: Double; sCtaGastos: WideString;
                nGastos: Currency; sCtaComisiones: WideString; nComisiones: Currency;
              sCentro1: WideString; const sCentro2: WideString;  sCentro3: WideString;
              sNumDoc: WideString);
procedure CobrarRemesaConComis(nRemesa: Currency; bGenRep: WordBool
                        ; bAlDescuento: WordBool;  sCtaBanco: WideString;
                        sCtaRiesgo: WideString;  sFecha: WideString;
                        sFechaValor: WideString;  sCodBan: WideString; nCambio: Double;
                         sCtaGastos: WideString; nGastos: Currency;
                         sCtaComisiones: WideString; nComisiones: Currency;
                          sCentro1: WideString; sCentro2: WideString;
                          sCentro3: WideString; sNumDoc: WideString);
procedure AnularAnticipo(nNumCart: Currency; nNumVen: Integer); Nuevo 14.0
procedure DudosoCobro(NumCar: Currency; NumVen: Integer;
                           const Fecha, Centro1, Centro2, Centro3: WideString;
                           Importe: Currency; const Documento: WideString;
                           GenRepContables: WordBool); Nuevo 14.0
procedure AnularDudosoCobro(NumCar: Currency; NumVen: Integer); Nuevo 14.0
procedure Impagar(NumCar: Currency; NumVen: Integer; const Fecha,
                    Centro1, Centro2, Centro3: WideString;
                    Importe: Currency; const Documento: WideString;
                    GenRepContables: WordBool); Nuevo 14.0
procedure AnularImpagar(NumCar: Currency; NumVen: Integer); Nuevo 14.0
```

El **objeto Cartera permitía crear, borrar o modificar vencimientos**. El objeto Operaciones cartera permite realizar distintas operaciones con estos vencimientos. A continuación, se detalla la especificación de todas esas operaciones.

procedure Cobrar(nNumCart: Currency; nNumVen: Integer; bGenRep: WordBool; bDescue: WordBool; sNumCta: WideString; sCtaRiesgo: WideString; sFecha: WideString; sCodBan: WideString);		
Procedimiento creado para cobrar un efecto de nexus generando las repercusiones contables si así se indica en los parámetros.		
Parámetros:	nNumCart	Numero del conjunto de cartera. Campo NUMCARTERA de la tabla CARTERA.
	nNumVen	Número del vencimiento. Campo NUMVEN de la tabla CARTERA.
	bGenRep	Indica si se desean generar repercusiones contables.
	bDescue	Indica si se trata de un cobro al descuento.
	sNumCta	Indica el nº de cuenta del banco (Si se deja en blanco se saca del banco indicado en el parámetro sCodBan).
	sCtaRiesgo	Indica el nº de cuenta de deudas por efectos al descuento del banco (Si se deja en blanco se saca del banco indicado en el parámetro sCodBan).
	sFecha	Fecha del cobro.
	sCodBan	Código del banco.

procedure CobrarNAX(nNumCart: Currency; nNumVen: Integer; bGenRep: WordBool; bDescue: WordBool; nImporte: Currency; sNumCta: WideString; sCtaRiesgo: WideString; sFecha: WideString; sFechaValor: WideString; sCodBan: WideString; nCambio: Double; sCtaGastos: WideString; nGastos: Currency; sCentro1: WideString; sCentro2: WideString; sCentro3: WideString; sNumDoc: WideString);		
Procedimiento creado para cobrar un efecto de nexus generando las repercusiones contables si así se indica en los parámetros. Funcionamiento igual que **Cobrar** pero con más parámetros.		
Parámetros:	nNumCart	Numero del conjunto de cartera. Campo NUMCARTERA de la tabla CARTERA.
	nNumVen	Número del vencimiento. Campo NUMVEN de la tabla CARTERA.
	bGenRep	Indica si se desean generar repercusiones contables.
	bDescue	Indica si se trata de un cobro al descuento.

	nImporte	Importe que se cobra. No tiene que coincidir con el del vencimiento (tiene que ser menor o igual). Si no coincide lo tratará como un cobro parcial.
	sNumCta	Indica el nº de cuenta del banco (Si se deja en blanco se saca del banco indicado en el parámetro sCodBan).
	sCtaRiesgo	Indica el nº de cuenta de deudas por efectos al descuento del banco (Si se deja en blanco se saca del banco indicado en el parámetro sCodBan).
	sFecha	Fecha del cobro.
	sFechaValor	Fecha valor del cobro.
	sCodBan	Código del banco.
	nCambio	Cambio entre la moneda del cobro y la moneda principal.
	sCtaGastos	Cuenta de gastos del cobro.
	nGastos	Gastos imputados al cobro. (Se incluirán en el asiento).
	sCentro1	Centro de coste 1.
	sCentro2	Centro de coste 2.
	sCentro3	Centro de coste 3.
	sNumDoc	Número de documento del asiento.

procedure Pagar(nNumCart: Currency; nNumVen: Integer; bGenRep: WordBool; sNumCta: WideString; sFecha: WideString; sCodBan: WideString);		
Procedimiento creado para pagar un efecto de nexus generando las repercusiones contables si así se indica en los parámetros.		
Parámetros:	nNumCart	Numero del conjunto de cartera. Campo NUMCARTERA de la tabla CARTERA.
	nNumVen	Número del vencimiento. Campo NUMVEN de la tabla CARTERA.
	bGenRep	Indica si se desean generar repercusiones contables.
	sNumCta	Indica el nº de cuenta del banco (Si se deja en blanco se saca del banco indicado en el parámetro sCodBan).
	sFecha	Fecha del pago.
	sCodBan	Código del banco.

procedure PagarNAX(nNumCart: Currency; nNumVen: Integer; bGenRep: WordBool; nImporte: Currency; sNumCta: WideString; sFecha: WideString; sFechaValor: WideString; sCodBan: WideString; nCambio: Double; sCtaGastos: WideString; nGastos: Currency; sCentro1: WideString; sCentro2: WideString; sCentro3: WideString; sNumDoc: WideString);		
Procedimiento creado para pagar un efecto de nexus generando las repercusiones contables si así se indica en los parámetros. Funcionamiento igual que **Pagar** pero con más parámetros.		
Parámetros:	nNumCart	Numero del conjunto de cartera. Campo NUMCARTERA de la tabla CARTERA.
	nNumVen	Número del vencimiento. Campo NUMVEN de la tabla CARTERA.
	bGenRep	Indica si se desean generar repercusiones contables.
	nImporte	Importe que se cobra. No tiene que coincidir con el del vencimiento (tiene que ser menor o igual). Si no coincide lo tratará como un pago parcial.
	sNumCta	Indica el nº de cuenta del banco (Si se deja en blanco se saca del banco indicado en el parámetro sCodBan).
	sFecha	Fecha del pago.
	sFechaValor	Fecha valor del pago.
	sCodBan	Código del banco.
	nCambio	Cambio entre la moneda del pago y la moneda principal.
	sCtaGastos	Cuenta de gastos del pago.
	nGastos	Gastos imputados al pago. (Se incluirán en el asiento).
	sCentro1	Centro de coste 1.
	sCentro2	Centro de coste 2.
	sCentro3	Centro de coste 3.
	sNumDoc	Número de documento del asiento.

procedure Recibir(nNumCart: Currency; nNumVen: Integer; bGenRep: WordBool; sFecha: WideString);		
Procedimiento creado para contabilizar la recepción de un efecto generando las repercusiones contables si así se indica en los parámetros.		
Parámetros:	nNumCart	Número del conjunto de cartera. Campo NUMCARTERA de la tabla CARTERA.
	nNumVen	Número del vencimiento. Campo NUMVEN de la tabla CARTERA.
	bGenRep	Indica si se desean generar repercusiones contables.

	sFecha	Fecha de la recepción..

procedure RecibirNAX(nNumCart: Currency; nNumVen: Integer; bGenRep: WordBool; sFecha: WideString; sFechaValor: WideString; sNumDoc: WideString);		
Procedimiento creado para recibir un efecto generando las repercusiones contables si así se indica en los parámetros. Funcionamiento igual que **Recibir** pero con más parámetros.		
Parámetros:	nNumCart	Numero del conjunto de cartera. Campo NUMCARTERA de la tabla CARTERA.
	nNumVen	Número del vencimiento. Campo NUMVEN de la tabla CARTERA.
	bGenRep	Indica si se desean generar repercusiones contables.
	sFecha	Fecha de la recepción.
	sFechaValor	Fecha valor de la recepción.
	sNumDoc	Número de documento del asiento.

procedure Enviar(nNumCart: Currency; nNumVen: Integer; bGenRep: WordBool; sFecha, sFechaValor, sNumDoc: WideString);		
Procedimiento creado para enviar un efecto generando las repercusiones contables si así se indica en los parámetros.		
Parámetros:	nNumCart	Numero del conjunto de cartera. Campo NUMCARTERA de la tabla CARTERA.
	nNumVen	Número del vencimiento. Campo NUMVEN de la tabla CARTERA.
	bGenRep	Indica si se desean generar repercusiones contables.
	sFecha	Fecha del envío.
	sFechaValor	Fecha valor del envío.
	sNumDoc	Número de documento del asiento.

procedure Bloquear(nNumCart: Currency; nNumVen: Integer);		
Procedimiento creado para bloquear un efecto y evitar así que se realicen operaciones de cartera		
Parámetros:	nNumCart	Numero del conjunto de cartera. Campo NUMCARTERA de la tabla CARTERA.
	nNumVen	Número del vencimiento. Campo NUMVEN de la tabla CARTERA.

procedure ActualizarRiesgo(nNumCart: Currency; nNumVen: Integer; bGenRep: WordBool; sFecha, sNumDoc: WideString);
Procedimiento creado para bloquear un efecto y evitar así que se realicen operaciones de cartera

Parámetros:	nNumCart	Numero del conjunto de cartera. Campo NUMCARTERA de la tabla CARTERA.
	nNumVen	Número del vencimiento. Campo NUMVEN de la tabla CARTERA.
	bConRep	Indicamos si queremos repercusiones contables.
	sFecha	Fecha de la actualización de riesgo.
	sNumDoc	Número de documento contable.

procedure Devolver(nNumCart: Currency; nNumVen: Integer; bGenRep: WordBool; sFecha: WideString; sFechaValor: WideString; nImpDev, nGastos: Currency; sCtaGastos: WideString; sCentro1, sCentro2, sCentro3, sNumDoc: WideString);		
Procedimiento creado para devolver total o parcialmente un efecto de nexus generando las repercusiones contables si así se indica en los parámetros.		
Parámetros:	nNumCart	Numero del conjunto de cartera. Campo NUMCARTERA de la tabla CARTERA.
	nNumVen	Número del vencimiento. Campo NUMVEN de la tabla CARTERA.
	bGenRep	Indica si se desean generar repercusiones contables.
	sFecha	Fecha de la devolución.
	sFechaValor	Fecha valor de la devolución.
	nImpDev	Importe de la devolución. No tiene que coincidir con el del vencimiento (tiene que ser menor o igual). Si no coincide lo tratará como una devolución parcial.
	nGastos	Gastos imputados a la devolución. (Se incluirán en el asiento). Estos gastos se consideran gastos de la empresa y no del cliente.
	sCtaGastos	Cuenta de gastos de la devolución.
	sCentro1	Centro de coste 1.
	sCentro2	Centro de coste 2.
	sCentro3	Centro de coste 3.
	sNumDoc	Número de documento del asiento.

procedure DevolverNAX(nNumCart: Currency; nNumVen: Integer; bGenRep: WordBool; const sFecha: WideString; const sFechaValor: WideString; nImpDev: Currency; nGastos: Currency; const sCtaGastos: WideString; const sCtaComision: WideString; nComisiones: Currency; const sCentro1: WideString; const sCentro2: WideString; const sCentro3: WideString; const sNumDoc: WideString; contabilizarGastos: WordBool);		
Procedimiento creado para devolver total o parcialmente un efecto de nexus generando las repercusiones contables si así se indica en los parámetros. Es idéntico al método Devolver pero dándonos la posibilidad de si queremos o no contabilizar los gastos.		
Parámetros:	nNumCart	Numero del conjunto de cartera. Campo NUMCARTERA de la tabla CARTERA.
	nNumVen	Número del vencimiento. Campo NUMVEN de la tabla CARTERA.
	bGenRep	Indica si se desean generar repercusiones contables.
	sFecha	Fecha de la devolución.
	sFechaValor	Fecha valor de la devolución.
	nImpDev	Importe de la devolución. No tiene que coincidir con el del vencimiento (tiene que ser menor o igual). Si no coincide lo tratará como una devolución parcial.
	nGastos	Gastos imputados a la devolución. (Se incluirán en el asiento). Estos gastos se consideran gastos de la empresa y no del cliente.
	sCtaGastos	Cuenta de gastos de la devolución.
	sCentro1	Centro de coste 1.
	sCentro2	Centro de coste 2.
	sCentro3	Centro de coste 3.
	sNumDoc	Número de documento del asiento.
	contabilizarGastos	Indica si se deben contabilizar los gastos

procedure ImputarGastos(nNumCart: Currency; nNumVen: Integer; bGenRep: WordBool; sFecha, sFechaValor: WideString; nGastos: Currency; sCtaGastos: WideString; sCentro1, sCentro2, sCentro3,sNumDoc: WideString);		
Procedimiento creado para imputar los gastos de la devolución al cliente total o parcialmente generando las repercusiones contables si así se indica en los parámetros.		
Parámetros:	nNumCart	Numero del conjunto de cartera. Campo NUMCARTERA de la tabla CARTERA.
	nNumVen	Número del vencimiento. Campo NUMVEN de la tabla CARTERA.

	bGenRep	Indica si se desean generar repercusiones contables.
	sFecha	Fecha de la imputación de gastos.
	sFechaValor	Fecha valor de la imputación de gastos.
	nGastos	Gastos imputados al cliente (Se incluirán en el asiento).
	sCtaGastos	Cuenta de gastos de la imputación.
	sCentro1	Centro de coste 1.
	sCentro2	Centro de coste 2.
	sCentro3	Centro de coste 3.
	sNumDoc	Número de documento del asiento.

procedure CrearAnticipo (bEsCobro: WordBool; sCodCliPro, sCodBan: WideString; sDocPag, sTipCon,sCodMon, sFecVen: WideString; sFecCon: WideString; nImpMon: Currency; Obs: WideString);		
Procedimiento para crear un anticipo de cliente o a proveedor.		
Parámetros:	bEsCobro	Indica si es de cobro (True) o de pago (False).
	sCodCliPro	Código del cliente o proveedor.
	sCodBan	Código del banco.
	sDocPag	Documento de pago.
	sTipCon	Tipo contable.
	sCodMon	Código de la moneda.
	sFecVen	Fecha del anticipo.
	sFecCon	Fecha de la repercusión contable.
	nImpMon	Importe del anticipo en la moneda indicada.
	Obs	Observaciones del anticipo.

procedure CrearAnticipoNAX (bEsCobro: WordBool; sCodCliPro: WideString; sCodBan, sDocPag, sTipCon:WideString; sCodMon: WideString; sFecVen:WideString; sFecCon: WideString; nImpMon: Currency; Obs: WideString; sNumDoc: WideString);		
Procedimiento para crear un anticipo de cliente o a proveedor. Es el mismo que CrearAnticipo pero con el parámetro adicional del campo nº de documento de los asientos contables.		
Parámetros:	bEsCobro	Indica si es de cobro (True) o de pago (False).
	sCodCliPro	Código del cliente o proveedor.
	sCodBan	Código del banco.
	sTipCon	Tipo contable.
	sCodMon	Código de la moneda.
	sFecVen	Fecha del anticipo.
	sFecCon	Fecha de la repercusión contable.
	nImpMon	Importe del anticipo en la moneda indicada.
	Obs	Observaciones del anticipo.

	sNumDoc	Nº de documento en contabilidad.

procedure ACuenta(bEsCobro: WordBool; nIdFact: Currency; nNumAnti: Currency; sCodMon: WideString; nImpACuentaMon: Currency);		
Este procedimiento está obsoleto, se mantiene la entrada por compatibilidad con versioens anteriores. En adelante, debe utilziarse el método 'Acuenta' del objeto 'Factura'		
Parámetros:	bEsCobro	Indica si es de cobro (True) o de pago (False).
	nIdFact	Identificador de la factura. Se identifica con el campo IDFACV o IDFACC de las tablas CABEFACV o CABEFACC respectivamente en función de si se trata de una venta o una compra.
	nNumAnti	Identificador del anticipo. Se corresponde con el campo NUMCARTERA de la tabla CARTERA.
	sCodMon	Código de la moneda
	nImpACuentaMon	Importe a cuenta en la moneda indicada.

Como es costumbre en a3ERP, todo aquello que puede hacerse también puede anularse. A continuación, se presentan todos los procedimientos que anulan las operaciones descritas hasta el momento.

procedure AnularCobro(nNumCart: Currency; nNumVen: Integer);		
Anula la operación realizada con los procedimientos **Cobrar o CobrarNAX**.		
Parámetros:	nNumCart	Número del conjunto de cartera. Se corresponde con el campo NUMCARTERA de la tabla CARTERA.
	nNumVen	Número del vencimiento. Se corresponde con el campo NUMVEN de la tabla CARTERA.

procedure AnularPago(nNumCart: Currency; nNumVen: Integer);		
Anula la operación realizada con los procedimientos **Pagar o PagarNAX**.		
Parámetros:	nNumCart	Número del conjunto de cartera. Se corresponde con el campo NUMCARTERA de la tabla CARTERA.
	nNumVen	Número del vencimiento. Se corresponde con el campo NUMVEN de la tabla CARTERA.

procedure AnularRecepcion(nNumCart: Currency; nNumVen: Integer);		
Anula la operación realizada con los procedimientos **Recibir o RecibirNAX**.		
Parámetros:	nNumCart	Número del conjunto de cartera. Se corresponde con el campo NUMCARTERA de la tabla CARTERA.
	nNumVen	Número del vencimiento. Se corresponde con el campo NUMVEN de la tabla CARTERA.

procedure AnularEnvio(nNumCart: Currency; nNumVen: Integer);		
Anula la operación realizada con el procedimiento **Enviar**.		
Parámetros:	nNumCart	Número del conjunto de cartera. Se corresponde con el campo NUMCARTERA de la tabla CARTERA.
	nNumVen	Número del vencimiento. Se corresponde con el campo NUMVEN de la tabla CARTERA.

procedure AnularBloqueo(nNumCart: Currency; nNumVen: Integer);		
Anula la operación realizada con el procedimiento **Bloquear**.		
Parámetros:	nNumCart	Número del conjunto de cartera. Se corresponde con el campo NUMCARTERA de la tabla CARTERA.
	nNumVen	Número del vencimiento. Se corresponde con el campo NUMVEN de la tabla CARTERA.

procedure AnularDevolucion(nNumCart: Currency; nNumVen: Integer);		
Anula la operación realizada con el procedimiento **Devolver**.		
Parámetros:	nNumCart	Número del conjunto de cartera. Se corresponde con el campo NUMCARTERA de la tabla CARTERA.
	nNumVen	Número del vencimiento. Se corresponde con el campo NUMVEN de la tabla CARTERA.

procedure AnularImputacion(nNumCart: Currency; nNumVen: Integer);		
Anula la operación realizada con el procedimiento **ImputarGastos**.		
Parámetros:	nNumCart	Número del conjunto de cartera. Se corresponde con el campo NUMCARTERA de la tabla CARTERA.
	nNumVen	Número del vencimiento. Se corresponde con el campo NUMVEN de la tabla CARTERA.

procedure AnularActualizacionRiesgo(nNumCart: Currency; nNumVen: Integer);		
Anula la operación realizada con el procedimiento **ActualizarRiesgo**.		
Parámetros:	nNumCart	Número del conjunto de cartera. Se corresponde con el campo NUMCARTERA de la tabla CARTERA.
	nNumVen	Número del vencimiento. Se corresponde con el campo NUMVEN de la tabla CARTERA.

Las dos funciones que siguen a continuación son las que **ligan o desligan la cartera con su factura**. Si los efectos están ligados con una factura se consideran automáticos, en caso contrario no.

Si los efectos no están ligados con la factura, al modificar la factura no se tocarán los vencimientos, en caso contrario se modificarán en función de las nuevas condiciones establecidas en la cabecera del documento.

procedure Ligar(nNumCart: Currency; nIdFact: Currency; bEsCobro: WordBool);		
Ligar un conjunto de efectos de cartera con una factura.		
Parámetros:	nNumCart	Número del conjunto de cartera. Se corresponde con el campo NUMCARTERA de la tabla CARTERA.
	nIdFact	Identificador de la factura de compra (IDFACC) o de venta (IDFACV) pertenecientes a las tablas CABEFACC y CABEFACV respectivamente en función del valor del parámetro **bEsCobro**.
	bEsCobro	Indica si se trata de un cobro (True) o un pago (False).

procedure Desligar(nIdFact: Currency; bEsCobro: WordBool);		
Desliga la factura del conjunto de efectos en cartera con el que esté ligado en ese momento.		
Parámetros:	nIdFact	Identificador de la factura de compra (IDFACC) o de venta (IDFACV) pertenecientes a las tablas CABEFACC y CABEFACV respectivamente en función del valor del parámetro **bEsCobro**.
	bEsCobro	Indica si se trata de un cobro (True) o un pago (False).

Los siguientes procedimientos son los que permiten **realizar y anular operaciones con las remesas tanto de cobros como de pagos.**

procedure CobrarRemesa(nRemesa: Currency; bGenRep: WordBool; bAlDescuento: WordBool; sCtaBanco: WideString; sCtaRiesgo, sFecha, sFechaValor, sCodBan: WideString; nCambio: Double; sCtaGastos: WideString; nGastos: Currency; sCentro1:WideString; sCentro2: WideString; sCentro3: WideString; sNumDoc: WideString);		
Procedimiento que permite cobrar una remesa entera.		
Parámetros:	nRemesa	Nº de remesa
	bGenRep	Indica si se desean repercusiones contables.
	bAlDescuento	Indica si la remesa se lleva al descuento.
	sCtaBanco	Cuenta del banco donde se ingresa la remesa. Si se deja en blanco se utiliza la cuenta del banco indicado en el parámetro **sCodBan**.
	sCtaRiesgo	Cuenta de riesgo bancario. Si se deja en blanco se utiliza la cuenta del banco indicado en el parámetro **sCodBan**.
	sFecha	Fecha de la remesa.
	sFechaValor	Fecha valor de la remesa.
	sCodBan	Código del banco.
	nCambio	Cambio entre la moneda de la remesa y la principal.
	sCtaGastos	Cuenta de gestos de la remesa.
	nGastos	Gastos de la remesa.
	sCentro1	Centro de coste 1.
	sCentro2	Centro de coste 2.

	sCentro3	Centro de coste 3.
	sNumDoc	Número del documento contable.

procedure PagarRemesa(nRemesa: Currency; bGenRep: WordBool; sCtaBanco, sFecha, sFechaValor, sCodBan: WideString; nCambio: Double; sCtaGastos: WideString; nGastos: Currency; sCentro1, sCentro2: WideString; sCentro3: WideString; sNumDoc: WideString);		
Procedimiento que permite pagar una remesa de proveedores.		
Parámetros:	nRemesa	Nº de remesa
	bGenRep	Indica si se desean repercusiones contables.
	sCtaBanco	Cuenta del banco donde se ingresa la remesa. Si se deja en blanco se utiliza la cuenta del banco indicado en el parámetro **sCodBan**.
	sFecha	Fecha de la remesa.
	sFechaValor	Fecha valor de la remesa.
	sCodBan	Código del banco.
	nCambio	Cambio entre la moneda de la remesa y la principal.
	sCtaGastos	Cuenta de gestos de la remesa.
	nGastos	Gastos de la remesa.
	sCentro1	Centro de coste 1.
	sCentro2	Centro de coste 2.
	sCentro3	Centro de coste 3.
	sNumDoc	Número del documento contable.

procedure AnularCobroRemesa(nRemesa: Currency);		
Procedimiento que permite anular el cobro de une remesa realizado con el procedimiento **CobrarRemesa**.		
Parámetros:	nRemesa	Nº de remesa

procedure AnularPagoRemesa(nRemesa: Currency);		
Procedimiento que permite anular el pago de une remesa de proveedores realizado con el procedimiento **PagarRemesa**.		
Parámetros:	nRemesa	Nº de remesa

procedure CobrarRemesaConComis (nRemesa: Currency; bGenRep: WordBool; bAlDescuento: WordBool; sCtaBanco: WideString; sCtaRiesgo, sFecha, sFechaValor, sCodBan: WideString; nCambio: Double; sCtaGastos: WideString; nGastos: Currency; sCtaComisiones: WideString; nComisiones: Currency; sCentro1:WideString; sCentro2: WideString; sCentro3: WideString; sNumDoc: WideString);		
Procedimiento que permite cobrar una remesa entera.		
Parámetros:	nRemesa	Nº de remesa

	bGenRep	Indica si se desean repercusiones contables.
	bAlDescuento	Indica si la remesa se lleva al descuento.
	sCtaBanco	Cuenta del banco donde se ingresa la remesa. Si se deja en blanco se utiliza la cuenta del banco indicado en el parámetro **sCodBan**.
	sCtaRiesgo	Cuenta de riesgo bancario. Si se deja en blanco se utiliza la cuenta del banco indicado en el parámetro **sCodBan**.
	sFecha	Fecha de la remesa.
	sFechaValor	Fecha valor de la remesa.
	sCodBan	Código del banco.
	nCambio	Cambio entre la moneda de la remesa y la principal.
	sCtaGastos	Cuenta de gestos de la remesa.
	sCtaComisiones	Cuenta de comisión de la remesa
	nComisiones	Importe de la comisión
	nGastos	Gastos de la remesa.
	sCentro1	Centro de coste 1.
	sCentro2	Centro de coste 2.
	sCentro3	Centro de coste 3.
	sNumDoc	Número del documento contable.

procedure PagarRemesaConComis(nRemesa: Currency; bGenRep: WordBool; sCtaBanco, sFecha, sFechaValor, sCodBan: WideString; nCambio: Double; sCtaGastos: WideString; nGastos: Currency; sCtaComisiones: WideString; nComisiones: Currency; sCentro1, sCentro2, sCentro3, sNumDoc: WideString);		
Procedimiento que permite pagar una remesa de proveedores.		
Parámetros:	nRemesa	Nº de remesa
	bGenRep	Indica si se desean repercusiones contables.
	sCtaBanco	Cuenta del banco donde se ingresa la remesa. Si se deja en blanco se utiliza la cuenta del banco indicado en el parámetro **sCodBan**.
	sFecha	Fecha de la remesa.
	sFechaValor	Fecha valor de la remesa.
	sCodBan	Código del banco.
	nCambio	Cambio entre la moneda de la remesa y la principal.
	sCtaGastos	Cuenta de gastos de la remesa.
	nGastos	Gastos de la remesa.
	sCtaComisiones	Cuenta de comisión de la remesa
	nComisiones	Importe de la comisión
	sCentro1	Centro de coste 1.

	sCentro2	Centro de coste 2.
	sCentro3	Centro de coste 3.
	sNumDoc	Número del documento contable.

procedure DevolverConcomis(nNumCart: Currency; nNumVen: Integer; bGenRep: WordBool; sFecha: WideString; sFechaValor: WideString; nImpDev, nGastos: Currency; sCtaGastos, sCtaComisiones: WideString; nComisiones:Currency; sCentro1, sCentro2, sCentro3, sNumDoc: WideString);		
Procedimiento creado para devolver total o parcialmente un efecto de nexus generando las repercusiones contables si así se indica en los parámetros.		
Parámetros:	nNumCart	Numero del conjunto de cartera. Campo NUMCARTERA de la tabla CARTERA.
	nNumVen	Número del vencimiento. Campo NUMVEN de la tabla CARTERA.
	bGenRep	Indica si se desean generar repercusiones contables.
	sFecha	Fecha de la devolución.
	sFechaValor	Fecha valor de la devolución.
	nImpDev	Importe de la devolución. No tiene que coincidir con el del vencimiento (tiene que ser menor o igual). Si no coincide lo tratará como una devolución parcial.
	nGastos	Gastos imputados a la devolución. (Se incluirán en el asiento). Estos gastos se consideran gastos de la empresa y no del cliente.
	sCtaGastos	Cuenta de gastos de la devolución.
	sCtaComisiones	Cuenta de comisión de la devolución
	nComisiones	Importe de la comisión
	sCentro1	Centro de coste 1.
	sCentro2	Centro de coste 2.
	sCentro3	Centro de coste 3.
	sNumDoc	Número de documento del asiento.

procedure CobrarNAXConComis (nNumCart: Currency; nNumVen: Integer; bGenRep: WordBool; bDescue: WordBool; nImporte: Currency; sNumCta: WideString; sCtaRiesgo: WideString; sFecha: WideString; sFechaValor: WideString; sCodBan: WideString; nCambio: Double; sCtaGastos: WideString; nGastos: Currency; sCtaComisiones: WideString; nComisiones: Currency; sCentro1: WideString; sCentro2: WideString; sCentro3: WideString; sNumDoc: WideString);		
Procedimiento creado para cobrar un efecto de nexus generando las repercusiones contables si así se indica en los parámetros. Funcionamiento igual que **Cobrar** pero con más parámetros.		
Parámetros:	nNumCart	Numero del conjunto de cartera. Campo NUMCARTERA de la tabla CARTERA.

	nNumVen	Número del vencimiento. Campo NUMVEN de la tabla CARTERA.
	bGenRep	Indica si se desean generar repercusiones contables.
	bDescue	Indica si se trata de un cobro al descuento.
	nImporte	Importe que se cobra. No tiene que coincidir con el del vencimiento (tiene que ser menor o igual). Si no coincide lo tratará como un cobro parcial.
	sNumCta	Indica el nº de cuenta del banco (Si se deja en blanco se saca del banco indicado en el parámetro sCodBan).
	sCtaRiesgo	Indica el nº de cuenta de deudas por efectos al descuento del banco (Si se deja en blanco se saca del banco indicado en el parámetro sCodBan).
	sFecha	Fecha del cobro.
	sFechaValor	Fecha valor del cobro.
	sCodBan	Código del banco.
	nCambio	Cambio entre la moneda del cobro y la moneda principal.
	sCtaGastos	Cuenta de gastos del cobro.
	nGastos	Gastos imputados al cobro. (Se incluirán en el asiento).
	sCtaComisiones	Cuenta de comisión del cobro
	nComisiones	Importe de la comisión
	sCentro1	Centro de coste 1.
	sCentro2	Centro de coste 2.
	sCentro3	Centro de coste 3.
	sNumDoc	Número de documento del asiento.

procedure PagarNaxConComis(nNumCart: Currency; nNumVen: Integer; bGenRep: WordBool; nImporte: Currency; sNumCta: WideString; sFecha: WideString; sFechaValor: WideString; sCodBan: WideString; nCambio: Double; sCtaGastos: WideString; nGastos: Currency; sCtaComisiones: WideString; nComisiones: Currency; sCentro1: WideString; sCentro2: WideString; sCentro3: WideString; sNumDoc: WideString);		
Procedimiento creado para pagar un efecto de nexus generando las repercusiones contables si así se indica en los parámetros. Funcionamiento igual que **Pagar** pero con más parámetros.		
Parámetros:	nNumCart	Numero del conjunto de cartera. Campo NUMCARTERA de la tabla CARTERA.
	nNumVen	Número del vencimiento. Campo NUMVEN de la tabla CARTERA.
	bGenRep	Indica si se desean generar repercusiones contables.
	nImporte	Importe que se cobra. No tiene que coincidir con el del vencimiento (tiene que ser menor o igual). Si no coincide lo tratará como un cobro parcial.

	sNumCta	Indica el nº de cuenta del banco (Si se deja en blanco se saca del banco indicado en el parámetro sCodBan).
	sFecha	Fecha del pago.
	sFechaValor	Fecha valor del pago.
	sCodBan	Código del banco.
	nCambio	Cambio entre la moneda del pago y la moneda principal.
	sCtaGastos	Cuenta de gastos del cobro.
	nGastos	Gastos imputados al pago. (Se incluirán en el asiento).
	sCtaComisiones	Cuenta de comisión del pago
	nComisiones	Importe de la comisión
	sCentro1	Centro de coste 1.
	sCentro2	Centro de coste 2.
	sCentro3	Centro de coste 3.
	sNumDoc	Número de documento del asiento.

procedure AnularAnticipo(nNumCart: Currency; nNumVen: Integer);		
Anula la operación realizada con el procedimiento **CrearAnticipo**.		
Parámetros:	nNumCart	Número del conjunto de cartera. Se corresponde con el campo NUMCARTERA de la tabla CARTERA.
	nNumVen	Número del vencimiento. Se corresponde con el campo NUMVEN de la tabla CARTERA.

Los siguientes métodos permiten trabajar con los conceptos de dudoso cobro e impagado.

procedure DudosoCobro(NumCar: Currency; NumVen: Integer; const Fecha, Centro1, Centro2, Centro3: WideString; Importe: Currency; const Documento: WideString; GenRepContables: WordBool); Nuevo 14.0		
Procedimiento que permite considerar un efecto como de dudoso cobro.		
Parámetros:	NumCar	Nº de remesa
	NumVen	Nº de vencimiento
	Fecha	Fecha (en format día/mes/año) del paso a dudoso cobro.
	Centro1	Centro de coste 1
	Centro2	Centro de coste 2
	Centro3	Centro de coste 3
	Importe	Importe del efecto a pasar a dudoso cobro.
	Documento	Nº de documento a asociar al efecto de dudoso cobro.
	GenRepContables	True para generar repercusiones contables, False en caso contrario.

procedure AnularDudosoCobro(NumCar: Currency; NumVen: Integer); Nuevo 14.0		
Procedimiento que permite anular el paso a dudoso cobro de un efecto de cartera.		
Parámetros:	NumCar	Nº de remesa
	NumVen	Nº de vencimiento

procedure Impagar(NumCar: Currency; NumVen: Integer; const Fecha, Centro1, Centro2, Centro3: WideString; Importe: Currency; const Documento: WideString; GenRepContables: WordBool); Nuevo 14.0		
Procedimiento que permite pasar a impagado un efecto de cartera.		
Parámetros:	NumCar	Nº de remesa
	NumVen	Nº de vencimiento
	Fecha	Fecha (en format día/mes/año) del paso a impagado.
	Centro1	Centro de coste 1
	Centro2	Centro de coste 2
	Centro3	Centro de coste 3
	Importe	Importe del efecto a impagar.
	Documento	Nº de documento a asociar al efecto de impago.
	GenRepContables	True para generar repercusiones contables, False en caso contrario.

procedure AnularImpagar(NumCar: Currency; NumVen: Integer); Nuevo 14.0		
Procedimiento que permite anular el impago de un efecto de cartera.		
Parámetros:	NumCar	Nº de remesa
	NumVen	Nº de vencimiento

28.- OBJETO AGRUPACION

Interface Agrupación
property Estado: EstadoMaestro **readonly**; **puproperty** OmitirMensajes: WordBool; **procedure** Iniciar; **procedure** Acabar; **procedure** Nueva(bEsCobro: WordBool; sFecha, sFecVen, sCodCliPro, sDocPag: WideString; sCodBan, sCodMon, sTipoCont: WideString); **procedure** Modifica(nNumAgru: Currency); **procedure** AnadirEfecto(nNumCart: Currency; nNumVen: Integer);

procedure QuitarEfecto(nNumCart: Currency; nNumVen: Integer);
function Anade: Currency;
procedure Cancela;
procedure Borrar(nNumAgru: Currency);

property Referencia: WideString read Get_Referencia write Set_Referencia; (Nuevo 14.04.00)

property Fecha: DateTime read Get_Fecha write Set_Fecha; (Nuevo 14.04.00)

property CodClienteProveedor: WideString read Get_CodClienteProveedor write Set_CodClienteProveedor; (Nuevo 14.04.00)

property CodDocPago: WideString read Get_CodDocPago write Set_CodDocPago; (Nuevo 14.04.00)

property CodBanco: WideString read Get_CodBanco write Set_CodBanco; (Nuevo 14.04.00)

property CodRepresentante: WideString read Get_CodRepresentante write Set_CodRepresentante; (Nuevo 14.04.00)

property CodFormaPago: WideString read Get_CodFormaPago write Set_CodFormaPago; (Nuevo 14.04.00)

property NumAgrupacion: Currency read Get_NumAgrupacion write Set_NumAgrupacion; (Nuevo 14.04.00)

property Observaciones: WideString read Get_Observaciones write Set_Observaciones; (Nuevo 14.04.00)

property Estado:EstadoMaestro		
Esta propiedad nos indica el estado del objeto. Posibles estados: ESTM_NOACTIVO: Agrupación cerrada. No se puede usar. ESTM_ACTIVO: Agrupación activa. Podemos añadir, borrar, modificar. ESTM_NUEVO: Agrupación en estado de inserción/eliminación de un nuevo efecto en una agrupación nueva. ESTM_EDICION: Agrupación en estado modificación, podemos añadir y quitar efectos en una agrupación que ya existía.		
Parámetros:	(ninguno)	
Valor retornado	EstadoMaestro	Retorna el estado de la agrupación (conjunto de vencimientos).

procedure Iniciar;
Este procedimiento es el que reserva los recursos necesarios para que la agrupación pueda ser usada. Al iniciar el estado cambia de ESTM_NOACTIVO a ESTM_ACTIVO.

procedure Acabar;
Este procedimiento es el que libera los recursos utilizados por el objeto. Al acabar el estado cambia de ESTM_ACTIVO a ESTM_NOACTIVO.

procedure Nueva(bEsCobro: WordBool; sFecha, sFecVen, sCodCliPro: WideString; sDocPag, sCodBan, sCodMon, sTipoCont: WideString);		
Prepara el objeto agrupación para recibir nuevos efectos. El estado cambia de ESTM_ACTIVO a ESTM_NUEVO.		
Parámetros:	bEsCobro	Indica si la agrupación es de cobros (True) o pagos (False).
	sFecha	Fecha en que se crea la agrupación.
	sFechaVen	Vencimiento de la agrupación.
	sCodCliPro	Código del cliente o proveedor.
	sDocPag	Documento de pago.
	sCodBan	Código del banco.
	sCodMon	Código de la moneda.
	sTipoCont	Tipo contable.

Procedure Modifica(nNumAgru: Currency);		
Prepara una agrupación para poder añadir o quitar efectos en ella.		
Parámetros:	nNumAgru	Identificador de la agrupación. Se corresponde con el campo NUMAGRU de la tabla CARTERA.

Procedure Borrar(nNumAgru: Currency);		
Borrar una agrupación.		
Parámetros:	nNumAgru	Identificador de la agrupación. Se corresponde con el campo NUMAGRU de la tabla CARTERA.

procedure AnadirEfecto(nNumCart: Currency; nNumVen: Integer);		
Añade el efecto indicado en la agrupación.		
Parámetros:	nNumCart	Identificador del conjunto de cartera de una factura. Se corresponde con el campo NUMCARTERA de la tabla CARTERA.
	nNumVen	Identificador del conjunto de cartera de una factura. Se corresponde con el campo NUMVEN de la tabla CARTERA.

procedure QuitarEfecto(nNumCart: Currency; nNumVen: Integer);		
Quita el efecto indicado de la agrupación.		
Parámetros:	nNumCart	Identificador del conjunto de cartera de una factura. Se corresponde con el campo NUMCARTERA de la tabla CARTERA.

	nNumVen	Identificador del conjunto de cartera de una factura. Se corresponde con el campo NUMVEN de la tabla CARTERA.

function Anade: Currency;		
Mediante este procedimiento se termina la edición de una agrupación y se guarda su contenido en la base de datos. Cambia el estado de ESTM_NUEVO o ESTM_EDICION a ESTM_ACTIVO.		
Valor retornado:	Currency	Identificador de la agrupación. Se corresponde con el campo NUMAGRU de la tabla CARTERA.

procedure Cancela;
Cancela los efectos que se hayan añadido o quitado en la agrupación desde que se llamó a la función **Nueva** o **Modificar**. Pasa de un estado ESTM_NUEVO o ESTM_EDICION a un estado ESTM_ACTIVO.

property Referencia: WideString; (Nuevo 14.04.00)	
property Referencia: WideString;	Permite acceder y modificar el valor de la referencia de la agrupación.

property Fecha: DateTime; (Nuevo 14.04.00)	
property Fecha: DateTime;	Permite acceder y modificar el valor de la fecha de la agrupación.

property CodClienteProveedor: WideString; (Nuevo 14.04.00)	
property CodClienteProveedor: WideString;	Permite acceder y modificar el valor del código de cliente, en agrupaciones de venta o del proveedor, en agrupaciones de compra.

property CodDocPago: WideString; (Nuevo 14.04.00)	
property CodDocPago: WideString;	Permite acceder y modificar el valor del código del documento de pago de la agrupación.

property CodBanco: WideString; (Nuevo 14.04.00)	
property CodBanco: WideString;	Permite acceder y modificar el valor del código del banco de la agrupación.

property CodRepresentante: WideString; (Nuevo 14.04.00)	
property CodRepresentante: WideString;	Permite acceder y modificar el valor del código del representante de la agrupación.

property CodFormaPago: WideString; (Nuevo 14.04.00)	
property CodFormaPago: WideString;	Permite acceder y modificar el valor del código de la forma de pago de la agrupación.

property NumAgrupacion: Currency; (Nuevo 14.04.00)	
property NumAgrupacion: Currency;	Permite acceder y modificar el valor del numero de la agrupación.

property Observaciones: WideString; (Nuevo 14.04.00)	
property Observaciones: WideString;	Permite acceder y modificar al valor de las observaciones de la agrupación.

29.- OBJETO REMESA

Interface Remesa
property Estado: EstadoMaestro **readonly**; **procedure** Iniciar; **procedure** Acabar; **procedure** Nueva(bEsCobro: WordBool; sFecha, sCodBan: WideString; sCodMon,sTipoCont:WideString; bDescue: WordBool); **procedure** NuevaV2(bEsCobro: WordBool; sFecha, sCodBan: WideString; sCodMon, sTipoCont, sTipoMandato:WideString; bDescue: WordBool); **procedure** NuevaV3(bEsCobro: WordBool; sFecha, sCodBan: WideString; sCodMon, sTipoCont); **procedure** Modifica(nNumReme: Currency); **procedure** AnadirEfecto(nNumCart: Currency; nNumVen: Integer); **procedure** QuitarEfecto(nNumCart: Currency; nNumVen: Integer); **function** Anade: Currency; **procedure** Cancela;

property Estado:EstadoMaestro		
Esta propiedad nos indica el estado del objeto. Posibles estados: ESTM_NOACTIVO: Remesa cerrada. No se puede usar. ESTM_ACTIVO: Remesa activa. Podemos añadir, borrar, modificar. ESTM_NUEVO: Remesa en estado de inserción/eliminación de un nuevos efectos en una remesa nueva. ESTM_EDICION: Remesa en estado modificación, podemos añadir y quitar efectos en una remesa que ya existía.		
Parámetros:	(ninguno)	
Valor retornado	EstadoMaestro	Retorna el estado de la cartera (conjunto de vencimientos).

procedure Iniciar;
Este procedimiento es el que reserva los recursos necesarios para que la remesa pueda ser usada. Al iniciar el estado cambia de ESTM_NOACTIVO a ESTM_ACTIVO.

procedure Acabar;
Este procedimiento es el que libera los recursos utilizados por el objeto. Al acabar el estado cambia de ESTM_ACTIVO a ESTM_NOACTIVO.

procedure Nueva(bEsCobro: WordBool; sFecha, sCodBan, sCodMon: WideString; sTipoCont : WideString; bDescue: WordBool);		
Prepara la remesa para añadir efectos en ella. El estado pasa de ESTM_ACTIVO a ESTM_NUEVO.		
Parámetros:	bEsCobro	Indica si la remesa es de cobros (True) o pagos (False).
	sFecha	Fecha en que se crea la remesa.
	sCodBan	Código del banco.
	sCodMon	Código de la moneda.
	sTipoCont	Tipo contable.
	bDscue	Indica si la remesa es al descuento.

procedure NuevaV2 (bEsCobro: WordBool; sFecha, sCodBan, sCodMon: WideString; sTipoCont, sTipoMandato: WideString; bDescue: WordBool);		
Prepara la remesa para añadir efectos en ella. Incorpora el parámetro 'sTipoMandato' para informar el mandato de adeudo. El estado pasa de ESTM_ACTIVO a ESTM_NUEVO.		
Parámetros:	bEsCobro	Indica si la remesa es de cobros (True) o pagos (False).
	sFecha	Fecha en que se crea la remesa.

	sCodBan	Código del banco.
	sCodMon	Código de la moneda.
	sTipoCont	Tipo contable.
	sTipoMandato	Tipo mandado. Admite 2 valores “B2B” o “B2C”
	bDscue	Indica si la remesa es al descuento.

procedure NuevaV3 (bEsCobro: WordBool; sFecha, sCodBan, sCodMon: WideString; sTipoCont);		
Prepara la remesa para añadir efectos en ella. Elimina los parámetros ‘sTipoMandato’ y ‘bDescue’. En esta versión el tipo de mandato y el descuento se calculan dependiendo de los efectos que vamos añadiendo y eliminando a la remesa. El estado pasa de ESTM_ACTIVO a ESTM_NUEVO.		
Parámetros:	bEsCobro	Indica si la remesa es de cobros (True) o pagos (False).
	sFecha	Fecha en que se crea la remesa.
	sCodBan	Código del banco.
	sCodMon	Código de la moneda.
	sTipoCont	Tipo contable.

Procedure Modifica(nNumReme: Currency);		
Prepara una remesa para poder añadir o quitar efectos en ella.		
Parámetros:	nNumReme	Identificador la remesa. Se corresponde con el campo NUMREME de la tabla CARTERA.

procedure AnadirEfecto(nNumCart: Currency; nNumVen: Integer);		
Añade el efecto indicado en la remesa.		
Parámetros:	nNumCart	Identificador del conjunto de cartera de una factura. Se corresponde con el campo NUMCARTERA de la tabla CARTERA.
	nNumVen	Identificador del conjunto de cartera de una factura. Se corresponde con el campo NUMVEN de la tabla CARTERA.

procedure QuitarEfecto(nNumCart: Currency; nNumVen: Integer);		
Quita el efecto indicado de la remesa.		
Parámetros:	nNumCart	Identificador del conjunto de cartera de una factura. Se corresponde con el campo NUMCARTERA de la tabla CARTERA.
	nNumVen	Identificador del conjunto de cartera de una factura. Se corresponde con el campo NUMVEN de la tabla CARTERA.

function Anade: Currency;		
Mediante este procedimiento se termina la edición de una remesa y se guarda su contenido en la base de datos. Cambia el estado de ESTM_NUEVO o ESTM_EDICION a ESTM_ACTIVO.		
Valor retornado:	Currency	Identificador de la remesa. Se corresponde con el campo NUMREME de la tabla CARTERA.

procedure Cancela;
Cancela los efectos que se hayan añadido o quitado en la remesa desde que se llamó a la función **Nueva** o **Modificar**. Pasa de un estado ESTM_NUEVO o ESTM_EDICION a un estado ESTM_ACTIVO.

30.- OBJETO PRESUPUESTO

Objeto que permite la **creación, modificación de presupuestos.**

Interface Presupuesto
property Mes[nMes: Integer]: Currency; **procedure** Iniciar; **procedure** Acabar; **procedure** Nuevo(sCuenta: WideString; nEjercicio: Integer; sTipoContable: WideString; sCentro1, sCentro2, sCentro3, sMoneda: WideString; nImporte: Currency); **procedure** Modifica(sCuenta: WideString; nEjercicio: Integer; sTipoContable: WideString; sCentro1, sCentro2, sCentro3, sMoneda: WideString); **procedure** Anade;

Propiedades	Tipo	Descripción
Mes[nMes]	Currency	Se usa para introducir el presupuesto del mes especificado en la variable nMes.

Métodos	Tipo	Descripción
Iniciar	Procedimiento	Inicia el objeto presupuesto
Acabar	Procedimiento	Cierra el objeto presupuesto
Nuevo	Procedimiento	Inicia un presupuesto para una cuenta contable.
Modifica	Procedimiento	Prepara el objeto para poder modificar el presupuesto mensual de la cuenta indicada.
Anade	Procedimiento	Añade el presupuesto a la base de datos.

property Mes[nMes: Integer]: Currency;		
Esta propiedad nos permite especificar el presupuesto del mes indicado en el parámetro nMes.		
Parámetros:	nMes	Mes del ejercicio.
Valor retornado	Currency	Presupuesto del mes indicado.

procedure Iniciar;
Este procedimiento es el que reserva los recursos necesarios para poder trabajar con el presupuesto.

procedure Acabar;
Este procedimiento es el que libera los recursos utilizados por el objeto.

procedure Nuevo(sCuenta: WideString; nEjercicio: Integer; sTipoContable: WideString; sCentro1, sCentro2, sCentro3, sMoneda: WideString; nImporte: Currency);		
Prepara el objeto para añadir el presupuesto mensual.		
Parámetros:	sCuenta	Cuenta de la que desea introducirse el presupuesto.
	nEjercicio	Ejercicio al que pertenece el presupuesto.
	sTipoContable	Tipo contable
	sCentro1	Centro de coste 1.
	scentro2	Centro de coste 2.
	scentro3	Centro de coste 3.
	sMoneda	Moneda
	nImporte	Importe total del ejercicio.

procedure Modifica(sCuenta: WideString; nEjercicio: Integer; sTipoContable: WideString; sCentro1, sCentro2, sCentro3, sMoneda: WideString);		
Prepara la cuenta para modificar su presupuesto.		
Parámetros:	sCuenta	Cuenta de la que desea introducirse el presupuesto.
	nEjercicio	Ejercicio al que pertenece el presupuesto.
	sTipoContable	Tipo contable
	sCentro1	Centro de coste 1.
	sCentro2	Centro de coste 2.
	sCentro3	Centro de coste 3.
	sMoneda	Moneda

procedure Anade;
Mediante este procedimiento se termina la edición de un presupuesto y se guarda su contenido en la base de datos.

31.- OBJETO OPCION

El objeto **Opcion** permite llamar a las opciones visuales de nexus. Cada una de estas opciones tiene un identificador único y un conjunto de parámetros.

Interface Opcion
property IdOpcion: WideString **writeonly**; **procedure** Iniciar; **procedure** Acabar; **procedure** AnadirParametro(const sNombre: WideString; const sValor: WideString); **procedure** Ejecutar;

property IdOpcion: WideString writeonly;
Texto que identifica a cada opción de forma única.

procedure Iniciar;
Reserva los recursos para que el objeto pueda funcionar correctamente.

procedure Acabar;
Libera los recursos del objeto.

procedure AnadirParametro(sNombre: WideString; sValor: WideString);		
Añade parámetros con su correspondiente valor. En función del valor de estos parámetros la opción tendrá diferentes comportamientos.		
Parámetros:	sNombre	Nombre del parámetro
	sValor	Valor del parámetro. Siempre es de tipo string.

Procedure Ejecutar;
Llama a la opción visual de nexus que tiene el identificador de la propiedad **IdOpcion** con los parámetros añadidos con **AnadirParametro**.

Interface OpcionModal
function EjecutarModal: EstadoModal; **property** Resultado: OleVariant; **readonly**; **A tener en cuenta:** esta interface solo es operativa en: acciones de CRM, expedientes, cuotas y documentos de compra y venta.

function EjecutarModal: EstadoModal;
Llama a la opción visual de nexus que tiene el identificador de la propiedad **IdOpcion** con los parámetros añadidos con **AnadirParametro** y espera a que la ejecución termine. Devuelve el estado de la ejecución. Por ejemplo, si se acepta la pantalla editada se retorna mrOK y si se cancela mrCancel.

property Resultado: OleVariant readonly;
Cuando se da de alta una entidad y se acepta la pantalla se almacena el código de dicha entidad en esta propiedad. Si se cancela el alta el valor de propiedad será nulo.

EstadoModal (Tipo enumerado)		
	Valor	Significado
mrNone	0	Valor por defecto
mrOK	1	Se acepta la pantalla de alta o edición
mrCancel	2	Se cancela la pantalla de alta o edición
mrAbort	3	Se ha seleccionado la opción abortar
mrRetry	4	Si la pantalla permite reintentarse se ha seleccionado esa opción
mrIgnore	5	Si la pantalla permite ignorarse se ha seleccionado esa opción
mrYes	6	Se ha seleccionado la opción Sí.
mrNo	7	Se ha seleccionado la opción No.
mrAll	8	Se ha seleccionado la opción Todos.
mrNoToAll	9	Se ha seleccionado la opción no a todo.
mrYesToAll	10	Se ha seleccionado la opción sí a todo.
mrClose	11	Se ha cerrado la ventana

32.- OBJETO LISTADO

Objeto que permite realizar **informes** pasando los parámetros deseados directamente.

Interface Listado
property IdListado: WideString **writeonly**; **property** Fichero: WideString **writeonly**; **property** Parametros: OleVariant **readonly**; **property** Destino: DestinoListado **writeonly**; **property** Copias: Integer **writeonly**; **property** Impresora: WideString **writeonly**; **property** Bandeja: WideString **writeonly**; **property** Modelo: WideString **writeonly**;

```
property MostrarProgreso: WordBool;
property IncluirFirmaDigital: WordBool;
procedure Iniciar;
procedure Acabar;
procedure Definir;
procedure AnadirParametro(sNombre: WideString; sValor: WideString);
procedure Imprimir;
```

DestinoListado (Tipo enumerado)		
	Valor	**Significado**
destImpresora	0	Impresora
destHTML	1	Fichero en formato HTML
destPDF	2	Fichero en formato PDF
destRTF	3	Fichero en formato RTF
destJPG	4	Fichero en formato JPG
destGIF	5	Fichero en formato GIF
destBMP	6	Fichero en formato BMP
destEMF	7	Fichero en formato EMF
destWMF	8	Fichero en formato WMF
destPantalla	9	Presentación preliminar
destExcel	10	Fichero en formato XLS
destFacturaE	11	Fichero en formato EFACTURA
destDocOnTime	12	Fichero en formato DOCONTIME

property IdListado: WideString writeonly;
Texto que identifica a cada listado de forma única.

property Parametros: OleVariant readonly;		
Valor retornado:	Variant representando un array.	[0] → Nº de parámetros (n). [1..n][0] → Nombre del parámetro. [1..n][1] → Descripción del parámetro.

property Destino: DestinoListado writeonly;
Destino del listado a elegir dentro de los posibles destinos especificados en el tipo enumerado DestinoListado.

property Fichero: WideString writeonly;
Si se ha elegido un Destino que se refiere a un fichero debe indicarse aquí la ruta y el nombre del fichero.

property Copias: integer writeonly;
Nº de copias a imprimir. Sólo tiene validez si el destino es **destImpresora**.

property Impresora: WideString writeonly;
Nombre de la impresora en la que se imprimirá el listado.

property bandeja: WideString writeonly;
Nombre de la bandeja de la impresora por la que saldrá el listado.

property modelo: WideString writeonly;
Nombre del modelo a imprimir o nombre del fichero en el que se encuentra dicho modelo. Ejemplo: 1. Listado de clientes por provincia 2. lstcli.003

property MostrarProgreso: WordBool;
Para indicar o comprobar si el listado tiene la barra de progeso activa.

property IncluirfirmalDigital: WordBool;
Si el destino del listado es destPDF y en los datosgenerales tenemos especificada el fichero con la firma digital, automáticamente se incluirá la firma digital en el fichero PDF. Con esta propiedad podemos indicar si, en el caso anterior, no queremos incluir la firma digital al fichero PDF.

procedure Iniciar;
Reserva los recursos para que el objeto pueda funcionar correctamente.

procedure Acabar;
Libera los recursos del objeto.

Procedure Definir;
Permite definir el listado especificado en **IdListado**.

procedure AnadirParametro(sNombre: WideString; sValor: WideString);		
Añade parámetros con su correspondiente valor. En función del valor de estos parámetros el listado tendrá diferentes comportamientos.		
Parámetros:	sNombre	Nombre del parámetro
	sValor	Valor del parámetro. Siempre es de tipo string.

Procedure Imprimir;
Llama a la impresión visual de nexus que tiene el identificador de la propiedad **IdListado** con los parámetros añadidos con **AnadirParametro**.

33.- OBJETO ESTRUCTURA

Objeto que permite la **creación, modificación y borrado de estructuras** (escandallos).

Interface Estructura

property Estado: EstadoMaestro readonly;
property AsString[const sCampo: WideString]: WideString;
property AsFloat[const sCampo: WideString]: Double;
property AsInteger[const sCampo: WideString]: Integer;
property AsBoolean[const sCampo: WideString]: WordBool;
procedure Iniciar;
procedure EditarEstructura(sCodArt:WideString);
procedure Acabar;
procedure Cancelar;
procedure CambiarUnidades(nUnidades:Double);
procedure Guardar;
procedure ActualizarCostes;
function CosteFabricacion: double;
procedure AnadeFase(const sFase: WideString; const sNomFase: WideString);
procedure AnadeComponente(const sFase: WideString; const sCodArt: WideString;nUnidades:double; lEsFinal: WordBool);
procedure AnadePerfilOperario(const sFase: WideString; const PerfilO: WideString; sTiempo:WideString);
procedure AnadePerfilMaquina(const sFase: WideString; const PerfilM: WideString; sTiempo:WideString);
procedure EditarElemento(nNumLin: Currency);
procedure BorrarFase(const sFase: WideString);
procedure BorrarElemento(nNumLin: Currency);
procedure AnadeElemento;
function ListaComponentes (const sCodArt: WideString; nUnidades: Currency) OleVariant;

Propiedades	Tipo	Descripción
AsString	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsInteger	Tabla(Integer)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsFloat	Tabla(Float)	Asigna/lee valores a/de los campos del registro nuevo/actual.

AsBoolean	Tabla(Lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
Estado	EstadoMaestro	Devuelve el estado en que se encuentra el objeto Estructura

*EstadoMaestro: Ver objeto Maestro.

Método	Tipo	Descripción
Iniciar	Procedimiento	Inicia el objeto
Acabar	Procedimiento	Cierra el objeto
Cancelar	Procedimiento	Cancela la edición de la estructura
Guardar	Procedimiento	Guarda la estructura
CambiarUnidades	Procedimiento	Modifica las unidades a generar del producto acabado
ActualizarCoste	Procedimiento	Recalcula el precio de coste de los componentes
CosteFabricacion	Función	Devuelve el coste del producto acabado.
AnadeFase	Procedimiento	Añade una nueva fase a la estructura
AnadeComponente	Procedimiento	Añade un nuevo componente a la estructura en la fase especificada.
AnadePerfilOperario	Procedimiento	Añade un nuevo perfil de operario a la estructura en la fase especificada.
AnadePerfilMaquina	Procedimiento	Añade un nuevo perfil de máquina a la estructura en la fase especificada.
EditarElemento	Procedimiento	Permite editar un componente o perfil ya existente de la estructura.
BorrarElemento	Procedimiento	Permite borrar un componente o perfil de la estructura.
BorrarFase	Procedimiento	Permite borrar una fase y su contenido de la estructura
EditarEstructura	Procedimiento	Carga, si existe, la estructura del artículo solicitado.
ListaComponentes	Function	Devuelve las hojas de la estructura
AnadeElemento	Procedimiento	Guarda los cambios pendientes en un elemento .

Las primeras funciones de la lista que se presentan a continuación se utilizan para **asignar / leer valores de la cabecera o de las líneas del documento**.

Deben tenerse en cuenta algunas normas importantes:

- Si el nombre del campo no existe se producirá un error.
- Si el valor asignado al campo especificado no es correcto, no se producirá un error. Este se producirá al intentar guardar el documento o la línea.

Hay que tener en cuenta lo siguiente:

- Los campos de tiempo se han de asignar como cadenas de texto como 'hh:mm:ss'

property AsString[sCampo:String]: String		
A través de esta propiedad podemos asignar valor a los campos especificando que su tipo es string.		
Parámetros:	SCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsInteger[sCampo:String]: Integer		
A través de esta propiedad podemos asignar valor a los campos especificando que su tipo es entero.		
Parámetros:	SCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloat [sCampo:String]: Double		
A través de esta propiedad podemos asignar valor a los campos especificando que su tipo es double.		
Parámetros:	SCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBoolean[sCampo:String]: Boolean		
A través de esta propiedad podemos asignar valor a los campos especificando que su tipo es lógico.		
Parámetros:	SCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

A continuación, se presentan las propiedades y métodos necesarios para **crear, borrar y modificar estructuras.**

property Estado:EstadoMaestro		
Esta propiedad nos indica el estado del objeto. Posibles estados: ESTM_NOACTIVO: Estructura cerrada. No se puede usar. ESTM_ACTIVO: Estructura activa. Podemos añadir, borrar, modificar. ESTM_NUEVO: Estructura en estado de inserción de un nuevo registro. ESTM_EDICION: Estructura en estado modificación, podemos asignar valores a los campos.		
Valor retornado	EstadoMaestro	Retorna el estado del documento.

procedure Iniciar;
Este procedimiento es el que reserva los recursos necesarios para que la estructura pueda ser usada. Al iniciar el estado cambia de ESTM_NOACTIVO a ESTM_ACTIVO.

procedure EditarEstructura(sCodArt:String);
Este procedimiento es que carga la estructura del código de artículo que queremos editar o crear su estructura.

procedure Acabar;
Este procedimiento es el que libera los recursos utilizados por el objeto. Al acabar el estado cambia de ESTM_ACTIVO a ESTM_NOACTIVO.

procedure Cancelar;
Cancela los cambios realizados. Pasa a estado ESTM_ACTIVO.

procedure CambiarUnidades(nUnidades: Double);
Cambia las unidades que se obtendrán en la fabricación del producto acabado.

Procedure Guardar;
Almacena los valores de los campos en la base de datos con los cambios introducidos por el programador. Pasa de los estados ESTM_NUEVO o EST_EDICION a ESTM_ACTIVO.

Procedure ActualizarCostes;
Recalcula los costes de todos los componentes del artículo a fabricar.

Funcion CosteFabricacion:double;
Devuelve el coste unitario de fabricación del producto acabado.

Procedure AnadeFase(sFase, sNomFase:String;		
Crea una fase en la estructura actual.		
Parámetros:	sFase	Código de la fase
	SNomFase	Nombre descriptivo de la fase

Procedure AnadeComponente(sFase, sCodArt: string; nUnidades: double; lEsFinal: WordBool);		
Crea un nuevo componente en la fase especificada con las unidades indicadas. A partir de ese momento podremos editar los valores de la línea.		
Parámetros:	sFase	Código de la fase.

	sCodArt	Código de artículo.
	NUnidades	Unidades a consumir del componente.
	lEsFinal	El componente es final. En las órdenes de fabricación no se fabrica, se coge de stock.

Procedure AnadePerfilOperario(sFase, sPerfilO, sTiempo: string);		
Crea un nuevo perfil de operario en la fase especificada con las unidades indicadas. A partir de ese momento podremos editar los valores de la línea.		
Parámetros:	Sfase	Código de la fase.
	SPerfilO	Código de perfil de operario
	STiempo	Tiempo a consumir del perfil (en formato hh:mm:ss)

Procedure AnadePerfilMaquina(sFase, sPerfilM, sTiempo: string);		
Crea un nuevo perfil de máquina en la fase especificada con las unidades indicadas. A partir de ese momento podremos editar los valores de la línea.		
Parámetros:	Sfase	Código de la fase.
	SperfilM	Código de perfil de máquina
	Stiempo	Tiempo a consumir del perfil (en formato hh:mm:ss)

procedure EditarElemento(nNumLin: Currency);		
Busca y pone el componente o perfil indicado en el parámetro nNumLin en edición, preparado para editar el contenido de sus campos.		
Parámetros:	NNumLin	Identificador de la línea. Se corresponde con el campo NUMLIN del fichero ESCANDALLO.

procedure BorrarFase(sFase: string);		
Borra la fase indicada, así como todos sus componentes y perfiles.		
Parámetros:	SFase	Código de la fase a borrar.

procedure BorrarElemento(nNumLin: Currency);		
Borra el componente o perfil correspondiente a dicho identificador.		
Parámetros:	nNumLin	Identificador de la línea (campo NumLin de la tabla ESCANDALLO)..

function ListaComponentes(const sCodArt: WideString: nUnidades: Currency) OleVariant;		
Devuelve lista de la estructura de un artículo, a partir de las unidades indicadas		
Parámetros:	sCodArt	Identificador del código del artículo
	nUnidades	Nº de unidades

Procedure AnadeComponente;
Guarda los cambios pendientes en un componente que hemos añadido o modificado.

34.- OBJETO ORDENPRODUCCION

Objeto que permite la creación, modificación, evolución y borrado de órdenes de produccion.

Interface OrdenProduccion
Procedure Iniciar; **procedure** Acabar; **procedure** Nuevo(sFecha:WideString); **procedure** Borra(nIdProd:Currency); **procedure** Modifica(nIdProd:Currency); **function** Anade:currency; **procedure** Cancela; **function** NuevoProducto(const sCodArt: WideString;nUnidades:double):Currency; **function** NuevaFase(nIdLinP:Currency; const sFase: WideString; const sNomFase: WideString):Currency; **function** NuevoComponente(nIdFase:Currency; const sCodArt: WideString;nUnidades:double; lEsFinal:wordbool):Currency; **function** NuevoOperario(nIdFase:Currency; const PerfilO: WideString; sTiempo:WideString):Currency; **function** NuevoOperarioV2(nIdFase:Currency; const PerfilO: WideString; sTiempo:WideString; sCodEmp: WideString):Currency; **function** NuevaMaquina(nIdFase:Currency; const PerfilM: WideString; sTiempo:WideString):Currency; **procedure** EditarLinea(nIdLinP: Currency); **procedure** BorrarFase(nIdFase:Currency); **procedure** BorrarLinea(nIdLinP: Currency); **procedure** FabricarOrden(const sFecha: WideString; const sTipoStock: WideString); **procedure** FabricarFase(nIdFase: Currency; const sFecha: WideString; const sTipoStock: WideString); **procedure** FabricarFaseProporcional(nIdFase: Currency; nUnidades: Currency; const sFecha: WideString; const sTipoStock: WideString); **procedure** FabricarProductoPrevisto(nIdLinP: Currency; const sFecha: WideString; const sTipoStock: WideString); safecall; **procedure** FabricarProductoProporcional(nIdLinP: Currency; nUnidades: Currency; const sFecha: WideString; const sTipoStock: WideString); safecall; **procedure** FabricarProductoSinConsumir(nIdLinP: Currency; nUnidades: Currency; const sFecha: WideString; const sTipoStock: WideString); safecall; **procedure** ConsumirComponente(nIdLinP: Currency; nUnidades: Currency; const sFecha: WideString; const sTipoStock: WideString); safecall; **procedure** ConsumirOperario(nIdLinP: Currency; const sCodEmp: WideString; const sTiempo: WideString; const sFecha: WideString; const sTipoStock: WideString); safecall;

procedure ConsumirMaquina(nIdLinP: Currency; const sCodMaq: WideString; const sTiempo: WideString; const sFecha: WideString; const sTipoStock: WideString); safecall;
procedure FabricarProductoEnCurso(nIdLinC: Currency; nUnidades: Currency; const sCodAlm: WideString); safecall;
procedure ConsumirComponenteEnCurso(nIdLinC: Currency; nUnidades: Currency; const sCodAlm: WideString); safecall;
procedure ConsumirMaquinaEnCurso(nIdLinC: Currency; const sTiempo: WideString); safecall;
procedure ConsumirOperarioEnCurso(nIdLinC: Currency; const sTiempo: WideString); safecall;
procedure FabricarConsumirOperacionEnCurso(nIdLogP: Currency); safecall;
procedure IndicarDetalle(nIdLogP, nIdLinP: Currency; nUnidades: Double; sNumSerie, sLote, sUbicacion, sFecCaduc: WideString); safecall;
procedure AnadirDetalle(nIdLinC, nNumGrupo: Currency; nUnidades: Double; sNumSerie, sLote, sUbicacion, sFecCaduc: WideString); safecall;
procedure CambiarDetalle(nIdLinC, nNumLin: Currency; nUnidades: Double; sNumSerie, sLote, sUbicacion, sFecCaduc: WideString); safecall;
procedure BorrarDetalle(nIdLinC, nNumLin: Currency); safecall;
procedure TerminarOrden(const sFecha: WideString); safecall;
procedure TerminarFase(nIdFase: Currency; const sFecha: WideString); safecall;
procedure TerminarLinea(nIdLinP: Currency; const sFecha: WideString); safecall;
procedure DeshacerFabricacionProducto(nIdLinC: Currency; nNumLin: Currency); safecall;
procedure DeshacerConsumoComponente(nIdLinC: Currency; nNumLin: Currency); safecall;
procedure DeshacerConsumoMaquina(nIdLinC: Currency; nNumLin: Currency); safecall;
procedure DeshacerConsumoOperario(nIdLinC: Currency; nNumLin: Currency); safecall;
procedure DeshacerOperacion(nIdLogP: Currency); safecall;
procedure DeshacerTerminarOrden; safecall;
procedure DeshacerTerminarFase(nIdFase: Currency); safecall;
procedure DeshacerTerminarLinea(nIdLinP: Currency); safecall;
function NuevoTrabajo(const sFecha: WideString; const sMotivo: WideString): Currency; safecall;
procedure AbrirTrabajo(nIdTrab: Currency); safecall;
procedure CerrarTrabajo(nIdTrab: Currency; const sFecha: WideString); safecall;
procedure BorrarTrabajo(nIdTrab: Currency); safecall;
Property TrabajoActivo: Currency readwrite;
Property Estado: EstadoMaestro readonly;
property AsStringCab[const sCampo: WideString]: WideString;
property AsFloatCab[const sCampo: WideString]: Currency;
property AsIntegerCab[const sCampo: WideString]: Integer;
property AsBooleanCab[const sCampo: WideString]: WordBool;
property AsStringLin[const sCampo: WideString]: WideString;
property AsFloatLin[const sCampo: WideString]: Double;
property AsIntegerLin[const sCampo: WideString]: Integer;
property AsBooleanLin[const sCampo: WideString]: WordBool;
property AsCurrencyLin[const sCampo: WideString]: Currency;

```
property AvisarStock: WordBool;
procedure CambiarUnidadesProducto(  nIdLinP, nUnidades:currency);
procedure ActualizarCostesComponentes;
```

Propiedades	Tipo	Descripción
AsStringCab	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsIntegerCab	Tabla(Integer)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsFloatCab	Tabla(Float)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsBooleanCab	Tabla(Lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsStringLin	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsIntegerLin	Tabla(Integer)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsFloatLin	Tabla(Float)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsBooleanLin	Tabla(Lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsCurrencyLin	Tabla(Currency)	Asigna/lee valores a/de los campos del registro nuevo/actual.
TrabajoActivo	Currency	Asigna/lee la orden de trabajo activa
Estado	EstadoMaestro	Devuelve el estado en que se encuentra el objeto Estructura
AvisarStock	Lógico	Asignar valor para activar o no los mensajes de aviso de stock

*EstadoMaestro: Ver objeto Maestro.

Método	Tipo	Descripción
Iniciar	Procedimiento	Inicia el objeto.
Acabar	Procedimiento	Cierra el objeto.
Cancela	Procedimiento	Cancela los cambios realizados en la orden.
Nuevo	Procedimiento	Crea una nueva orden.
Borra	Procedimiento	Borra la orden indicada.
Modifica	Procedimiento	Carga la orden indicada para su edición.
Anade	Función	Guarda y devuelve el identificador de

		la orden.
NuevaFase	Funcion	Añade una nueva fase al artículo especificado.
NuevoComponente	Funcion	Añade un nuevo componente a la fase especificada.
NuevoOperario	Funcion	Añade un nuevo perfil de operario a la fase especificada. EL operario es el defecto de su perfil
NuevoOperarioV2	Función	Añade un nuevo perfil de operario a la fase especificada, y se puede informar cualquier operario que cumpla el perfil.
NuevaMaquina	Funcion	Añade un nuevo perfil de máquina a la fase especificada.
EditarLinea	Procedimiento	Busca y pone en edición el registro especificado.
BorrarLinea	Procedimiento	Borra el registro especificado.
BorrarFase	Procedimiento	Borra la fase especificada y todo su contenido.
FabricarOrden	Procedimiento	Fabrica y consume todos los elementos de la orden.
FabricarFase	Procedimiento	Fabrica y consume todos los elementos de la fase especificada
FabricarFaseProporcional	Procedimiento	Fabrica y consume proporcionalmente a las unidades del artículo al que pertenece todos los elementos de la fase especificada
FabricarProductoPrevisto	Procedimiento	Fabrica y consume todos los componentes/tiempos del producto acabado especificado.
FabricarProductoProporcional	Procedimiento	Fabrica y consume todos los componentes/tiempos del producto acabado especificado proporcionalmente a las unidades a fabricar.
FabricarProductoSinConsumir	Procedimiento	Solo fabrica el producto acabado especificado, sin rebajar los componentes.
ConsumirComponente	Procedimiento	Consume unidades del componente especificado.
ConsumirOperario	Procedimiento	Consume tiempo del operario especificado.
ConsumirMaquina	Procedimiento	Consume tiempo de la máquina especificada.

FabricarProductoEnCurso	Procedimiento	Hace la fabricación de un producto acabado en una orden de trabajo.
ConsumirComponteEnCurso	Procedimiento	Hace el consumo de un componente en una orden de trabajo.
ConsumirMaquinaEnCurso	Procedimiento	Hace el consumo de una máquina en una orden de trabajo.
ConsumirOperarioEnCurso	Procedimiento	Hace el consumo de un operario en una orden de trabajo.
FabricarConsumirOperarionEnCurso	Procedimiento	Hace la fabricación y consumos de una operación de una orden de trabajo.
IndicarDetalle	Procedimiento	Permite indicar los números de serie / lotes / … al fabricar / consumir, cuando no se conoce el identificadore de la línea de consumo / fabricación porque todavía no se ha grabado en la base de datios
AnadirDetalle	Procedimiento	Permiter añadir líneas de detalle donde se indica el número de serie / lote / … cuando se conoce el identificador de la línea de consumo / fabricación
CambiarDetalle	Procedimiento	Permite cambiar líneas de detalle cuando se conoce el identificador de la línea de consumo / fabricación
BorrarDetalle	Procedimiento	Permite borrar líneas de detalle cuando se conoce el identificador de la línea de consumo / fabricación
TerminarOrden	Procedimiento	Da por finalizada la orden de producción.
TerminarFase	Procedimiento	Da por finalizada la fase especificada.
TerminarLinea	Procedimiento	Da por finalizada la línea especificada.
DeshacerFabricacionProducto	Procedimiento	Borra la fabricación especificada.
DeshacerConsumoComponente	Procedimiento	Borra el consumo de componente especificado.
DeshacerConsumoMaquina	Procedimiento	Borra el consumo de máquina especificada.
DeshacerConsumoOperario	Procedimiento	Borra el consumo de operario especificado.
DeshacerOperacion	Procedimiento	Borra todas las acciones de la operación especificada.
DeshacerTerminarOrden	Procedimiento	Deshace la finalización de la orden.
DeshacerTerminarFase	Procedimiento	Deshace la finalización de la fase.
DeshacerTerminarLinea	Procedimiento	Deshace la finalización de la línea.

NuevoTrabajo	Funcion	Crea una nueva orden de trabajo
AbrirTrabajo	Procedimiento	Abre una orden de trabajo.
CerrarTrabajo	Procedimiento	Cierra una orden de trabajo.
BorrarTrabajo	Procedimiento	Borra una orden de trabajo.
CambiarUnidadesProducto	Procedimiento	Modifica las unidades a fabricar de un producto acabado.
ActualizarCostesComponentes	Procedimiento	Carga los costes de los componentes de la orden con los precios actuales.

Las **tablas que intervienen en los procesos**, así como las relaciones entre éllas son las siguientes:

Tabla:	CABEPROD	Cabecera de órdenes de producción.
Campos clave:	IDPROD	Identificador de orden de producción

Tabla:	LINEPROD	Líneas de órdenes de producción.
Campos clave:	IDPROD	Identificador de orden de producción
	IDLINP	Identificador de línea
	IDFASE	Identificador de la fase a la que pertenece la línea.
	IDPADRE	Identificador de la línea del producto acabado al que pertenece dicha línea; si es 0, indica que es un producto acabado de primer nivel.

Tabla:	PRODFASES	Fases de órdenes de producción.
Campos clave:	IDPROD	Identificador de orden de producción
	IDFASE	Identificador de fase
	IDLINP	Identificador de la línea a la que pertenece la fase.

Tabla:	LOGPROD	Operaciones generadas, entendiendo como operación el conjunto de acciones necesarias (que se guardarán en la tabla histprod) para completar una evolución de la orden de producción (por ejemplo, si lanzamos una fase, las acciones serán todos los consumos generados por dicha fase).
Campos clave:	IDPROD	Identificador de orden de producción
	IDLOGP	Identificador de la operación
	IDTRAB	Identificador de la orden de trabajo a la que pertenece.

Tabla:	HISTPROD	Acciones generadas al evolucionar una orden (consumos de componentes, consumos de tiempo, fabricaciones de productos acabados, marcas de terminación de líneas...).
Campos clave:	IDPROD	Identificador de orden de producción
	IDLOGP	Identificador de la operación a la que pertenece
	IDTRAB	Identificador de la orden de trabajo a la que pertenece.
	IDLINP	Identificador de la línea que ha generado la acción.
	IDLINC	Identificador de la acción.
	NUMLIN	Identificador de línea de evolución; cuando enviamos una acción a trabajo en curso (este campo valdrá 0), se pueden ir haciendo fabricaciones o consumos parciales sobre la acción (y cada evolución tendrá un valor distinto)

Tabla:	TRABAJOS	Ordenes de trabajo.
Campos clave:	IDPROD	Identificador de orden de producción
	IDTRAB	Identificador de trabajo.

Property AsStringCab[sCampo:String]: String		
A través de esta propiedad podemos asignar valor a los campos especificando que su tipo es string.		
Parámetros:	SCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerCab[sCampo:String]: Integer		
A través de esta propiedad podemos asignar valor a los campos especificando que su tipo es entero.		
Parámetros:	SCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatCab [sCampo:String]: Double		
A través de esta propiedad podemos asignar valor a los campos especificando que su tipo es double.		
Parámetros:	SCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanCab[sCampo:String]: Boolean
A través de esta propiedad podemos asignar valor a los campos especificando que su tipo es lógico.

Parámetros:	SCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

Property AsStringLin[sCampo:String]: String		
Con esta propiedad podemos asignar valor a los campos especificando que su tipo es string.		
Parámetros:	SCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerLin[sCampo:String]: Integer		
A través de esta propiedad podemos asignar valor a los campos especificando que su tipo es entero.		
Parámetros:	SCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatLin [sCampo:String]: Double		
Con esta propiedad podemos asignar valor a los campos especificando que su tipo es double.		
Parámetros:	SCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanLin[sCampo:String]: Boolean		
Con esta propiedad podemos asignar valor a los campos especificando que su tipo es lógico.		
Parámetros:	SCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsCurrencyLin[sCampo:String]: Currency		
Con esta propiedad podemos asignar valor a los campos especificando que su tipo es currency.		
Parámetros:	SCampo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AvisarStock: WordBool		
Con esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es lógico.		
Valor retornado	Currency	Valor del campo

A continuación, se presentan las propiedades y métodos necesarios para **crear, borrar y modificar** órdenes.

property Estado:EstadoMaestro

Esta propiedad nos indica el estado del objeto. Posibles estados: ESTM_NOACTIVO: Estructura cerrada. No se puede usar. ESTM_ACTIVO: Estructura activa. Podemos añadir, borrar, modificar. ESTM_NUEVO: Estructura en estado de inserción de un nuevo registro. ESTM_EDICION: Estructura en estado modificación, podemos asignar valores a los campos.		
Valor retornado	EstadoMaestro	Retorna el estado del documento.

property TrabajoActivo[nIdTrab:Currency]: Currency		
A través de esta propiedad podemos cambiar o consultar la orden de trabajo activa.		
Parámetros:	NIdTrab	Identificador de trabajo a activar
Valor retornado	Currency	Identificador de trabajo activo.

procedure Iniciar;
Este procedimiento es el que reserva los recursos necesarios para que la orden pueda ser usada. Al iniciar el estado cambia de ESTM_NOACTIVO a ESTM_ACTIVO.

procedure Acabar;
Este procedimiento es el que libera los recursos utilizados por el objeto. Al acabar el estado cambia de ESTM_ACTIVO a ESTM_NOACTIVO.

procedure Cancela;
Cancela los cambios realizados. Pasa a estado ESTM_ACTIVO.

Function Anade:Currency;
Almacena los valores de los campos en la base de datos con los cambios introducidos por el programador y devuelve el identificador de orden (IDPROD). Pasa de los estados ESTM_NUEVO o EST_EDICION a ESTM_ACTIVO.

Procedure Nuevo(sFecha:String);		
Crea una nueva orden de producción. Pasa a estado ESTM_NUEVO.		
Parámetros:	SFecha	Fecha de la orden

Procedure Borra(nIdProd:Currency);
Borra la orden de producción indicada. Pasa a estado ESTM_ACTIVO.

Parámetros:	NIdProd	Identificador de la orden

Procedure Modifica(nIdProd:Currency);		
Pone en edición una orden de producción. Pasa a estado ESTM_EDICION.		
Parámetros:	NIdProd	Identificador de la orden.

Function NuevoProducto(sCodArt:String; nUnidades:Double):Currency;		
Añade un artículo a fabricar en la orden y devuelve el identificador de línea generado (IDLINP).		
Parámetros:	sCodArt	Código de artículo.
	nUnidades	Unidades a fabricar.

Function NuevaFase(nIdLinP:currency; sFase, sNomFase:String):Currency;		
Crea una fase en el artículo a fabricar indicado y devuelve el identificador de fase generado (IDFASE).		
Parámetros:	nIdLinP	Identificador de la línea de artículo a fabricar.
	sFase	Código de la fase
	SNomFase	Nombre descriptivo de la fase

Function NuevoComponente(nIdFase:currency;CodArt: string; nUnidades: double;lEsFinal:boolean):Currency;		
Crea un nuevo componente en la fase especificada con las unidades indicadas y devuelve el identificador de la línea generado (IDLINP)		
Parámetros:	NIdFase	Identificador de la fase.
	SCodArt	Código de artículo.
	NUnidades	Unidades a consumir del componente.
	LEsFinal	Si el componente añadido también puede fabricarse, si ponemos true solo añadirá está línea, pero si le decimos false, pondrá este componente así como su estructura (será un artículo ficticio).

Function NuevoOperario(nIdFase:currency; sPerfilO, sTiempo: string):Currency;		
Crea un nuevo perfil de operario en la fase especificada con el tiempo indicado y devuelve el identificador de la línea generada (IDLINP). Utiliza siempre como operario el que tenga pro defecto en el prefil de operario.		
Parámetros:	NIdFase	Identificador de la fase.
	SPerfilO	Código de perfil de operario
	STiempo	Tiempo a consumir del perfil (en formato hh:mm:ss)

Function NuevoOperarioV2(nIdFase:currency; sPerfilO, sTiempo, sCodemp: string) :Currency;		
Crea un nuevo perfil de operario en la fase especificada con el tiempo indicado y devuelve el identificador de la línea generada (IDLINP). Además, permite informar un operario cualqueira del perfil.		
Parámetros:	NIdFase	Identificador de la fase.
	SPerfilO	Código de perfil de operario
	STiempo	Tiempo a consumir del perfil (en formato hh:mm:ss)
	sCodEmp	Código del operario.

Function NuevaMaquina(nIdFase:currency;sPerfilM, sTiempo: string):Currency;		
Crea un nuevo perfil de máquina en la fase especificada con el tiempo y devuelve el identificador de línea generado (IDLINP).		
Parámetros:	NIdFase	Identificador de la fase.
	SperfilM	Código de perfil de máquina
	Stiempo	Tiempo a consumir del perfil (en formato hh:mm:ss)

procedure EditarLinea(nIdLinP: Currency);		
Busca y pone la línea indicada preparada para editar el contenido de sus campos.		
Parámetros:	NIdLinP	Identificador de la línea.

procedure BorrarFase(nIdFase: currency);		
Borra la fase indicada, así como todos sus componentes y perfiles.		
Parámetros:	NIdFase	Identificador de la fase a borrar.

procedure BorrarLinea(nIdLinP: Currency);		
Borra el componente o perfil correspondiente a dicho identificador.		
Parámetros:	NIdLinP	Identificador de la línea.

A tener en cuenta: ParámetrosTipoStock: en todas las operaciones que se hagan para fabricar o consumir la orden de producción existe este parámetro, que nos indica si la operación se envía a trabajo en curso, es decir que se podrá evolucionar desde trabajos (pondríamos “P”) o por el contrario si es definitiva, que para los artículos ya actualizaría el stock (pondríamos “S”). Si se deja en blanco, cogerá el valor que tenga la cabecera de la orden sobre la que estemos trabajando.

procedure FabricarOrden(sFecha, sTipoStock:string);		
Fabrica y consume todo lo pendiente de la orden de producción.		
Parámetros:	SFecha	Fecha de fabricación
	STipoStock	“S” o “P”

procedure FabricarFase(nIdFase:currency;sFecha, sTipoStock:string);		
Fabrica y consume todo lo pendiente de la fase indicada.		
Parámetros:	nIdFase	Identificador de la fase
	SFecha	Fecha de fabricación/consumo.
	STipoStock	“S” o “P”

Procedure FabricarFaseProporcional(nIdFase,nUnidades:currency; sFecha, sTipoStock:string);		
Fabrica y consume las unidades y tiempos proporcionalmente a las unidades indicadas respecto a las unidades a fabricar del artículo acabado de dicha fase.		
Parámetros:	nIdFase	Identificador de la fase.
	NUnidades	Unidades a tratar.
	SFecha	Fecha de fabricación/consumo.
	STipoStock	“S” o “P”

procedure FabricarProductoPrevisto(nIdLinP:currency;sFecha, sTipoStock:string);		
Fabrica y consume todo lo pendiente del producto acabado indicado.		
Parámetros:	NIdLinP	Identificador del producto acabado.
	SFecha	Fecha de fabricación/consumo.
	STipoStock	“S” o “P”

Procedure FabricarProductoProporcional(nIdLinP,nUnidades:currency; SFecha, sTipoStock:string);		
Fabrica y consume las unidades y tiempos proporcionalmente a las unidades indicadas respecto a las unidades a fabricar del artículo acabado.		
Parámetros:	nIdLinP	Identificador del producto acabado.
	NUnidades	Unidades a tratar.
	SFecha	Fecha de fabricación/consumo.
	STipoStock	“S” o “P”

Procedure FabricarProductoSinConsumir(nIdLinP,nUnidades:currency; SFecha, sTipoStock:string);		
Fabrica las unidades indicadas sin consumir componentes ni tiempos.		
Parámetros:	nIdLinP	Identificador del producto acabado.
	NUnidades	Unidades a tratar.
	SFecha	Fecha de fabricación.
	STipoStock	“S” o “P”

Procedure ConsumirComponente(nIdLinP,nUnidades:currency; SFecha, sTipoStock:string);		
Consume las unidades del componente indicado.		
Parámetros:	NIdLinP	Identificador del componente.
	NUnidades	Unidades a consumir.
	SFecha	Fecha de consumo.
	STipoStock	“S” o “P”

Procedure ConsumirOperario(nIdLinP:currency; sCodEmp, sTiempo sFecha, sTipoStock:string);		
Consume dicho tiempo para el perfil de operario indicado.		
Parámetros:	NIdLinP	Identificador de la línea del operario.
	SCodEmp	Código de operario que realiza la acción; si va en blanco, coge por defecto el de la línea de la orden.
	STiempo	Tiempo a consumir (en formato hh:mm:ss).
	SFecha	Fecha de consumo.
	STipoStock	“S” o “P”

Procedure ConsumirMaquina(nIdLinP:currency; sCodMaq, sTiempo sFecha, sTipoStock:string);		
Consume dicho tiempo para el perfil de máquina indicado.		
Parámetros:	NIdLinP	Identificador de la línea de la máquina.
	SCodMaq	Código de máquina que realiza la acción; si va en blanco, coge por defecto el de la línea de la orden.
	STiempo	Tiempo a consumir (en formato hh:mm:ss).
	SFecha	Fecha de consumo.
	STipoStock	“S” o “P”

Procedure FabricarProductoEnCurso(nIdLinC, nUnidades:currency; SCodAlm:string);		
Fabrica un producto acabado de una fabricación que previamente habíamos hecho marcando como TipoStock “Trabajo en curso”. El identificador que tendremos que pasar (IdLinC) corresponde a la tabla HistProd y para encontrarlo tendremos que buscar en dicha tabla por el identificador de línea de producción (IdLinP).		
Parámetros:	NIdLinC	Identificador de la línea de consumo.
	nUnidades	Unidades a fabricar del producto acabado.
	SCodAlm	Almacén donde repercutirá la entrada de stock; si va en blanco, coge por defecto la línea de consumo.

Procedure ConsumirComponenteEnCurso(nIdLinC, nUnidades:currency; SCodAlm:string);

Consume un componente de una fabricación que previamente habíamos hecho marcando como TipoStock “Trabajo en curso”. El identificador que tendremos que pasar (IdLinC) corresponde a la tabla HistProd y para encontrarlo tendremos que buscar en dicha tabla por el identificador de línea de producción (IdLinP).		
Parámetros:	NIdLinC	Identificador de la línea de consumo.
	nUnidades	Unidades a consumir del componente.
	SCodAlm	Almacén donde repercutirá la salida de stock; si va en blanco, coge por defecto la línea de consumo.

Procedure ConsumirMaquinaEnCurso(nIdLinC:currency; sTiempo:string)		
Consume un tiempo de una operación que previamente habíamos hecho marcando como TipoStock “Trabajo en curso”. El identificador que tendremos que pasar (IdLinC) corresponde a la tabla HistProd y para encontrarlo tendremos que buscar en dicha tabla por el identificador de línea de producción (IdLinP).		
Parámetros:	NIdLinC	Identificador de la línea de consumo.
	STiempo	Tiempo a consumir de máquina.

Procedure ConsumirOperarioEnCurso(nIdLinC:currency; sTiempo:string)		
Consume un tiempo de una operación que previamente habíamos hecho marcando como TipoStock “Trabajo en curso”. El identificador que tendremos que pasar (IdLinC) corresponde a la tabla HistProd y para encontrarlo tendremos que buscar en dicha tabla por el identificador de línea de producción (IdLinP).		
Parámetros:	NIdLinC	Identificador de la línea de consumo.
	STiempo	Tiempo a consumir de operario.

Procedure FabricarConsumirOperacionEnCurso(nIdLogP:currency)		
Fabrica y consume una operación múltiple (por ejemplo, fabricar una fase) que previamente habíamos hecho marcando como TipoStock “Trabajo en curso”. El identificador que tendremos que pasar (IdLogP) corresponde a la tabla LogProd.		
Parámetros:	nIdLogP	Identificador de la operación.

Procedure IndicarDetalle(nIdLogP, nIdLinP: Currency; nUnidades: Double; sNumSerie, sLote, sUbicacion, sFecCaduc: WideString)		
Permite indicar el número de serie / lote / etc cuando no se conoce el identificador de la línea de consumo / fabricación. Este caso se da cuando: 1. **Se trabaja contra stock**. Al consumir / fabricar no se puede guardar hasta que no se ha indicado el detalle; pero como el método fabicar consumir no devuelve el identificador de la línea de consumo (se puede estar fabricando toda una estructura, lo que significa que en realidad se están añadiendo muchas líneas de consumo) solo podemos indicar el IdLinp. El IdLogP se deja a 0 ya que estamos indicando el detalle en la última operación (la que acabamos de hacer). 2. **Se trabaja en curso**. Al consumir / fabricar en curso no se puede guardar hasta que no se ha indicado el detalle; pero como el método fabicar consumir no devuelve el identificador de la línea de consumo (se puede estar fabricando toda una estructura, lo que significa que en realidad se están añadiendo muchas líneas de consumo) solo podemos indicar el IdLinp. El IdLogP se puede dejar a 0 cuando estamos indicando el detalle en la última operación (la que acabamos de hacer).		
Parámetros:	nIdLogP	Identificador de la operación. Si se indica 0, se buscará el último trabajo
	nIdLinP	Identificador de la línea de producción
	nUnidades	Unidades del detalle
	sNumSerie	Número de serie del detalle
	sLote	Lote del detalle
	sUbicacion	Ubicación del detalle
	sFecCaduc	Fecha de caducidad del detalle.

Procedure AnadirDetalle(nIdLinC, nNumGrupo: Currency; nUnidades: Double; sNumSerie, sLote, sUbicacion, sFecCaduc: WideString)		
Permite indicar el número de serie / lote / etc cuando se conoce el identificador de la línea de consumo / fabricación. Este caso se da cuando se está cambiando el detalle de un trabajo previamente guardado.		
Parámetros:	nIdLinC	Identificador de la línea de consumo (HISTPROD)
	nNumGrupo	Número de grupo del consumo (HISTPROD)
	nUnidades	Unidades del detalle
	sNumSerie	Número de serie del detalle
	sLote	Lote del detalle
	sUbicacion	Ubicación del detalle
	sFecCaduc	Fecha de caducidad del detalle.

Procedure CambiarDetalle(nIdLinC, nNumLin: Currency; nUnidades: Double; sNumSerie, sLote, sUbicacion, sFecCaduc: WideString)		
Permite cambiar el detalle.		
Parámetros:	nIdLinC	Identificador de la línea de consumo (HISTPROD)

	nNumLin	Número de la línea (HISTPROD)
	nUnidades	Unidades del detalle
	sNumSerie	Número de serie del detalle
	sLote	Lote del detalle
	sUbicacion	Ubicación del detalle
	sFecCaduc	Fecha de caducidad del detalle.

Procedure BorrarDetalle(nIdLinC, nNumLin: Currency)		
Permite borrar el detalle.		
Parámetros:	nIdLinC	Identificador de la línea de consumo (HISTPROD)
	nNumLin	Número de la línea (HISTPROD)

Procedure TerminarOrden(sFecha:string)		
Termina todos los artículos a fabricar, cierra todos los trabajos y deshabilita cualquier acción sobre la orden.		
Parámetros:	sFecha	Fecha de finalización de la orden.

Procedure TerminarFase(nIdFase:currency; sFecha:string)		
Termina todos los artículos que dependan de la fase indicada deshabilitando cualquier acción sobre ésta.		
Parámetros:	nIdFase	Identificador de la fase
	SFecha	Fecha de finalización de la fase.

Procedure TerminarLinea(nIdLinP:currency; sFecha:string)		
Termina la línea indicada (sea artículo acabado, componente o tiempo) deshabilitando cualquier acción sobre ésta.		
Parámetros:	NIdLinp	Identificador de la línea
	SFecha	Fecha de finalización de la línea.

Procedure DeshacerFabricacionProducto(nIdLinC, nNumLin:currency)		
Deshace la fabricación de la línea de consumos indicada. Si la línea de consumo fue generada como TipoStock “Trabajo en curso”, el nNumLin puede tener dos valores: 0, con lo que borraría la línea de consumo y todas las evoluciones que hubiesen sobre ésta, o un valor distinto de 0, con que borraría la evolución identificada por el campo NUMLIN; si por el contrario la línea de consumo fue generada como TipoStock= ‘S’, siempre se tendrá que poner 0.		
Parámetros:	NIdLinC	Identificador de la línea de consumo
	nNumLin	Identificador de detalle de consumo.

Procedure DeshacerConsumoComponente(nIdLinC, nNumLin:currency)		
Deshace la línea de consumos indicada. Si la línea de consumo fue generada como TipoStock "Trabajo en curso", el nNumLin puede tener dos valores : 0, con lo que borraría la línea de consumo y todas las evoluciones que hubiesen sobre ésta, o un valor distinto de 0, con que borraría la evolución identificada por el campo NUMLIN; si por el contrario la línea de consumo fue generada como TipoStock= 'S', siempre se tendrá que poner 0.		
Parámetros:	NIdLinC	Identificador de la línea de consumo
	nNumLin	Identificador de detalle de consumo.

Procedure DeshacerConsumoOperario(nIdLinC, nNumLin:currency)		
Deshace la línea de consumos indicada. Si la línea de consumo fue generada como TipoStock "Trabajo en curso", el nNumLin puede tener dos valores : 0, con lo que borraría la línea de consumo y todas las evoluciones que hubiesen sobre ésta, o un valor distinto de 0, con que borraría la evolución identificada por el campo NUMLIN; si por el contrario la línea de consumo fue generada como TipoStock= 'S', siempre se tendrá que poner 0.		
Parámetros:	NIdLinC	Identificador de la línea de consumo
	nNumLin	Identificador de detalle de consumo.

Procedure DeshacerConsumoMaquina(nIdLinC, nNumLin:currency)		
Deshace la línea de consumos indicada. Si la línea de consumo fue generada como TipoStock "Trabajo en curso", el nNumLin puede tener dos valores: 0, con lo que borraría la línea de consumo y todas las evoluciones que hubiesen sobre ésta, o un valor distinto de 0, con que borraría la evolución identificada por el campo NUMLIN; si por el contrario la línea de consumo fue generada como TipoStock= 'S', siempre se tendrá que poner 0.		
Parámetros:	NIdLinC	Identificador de la línea de consumo
	NNumLin	Identificador de detalle de consumo.

Procedure DeshacerOperacion(nIdLogP:currency)		
Deshace las líneas de consumos indicadas por este identificador de operación múltiple (campo IdLogP de la tabla LogProd).		
Parámetros:	NIdLogP	Identificador de la operación.

Procedure DeshacerTerminarOrden
Deshace, si existe, la operación múltiple de terminar la orden.

Procedure DeshacerTerminarFase(nIdFase:currency)		
Deshace el terminar de la fase indicada (pero no sus hijos), con lo que pueden añadirse elementos a ésta.		
Parámetros:	nIdFase	Identificador de la fase.

Procedure DeshacerTerminarLinea(nIdLinP:currency)		
Deshace el terminar de la línea indicada (pero no sus posibles hijos), con lo que se puede modificar ésta.		
Parámetros:	nIdFase	Identificador de la fase.

Function NuevoTrabajo(sFecha, sMotivo:string):Currency		
Crea un nuevo parte de trabajo en la orden actual; devuelve el identificador de trabajo generado (IDTRAB).		
Parámetros:	SFecha	Fecha del trabajo.
	SMotivo	Motivo del trabajo.

Procedure AbrirTrabajo(nIdTrab:Currency)		
Abre el trabajo indicado, lo que habilita las modificaciones sobre él.		
Parámetros:	nIdTrab	Identificador de trabajo

Procedure CerrarTrabajo(nIdTrab:Currency)		
Cierra el trabajo indicado, lo que deshabilita las modificaciones sobre él.		
Parámetros:	NIdTrab	Identificador de trabajo

Procedure BorrarTrabajo(nIdTrab:Currency)		
Borra el trabajo indicado si no tiene movimientos.		
Parámetros:	nIdTrab	Identificador de trabajo

Procedure CambiarUnidadesProducto(nIdLinP, nUnidades:Currency)		
Modifica las unidades a fabricar de un producto acabado.		
Parámetros:	NIdLinP	Identificador de línea
	NUnidades	Unidades a fabricar

Procedure ActualizarCostesComponentes)
Recalcula los costes de todos los componentes de la orden activa.

35.- OBJETO EXPEDIENTES

Interface Expedientes
property ActivarFacturado: WordBool; **property** ActivarPreFactura: WordBool; **property** Estado: EstadoMaestro readonly; **property** AsStringCab[const sCampo: WideString]: WideString; **property** AsFloatCab[const sCampo: WideString]: Double; **property** AsIntegerCab[const sCampo: WideString]: Integer; **property** AsBooleanCab[const sCampo: WideString]: WordBool; **property** AsStringLin[const sCampo: WideString]: WideString; **property** AsFloatLin[const sCampo: WideString]: Double; **property** AsIntegerLin[const sCampo: WideString]: Integer; **property** AsBooleanLin[const sCampo: WideString]: WordBool; **property** AsVariantCab[const sCampo: WideString]: OleVariant; **property** AsVariantLin[const sCampo: WideString]: OleVariant; **property** Cerrado: WordBool; **property** OmitirMensajes: WordBool; **procedure** Iniciar; **procedure** Acabar; **procedure** Nuevo(const Cliente: WideString; const Fecha: WideString); **procedure** Modifica(IDEXPE: Currency); **function** Anade: Currency; **procedure** Borra(IDEXPE: Currency); **procedure** Cancela; **procedure** NuevaLinea(TIPO: LineaExpediente); **procedure** EditarLinea(TIPO: LineaExpediente; NUMLIN: Currency); **procedure** BorrarLinea(TIPO: LineaExpediente; NUMLIN: Currency); **procedure** AnadirLinea; **procedure** CancelaLin; **procedure** Facturar(IDEXPE: Currency; Factura: WordBool); **procedure** PreFactura(IDEXPE: Currency); **procedure** InicioFactBatch(const TIPO: WideString); **procedure** AnadeExpeBatch(IDEXPE: Currency); safecall; **function** FacturarBatch: OleVariant; safecall; **procedure** ContabilizacionLinea(Tipo: LineaExpediente; NumLin: Currency; FechaContabilizacion: TDateTime); **procedure** AnularContabilizacionLinea (Tipo: LineaExpediente; NumLin: Currency);

Propiedades	Tipo	Descripción
AsStringCab	Tabla(string)	Asigna/lee valores a/de los campos del registro

		nuevo/actual.
AsIntegerCab	Tabla(Integer)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsFloatCab	Tabla(Float)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsBooleanCab	Tabla(Lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsCurrencyCab	Tabla(Currency)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsStringLin	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsIntegerLin	Tabla(Integer)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsFloatLin	Tabla(Float)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsCurrencyLin	Tabla(Currency)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsBooleanLin	Tabla(Lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
Estado	EstadoMaestro	Devuelve el estado en que se encuentra el objeto
ActivarFacturado	Tabla(string)	Flag que se asigna para poderse facturar el expediente
ActivarPreFactura	Tabla(Integer)	Flag que se asigna para porder realizar la prefactura
AsVariantCab	Tabla(Variant)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsVariantLin	Tabla(Variant)	Asigna/lee valores a/de los campos del registro nuevo/actual.
Cerrado	Tabla(Boolean)	Permite indicar si el expediente está cerrado.
OmitirMensajes	Tabla(Boolean)	Permite indicar si queremos que se muestren los mensajes de aviso o decisión que genere el expediente.

*EstadoMaestro: Ver objeto Maestro.

Método	Tipo	Descripción
Iniciar	Procedimiento	Inicia el objeto
Acabar	Procedimiento	Cierra el objeto
Nuevo	Procedimiento	Inicia un nuevo expediente
Borra	Procedimiento	Borra el expediente
Modifica	Procedimiento	Prepara el expediente para poderlo modificar.
Cancela	Procedimiento	Cancela la edición del expediente
Anade	Función	Añade los datos editados del expediente a la base de datos y devuelve el identificador con

		el que se almacenará.
NuevaLinea	Procedimiento	Inicia una nueva línea
AnadirLinea	Procedimiento	Añade la línea
CancelaLin	Procedimiento	Cancela la línea en edición
EditarLinea	Procedimiento	Permite editar una línea ya existente del expediente.
BorrarLinea	Procedimiento	Permite borrar una línea del expediente.
Facturar	Procedimiento	Permite facturar el expediente
PreFactura	Procedimiento	Permite realizar al prefactura
InicioFactBatch	Procedimiento	Inicia el proceso de generar facturas a partir de expedientes en “batch”
AnadeExpeBatch	Procedimiento	Permite añadir un expediente a la lista de expedientes a facturar en “batch”
FacturarBatch	Función	Realiza la facturación en “batch” de los expedientes que hayan sido añadidos con el método AnadeExpeBatch.
ContabilizacionLinea	Procedimiento	Permite la contabilización de la línea.
AnularContabilizacionLinea	Procedimiento	Anula la contabilización de la línea.

Las primeras funciones de la lista que se presentan a continuación se utilizan para **asignar / leer valores de la cabecera o de las líneas del expediente**.

Deben tenerse en cuenta algunas normas importantes.

- Si el nombre del campo no existe se producirá un error.
- Si el valor asignado al campo especificado no es correcto, no se producirá un error. Este se producirá al intentar guardar el documento o la línea.

Hay que tener en cuenta lo siguiente:

- Los campos fecha se han de asignar como cadenas de texto como ‘01/01/2003’
- Uso en Delphi: a3ERPACTIVEXDocumento.AsStringCab[‘CodCli’]
- Uso en Visual Basic: a3ERPACTIVEXDocumento. AsStringCab (‘CodCli’)
- Las lineas del expediente se dividen en 4 tipos (ctSuplidos (entero 0), ctTrabajo (entero 1), ctProvision (entero 2), ctHonorario (entero 3).

property AsStringCab[sCampo:String]: String		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerCab[sCampo:String]: Integer		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatCab[sCampo:String]: Double		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanCab[sCampo:String]: Boolean		
A través de esta propiedad podemos asignar valor a los campos de la cabecera especificando que su tipo es lógico.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsStringLin[sCampo:String]: String		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsIntegerLin[sCampo:String]: Integer		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatLin[sCampo:String]: Double		
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanLin[sCampo:String]: Boolean
A través de esta propiedad podemos asignar valor a los campos de la línea especificando que su tipo es lógico.

Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsVariantCab[sCampo:String]: Boolean		
A través de esta propiedad podemos asignar valor a los campos de la cabecera dejando que se convierta a su tipo de manera implícita. Leyendo esta propiedad podemos saber si el valor de l campo es NULL.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	OleVariant	Valor del campo

property AsVariantLin[sCampo:String]: Boolean		
A través de esta propiedad podemos asignar valor a los campos de la linea dejando que se convierta a su tipo de manera implícita. Leyendo esta propiedad podemos saber si el valor de l campo es NULL.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	OleVariant	Valor del campo

A continuación, se presentan las **propiedades y métodos necesarios para crear, borrar y modificar expedientes.**

property Estado: EstadoMaestro		
Esta propiedad nos indica el estado del objeto. Posibles estados: ESTM_NOACTIVO: Expediente cerrado. No se puede usar. ESTM_ACTIVO: Expediente activo. Podemos añadir, borrar, modificar. ESTM_NUEVO: Expediente en estado de inserción de un nuevo registro. ESTM_EDICION: Expediente en estado modificación, podemos asignar valores a los campos.		
Valor retornado	EstadoMaestro	Retorna el estado del Expediente.

Constante: Lineaexpediente		
Estas constantes, se utiliza para indicar el tipo de linea de expediente que queremos tratar. Posibles estados: CTSUPLIDOS: Indicamos que la linea es de suplidos. Su valor equivalente es 0 (tipo de datos entero) CTTRABAJO: Indicamos que la linea es de trabajo. Su valor equivalente es 1 (tipo de datos entero) CTPROVISION: Indicamos que la linea es de provisión. Su valor equivalente es 2 (tipo de datos entero) CTHONORARIO: Indicamos que la linea es de honorarios. Su valor equivalente es 3 (tipo de datos entero)		
Valor retornado	EstadoMaestro	Retorna el estado del documento.

procedure Iniciar;
Este procedimiento es el que reserva los recursos necesarios para que el expediente pueda ser usado. Al iniciar el expediente el estado cambia de ESTM_NOACTIVO a ESTM_ACTIVO.

procedure Acabar;
Este procedimiento es el que libera los recursos utilizados por el objeto. Al acabar el expediente el estado cambia de ESTM_ACTIVO a ESTM_NOACTIVO.

procedure Nuevo(Cliente: WideString; Fecha: WideString);		
Inserta un nuevo registro y los prepara para ser editado. Pasa de estado ESTM_ACTIVO a ESTM_NUEVO.		
Parámetros:	Cliente	Código del cliente
	Fecha	Fecha del expediente (string formato dd/mm/aaaa)

procedure Modifica(IDEXPE: Currency);		
Pone el documento que tiene el identificador indicado en edición.		
Parámetros:	IdExpe	Identificador del expediente

Procedure Anade;
Almacena los valores de los campos en la base de dato con los cambios introducidos por el programador. Genera las repercusiones del expediente (si procede). Pasa de los estados ESTM_NUEVO o EST_EDICION a ESTM_ACTIVO.

Procedure Cancela;
Cancela la edición del expediente. Pasa de los estados ESTM_NUEVO o EST_EDICION a ESTM_ACTIVO.

procedure Borra Borra(IDEXPE: Currency);		
Elimina el documento con el identificador indicado eliminando también sus repercusiones en stocks, contabilidad, cartera, estadísticas e IVA.		
Parámetros:	IdExpe	Identificador del expediente

Procedure NuevaLinea(TIPO: LineaExpediente);		
Crea una línea nueva del tipo especificado por el parámetro. (Ver constantes LineaExpediente) A partir de ese momento podremos editar los valores de la línea.		
Parámetros:	Tipo	Tipo de linea de expediente. (Ver constantes LineaExpediente)

Procedure AnadirLinea;
Almacena los valores de los campos de la línea en la base de datos.

Procedure CancelaLin;
Cancela la edición de la línea activa.

procedure EditarLinea(TIPO: LineaExpediente; NUMLIN: Currency);		
Pone la línea indicada en el parámetro NumLina en edición, preparada para editar el contenido de sus campos. Al final de las modificaciones deberá llamarse a AnadirLinea. Hay que indicarle el tipo para saber que linea editar.		
Parámetros:	Tipo	Tipo de linea de expediente. (Ver constantes LineaExpediente)
	NumLin	Identificador de la línea. Se corresponde con el campo NUMLIN del fichero __LINEEXPE. Para localizarlo debe realizarse una consulta de este campo (directamente a la base de datos) en los registros con IdEXPE = a3ERPACTIVEXExpediente.AsFloatLin['IdExpe'] y con el campo TIPLIN = a3ERPACTIVEXExpediente.AsStringLin['TipLin'] donde su contenido es: G => Suplidos / T => Trabajo / R=> Provisión / H => Honorarios

Procedure BorrarLinea(TIPO: LineaExpediente; NUMLIN: Currency);		
Se situa en la linea indicada por los parámetros Tipo, NumLin y la borra.		
Parámetros:	Tipo	Tipo de linea de expediente. (Ver constantes LineaExpediente)
	NumLin	Identificador de la línea. Se corresponde con el campo NUMLIN del fichero __LINEEXPE. Para localizarlo debe realizarse una consulta de este campo (directamente a la base de datos) en los registros con IdEXPE = a3ERPACTIVEXExpediente.AsFloatLin['IdExpe'] y con el campo TIPLIN = a3ERPACTIVEXExpediente.AsStringLin['TipLin'] donde su contenido es: G => Suplidos / T => Trabajo / R=> Provisión / H => Honorarios

Procedure Facturar(IDEXPE: Currency; Factura: WordBool);		
Indicamos que deseamos facturar/desfacturar el expediente mediante el campo IdExpe y mediante el booleano Factura.		
Parámetros:	IdExpe	Identificar del expediente
	Factura	Es un boleando donde si su valore es cierto facturará el expediente si éste no está facturado y si su valor es falso desfacturará el expediente si éste estaba facturado.

PreFactura(IDEXPE: Currency);		
Indicamos que deseamos prefacturar el expediente. (Impresión del expediente)		
Parámetros:	IdExpe	Identificar del expediente

InicioFactBatch(const TIPO: WideString)		
Inicia el proceso de generar facturas a partir de expedientes en “batch”. Este método no requiere que se invoque previamente al método NUEVO o MODIFICA.		
Parámetros:	Tipo	Indica como queremos que se generen las facturas. A dia de hoy se pueden asignar dos valores diferentes para este parámetro: “UNA_A_UNA” → Indica que queremos generar una factura por cada expediente “AGP_CODCLIFAC_MONEDA_ANALITICA” → Indica que las facturas se generarán agrupadas por CODCLIFAC, MONEDA y ANALITICA.

AnadeExpeBatch (IDEXPE: Currency)		
Permite añadir un expediente a la lista de expedientes a facturar en “batch”. Este método no requiere que se invoque previamente al método NUEVO o MODIFICA.		
Parámetros:	IdExpe	Identificador del expediente

FacturarBatch		
Realiza la facturación en “batch” de los expedientes que hayan sido añadidos con el método AnadeExpeBatch. Este método no requiere que se invoque previamente al método NUEVO o MODIFICA.		
Valor retornado:	OleVariant	Se devuelve un variant con la lista de los IDFAC generados durante el proceso “batch”

ContabilizacionLinea (Tipo: LineaExpediente; NumLin: Currency; FechaContabilizacion: TDateTime)		
Aunque el expediente esté facturado, permite indicar que una línea de expediente. Sólo es aplicable a las líneas de suplidos y provisiones.		
Parámetros:	Tipo	Unos de los valores del tipo Lineaexpediente
	NumLin	Número de línea
	FechaContabilizacion	Fecha de contabilización

AnularContabilizacionLinea (Tipo: LineaExpediente; NumLin: Currency);		
Aunque el expediente esté facturado, permite indicar que una línea de expediente no tiene repercusiones contables. Sólo es aplicable a las líneas de suplidos y provisiones.		
Parámetros:	Tipo	Unos de los valores del tipo Lineaexpediente
	NumLin	Número de línea

36.- OBJETO CUOTAS

Objeto que permite la **creación, modificación y borrado de cuotas de facturas y albaranes**.

Interface Cuotas
property AsStringCab[const sCampo: WideString]: WideString; **property** AsCurrencyCab[const sCampo: WideString]: Currency; **property** AsFloatCab[const sCampo: WideString]: Double; **property** AsIntegerCab[const sCampo: WideString]: Integer; **property** AsBooleanCab[const sCampo: WideString]: WordBool; **property** AsVariantCab[const sCampo: WideString]: Variant; **property** OmitirMensajes:WordBool; **property** IdLinea: Currency; **property** AsStringLin[const sCampo: WideString]: WideString; **property** AsCurrencyLin [const sCampo: WideString]: Currency; **property** AsFloatLin [const sCampo: WideString]: Double; **property** AsIntegerLin [const sCampo: WideString]: Integer; **property** AsBooleanLin [const sCampo: WideString]: WordBool; **property** AsVariantLin [const sCampo: WideString]: Variant; **procedure** Iniciar; **procedure** Acabar; **procedure** Nuevo; **procedure** Modificar(IDAuto: Currency); **procedure** Borrar(IDAuto: Currency); **function** Guarda: Currency; **procedure** Cancelar; **procedure** ActivarMesCab(Mes: Integer); **procedure** DesactivaMesCab(Mes: Integer); **procedure** NuevaLinea(Tipo: TipoLineaCuota); **procedure** EditaLinea(IdAuto: Currency); **procedure** BorraLinea(IdAuto: Currency); **function** GuardaLinea: Currency; **procedure** CancelaLinea; **function** IrPrimeraLinea: WordBool; **function** IrSiguienteLinea : WordBool;

Las primeras funciones de la lista que se presentan a continuación se utilizan para **asignar / leer valores de la cuota.**

Deben tenerse en cuenta algunas normas importantes:

- Si el nombre del campo no existe se producirá un error.
- Si el valor asignado al campo especificado no es correcto, no se producirá un error. Este se producirá al intentar guardar el documento o la línea.

Hay que tener en cuenta lo siguiente:

- Los campos fecha se han de asignar como cadenas de texto como '01/01/2003'
- Uso en Delphi: a3ERPACTIVEXCuotas.AsString['CodCli']
- Uso en Visual Basic: a3ERPACTIVEXCuotas.AsString ('CodCli')

property AsStringCab[sCampo:String]: String		
A través de esta propiedad podemos asignar valor a los campos especificando que su tipo es string.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	String	Valor del campo

property AsCurrencyCab[sCampo:String]: Currency		
Con esta propiedad podemos asignar valor a los campos especificando que su tipo es money.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsIntegerCab[sCampo:String]: Integer		
Con esta propiedad podemos asignar valor a los campos especificando que su tipo es entero.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloatCab[sCampo:String]: Double		
A través de esta propiedad podemos asignar valor a los campos especificando que su tipo es double.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBooleanCab[sCampo:String]: Boolean		
A través de esta propiedad podemos asignar valor a los campos especificando que su tipo es lógico.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property AsCurrencyCab[sCampo:String]: Currency		
A través de esta propiedad podemos asignar valor a los campos especificando que su tipo es moneda.		
Parámetros:	sCampo	Nombre del campo
Valor retornado	Lógico	Valor del campo

procedure Iniciar;
Este procedimiento es el que reserva los recursos necesarios para que la cuota pueda ser usada. Al iniciar la cuota el estado cambia de ESTM_NOACTIVO a ESTM_ACTIVO.

procedure Acabar;
Este procedimiento es el que libera los recursos utilizados por el objeto. Al acabar la cuota el estado cambia de ESTM_ACTIVO a ESTM_NOACTIVO.

procedure Nuevo;
Inserta un nuevo registro y lo prepara para ser editado. Pasa de estado ESTM_ACTIVO a ESTM_NUEVO.

procedure Modificar(IDCuota: Currency);		
Pone la cuota que tiene el identificador indicado en edición.		
Parámetros:	IdExpe	Identificador de la cuota

Procedure Borrar(IdCuota: Currency);		
Borra la cuota del identificador indicado.		
Parámetros:	IdCuota	Identificador de la cuota

Procedure Cancelar;
Cancela la edición de la cuota. Pasa de los estados ESTM_NUEVO o EST_EDICION a ESTM_ACTIVO.

Funcion Guardar:currency		
Almacena los valores de los campos en la base de dato con los cambios introducidos por el programador. Pasa de los estados ESTM_NUEVO o EST_EDICION a ESTM_ACTIVO.		
Valor retornado	Currency	Devuelve el identificador de la cuota

Procedure ActivarMesCab (Mes:integer)		
Indicamos el mes (de 1 a 12) en que tiene que aplicarse la cuota		
Parámetros:	Mes	Mes aplicar cuota

Procedure DesactivarMesCab (Mes:integer)		
Indicamos el mes (de 1 a 12) en que tiene que anularse la cuota		
Parámetros:	Mes	Mes anular cuota

procedure NuevaLinea(Tipo: TipoLineaCuota);
Inserta una nueva línea de cuota del tipo especificado y la prepara para ser editada. Pasa de estado ESTM_ACTIVO a ESTM_NUEVO.

procedure EditaLinea(IdAuto: Currency);		
Pone la línea de cuota que tiene el identificador indicado como parámetro, en edición.		
Parámetros:	IdAuto	Identificador de la línea de la cuota

Procedure BorraLinea(IdAuto: Currency);		
Borra la línea de cuota que tiene el identificador indicado como parámetro.		
Parámetros:	IdAuto	Identificador de la línea de la cuota

Funcion GuardarLinea:currency		
Almacena los valores de los campos en la base de dato con los cambios introducidos por el programador. Pasa de los estados ESTM_NUEVO o EST_EDICION a ESTM_ACTIVO.		
Valor retornado	Currency	Devuelve el identificador de la línea de cuota

Procedure CancelaLinea;
Cancela la edición de la línea de cuota. Pasa de los estados ESTM_NUEVO o EST_EDICION a ESTM_ACTIVO.

function IrPrimeraLinea: WordBool;
Se situa en la primera línea de cuota. Devuelve verdadero si lo ha conseguido.

function IrSiguienteLinea: WordBool;
Se situa en la siguiente línea de cuota. Devuelve verdadero si lo ha conseguido. Útil para realizar un bucle por las líneas.

37.- OBJETO COBROPARCIALREMESA

Objeto que permite el cobro parcial de efectos de una remesa.

Interface CobroParcialRemesa
property ConRepercusiones: WordBool; **property** AlDescuento: WordBool; **property** CuentaBanco: WideString; **property** CuentaRiesgo: WideString; **property** Fecha: WideString; **property** FechaValor: WideString; **property** CodigoBanco: WideString; **property** CuentaGastos: WideString;

```
property Cambio: Double;
property Gastos: Double;
property Centro1: WideString;
property Centro2: WideString;
property Centro3: WideString;
property NumeroDocumento: WideString;
property UnAsiento: WordBool
procedure Iniciar(IdRemesa: Integer);
procedure AnadeEfecto(NumeroCartera: Currency; NumeroVencimiento: Integer);
procedure Procesar(PedirDatos: WordBool);
procedure Finalizar;
```

Las primeras funciones de la lista que se presentan a continuación se utilizan para **asignar / leer valores para realizar el cobro parcial de la remesa.**

Deben tenerse en cuenta algunas normas importantes:

- Si el nombre del campo no existe se producirá un error.
- Si el valor asignado al campo especificado no es correcto, no se producirá un error. Este se producirá al intentar guardar el documento o la línea.

Hay que tener en cuenta lo siguiente:

- Los campos fecha se han de asignar como cadenas de texto como '01/01/2003'
- Uso en Delphi: a3ERPACTIVEXCuotas.AsString['CodCli']
- Uso en Visual Basic: a3ERPACTIVEXCuotas.AsString ('CodCli')

Propiedades	Tipo	Descripción
ConRepercusiones	Boolean	Permite indicar si queremos el cobro con o sin repercusiones contables.
AlDescuento	Boolean	Permite indicar si queremos el cobro al descuento o al cobro.
CuentaBanco	String	Permite indicar la cuenta bancaria del asiento del cobro.
CuentaRiesgo	String	Permite indicar la cuenta bancaria de riesgo del asiento del cobro.
Fecha	String	Permite indicar la fecha efectiva del cobro y la fecha del asiento del cobro.
FechaValor	String	Permite indicar la fecha valor del cobro y la fecha valor del asiento del cobro.
CodigoBanco	String	Permite indicar el código de banco con el que se realiza el cobro.
CuentaGastos	String	Permite indicar la cuenta de gastos del asiento del cobro.
Cambio	Float	Permite indicar el cambio de la operación (para cobros en divisa).

Gastos	Currency	Permite indicar el importe de los gastos del asiento del cobro.
Centro1	String	Permite indicar el centro coste 1 del asiento del cobro.
Centro2	String	Permite indicar el centro coste 2 del asiento del cobro.
Centro3	String	Permite indicar el centro coste 3 del asiento del cobro.
NumeroDocumento	String	Permite indicar el número de documento del asiento del cobro.
UnAsiento	Boolean	Permite indicar si queremos un asiento por cada cobro parcial o todos los cobros parciales en el mismo asiento.

procedure Iniciar (IdRemesa:integer);		
Este procedimiento es el que reserva los recursos necesarios para que la remesa pueda ser usada. Al iniciar la cuota el estado cambia de ESTM_NOACTIVO a ESTM_ACTIVO.		
Parámetros:	Remesa	Identificador de la remesa

procedure Procesar(PedirDatos: WordBool);		
Realiza el cobro parcial de todos los efectos añadidos anteriormente.		
Parámetros:	PedirDatos	Permite mostrar pantalla de petición de datos de la operación.

procedure Finalizar;
Este procedimiento es el que libera los recursos utilizados por el objeto. Al acabar la cuota el estado cambia de ESTM_ACTIVO a ESTM_NOACTIVO.

Procedure AnadeEfecto(NumeroCartera: Currency; NumeroVencimiento: Integer);		
Indicamos que efectos de la remesa queremos cobrar parcialmente		
Parámetros:	NumeroCartera	Identificamos la cartera a cobrar
	NumeroVencimiento	**Identificamos el nº de vencimiento dentro del numerocartera**

38.- OBJETO PAGOPARCIALREMESA

Objeto que permite el pago parcial de efectos de una remesa.

Interface PagoParcialRemesa
property ConRepercusiones: WordBool; **property** CuentaBanco: WideString; **property** Fecha: WideString; **property** FechaValor: WideString; **property** CodigoBanco: WideString; **property** CuentaGastos: WideString; **property** Cambio: Double; **property** Gastos: Double; **property** Centro1: WideString; **property** Centro2: WideString; **property** Centro3: WideString; **property** NumeroDocumento: WideString; **property** UnAsiento: WordBool **procedure** Iniciar(IdRemesa: Integer); **procedure** AnadeEfecto(NumeroCartera: Currency; NumeroVencimiento: Integer); **procedure** Procesar(PedirDatos: WordBool); **procedure** Finalizar;

Las primeras funciones de la lista que se presentan a continuación se utilizan **para asignar / leer valores para realizar el pago parcial de la remesa.**

Deben tenerse en cuenta algunas normas importantes:

- Si el nombre del campo no existe se producirá un error.
- Si el valor asignado al campo especificado no es correcto, no se producirá un error. Este se producirá al intentar guardar el documento o la línea.

Hay que tener en cuenta lo siguiente:

- Los campos fecha se han de asignar como cadenas de texto como ‘01/01/2003’
- Uso en Delphi: a3ERPACTIVEXCuotas.AsString[‘CodCli’]
- Uso en Visual Basic: a3ERPACTIVEXCuotas.AsString (‘CodCli’)

Propiedades	Tipo	Descripción
ConRepercusiones	Boolean	Permite indicar si queremos el pago con o sin repercusiones contables.
CuentaBanco	String	Permite indicar la cuenta bancaria del asiento del pago.

Fecha	String	Permite indicar la fecha efectiva del pago y la fecha del asiento del pago.
FechaValor	String	Permite indicar la fecha valor del pago y la fecha valor del asiento del pago.
CodigoBanco	String	Permite indicar el código de banco con el que se realiza el pago.
CuentaGastos	String	Permite indicar la cuenta de gastos del asiento del pago.
Cambio	Float	Permite indicar el cambio de la operación (para pagos en divisa).
Gastos	Currency	Permite indicar el importe de los gastos del asiento del pago.
Centro1	String	Permite indicar el centro coste 1 del asiento del pago.
Centro2	String	Permite indicar el centro coste 2 del asiento del pago.
Centro3	String	Permite indicar el centro coste 3 del asiento del pago.
NumeroDocumento	String	Permite indicar el número de documento del asiento del pago.
UnAsiento	Boolean	Permite indicar si queremos un asiento por cada pago parcial o todos los pagos parciales en el mismo asiento.

procedure Iniciar (IdRemesa:integer);		
Este procedimiento es el que reserva los recursos necesarios para que la remesa pueda ser usada. Al iniciar la cuota el estado cambia de ESTM_NOACTIVO a ESTM_ACTIVO.		
Parámetros:	Remesa	Identificador de la remesa

procedure Procesar(PedirDatos: WordBool);		
Realiza el pago parcial de todos los efectos añadidos anteriormente.		
Parámetros:	PedirDatos	Permite mostrar pantalla de petición de datos de la operación.

procedure Finalizar;
Este procedimiento es el que libera los recursos utilizados por el objeto. Al acabar la cuota el estado cambia de ESTM_ACTIVO a ESTM_NOACTIVO.

Procedure AnadeEfecto(NumeroCartera: Currency; NumeroVencimiento: Integer);		
Indicamos que efectos de la remesa queremos pago parcialmente		
Parámetros:	NumeroCartera	Identificamos la cartera a pago
	NumeroVencimiento	**Identificamos el nº de vencimiento dentro del numerocartera**

39.- OBJETO ANULARCOBROPARCIALREMESA

Objeto que permite la **anulación del cobro** (realizado como cobro parcial de efectos de una remesa).

Interface AnularCobroParcialRemesa
procedure Iniciar(IdRemesa: Integer); **procedure** AnadeOperacion(Operacion: Currency); **procedure** Procesar; **procedure** Finalizar;

procedure Iniciar (IDRemesa:integer);		
Este procedimiento es el que reserva los recursos necesarios para que la remesa pueda ser usada. Al iniciar la cuota el estado cambia de ESTM_NOACTIVO a ESTM_ACTIVO.		
Parámetros:	Remesa	Identificador de la remesa

procedure Finalizar;
Este procedimiento es el que libera los recursos utilizados por el objeto. Al acabar la cuota el estado cambia de ESTM_ACTIVO a ESTM_NOACTIVO.

Procedure AnadeOperacion(Operacion: Currency);		
Indicamos que operaciones queremos anular		
Parámetros:	Operación	Identificador de la operación a anular

procedure Procesar;
Anula todas las operaciones indicadas en el método anterior

40.- OBJETO ANULARPAGOPARCIALREMESA

Objeto que permite la **anulación del pago** (realizado como pago parcial de efectos de una remesa).

Interface AnularPagoParcialRemesa
procedure Iniciar(IdRemesa: Integer); **procedure** AnadeOperacion(Operacion: Currency); **procedure** Procesar; **procedure** Finalizar;

procedure Iniciar (IDRemesa:integer);		
Este procedimiento es el que reserva los recursos necesarios para que la remesa pueda ser usada. Al iniciar la cuota el estado cambia de ESTM_NOACTIVO a ESTM_ACTIVO.		
Parámetros:	Remesa	Identificador de la remesa

procedure Finalizar;
Este procedimiento es el que libera los recursos utilizados por el objeto. Al acabar la cuota el estado cambia de ESTM_ACTIVO a ESTM_NOACTIVO.

Procedure AnadeOperacion(Operacion: Currency);		
Indicamos que operaciones queremos anular		
Parámetros:	Operación	Identificador de la operación a anular

procedure Procesar;
Anula todas las operaciones indicadas en el método anterior

41.- OBJETO ACCION

Este objeto permite **crear acciones de tipo cita, e-mail u otras** (genéricas). Sirven tanto para a3ERP de Windows como para el CRM (a3ERP .Net)

Interface Accion
Property AsString[const sCampo: WideString]: WideString; **property** AsFloat[const sCampo: WideString]: Double; **property** AsCurrency[const sCampo:WideString]:Currency; **property** AsInteger[const sCampo: WideString]: Integer; **property** AsBoolean[const sCampo: WideString]: WordBool; **property** AsVariant[const sCampo: WideString]: Variant; **property** Estado: EstadoMaestro **readonly**; **property** OmitirMensajes:WordBool; **procedure** Iniciar; **procedure** Acabar; **procedure** Nuevo(Accion:TipoAccion); **function** Guarda:integer; **procedure** Modifica(IdAccion:Integer); **procedure** Cancela; **procedure** Borra(IdAccion:Integer); **Enumerado** RelacionAccion_Entidad; **procedure** Relaciona(IDAccion: Integer; ID: Integer; Tipo: RelacionAccion_Entidad); **procedure** QuitarRelacion(IDAccion: Integer; ID: Integer; Tipo: RelacionAccion_Entidad); **procedure** QuitarRelacionConAcciones (IDAccion: Integer; Tipo: RelacionAccion_Entidad); **function** GetEntidadProcedencia(IDAccion: Integer): IAccionEntidadProcedencia;

Propiedades	Tipo	Descripción
AsString	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsFloat	Tabla(float)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsInteger	Tabla(integer)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsBoolean	Tabla(lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsVariant	Tabla(variant)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsCurrency	Tabla(currency)	Asigna/lee valores a/de los campos del registro nuevo/actual.
Estado	EstadoMaestro	Devuelve en que estado se encuentra el maestro.
OmitirMensajes	Lógico	Asignar valor para ocultar mensajes de información

TipoAccion (Tipo enumerado)		
	Valor	**Significado**
tipoaccionOtras	0	Para indicar que la acción es general.
tipoaccionCita	1	Para indicar que la acción es de tipo cita.
tipoaccionEmail	2	Para indicar que la acción es de tipo email.

procedure Iniciar;
Inicia el objeto acción y pasa al estado estM_ACTIVO

procedure Acabar;
Destruye los recursos usados para el objeto acción y pasa al estado estM_NOACTIVO.

procedure Nuevo(Accion:TipoAccion);
Pasa estado estM_NUEVO en el que se pueden asignar valores a los campos del registro. Hay que indicarle el tipo de acción que se desea crear.

function Guarda: integer
Inserta o modifica el registro actual de la acción. Te devuelve el IDACCION de la acción creada/modificada. Pasa al estado estM_ACTIVO después de guardar

procedure Modifica(IdAccion:Integer);
Pasa estado estM_EDICION en el que se pueden asignar valores a los; campos del registro. Se le indica el IDACCION para modifcar una acción.

procedure Cancela;
Cancela los cambios realizados. Tanto para un alta como para una modificación Pasa al estado estM_NOACTIVO, descartando el registro actual.

procedure Borrar(IdAccion:Integer);
Borra la acción pasada por parámetro. Hace referencia al campo IDACCION de las tablas de acciones. Pasa al estado estM_NOACTIVO despés de borrar la acción

Enumerado RelacionAccion_Entidad		
	Valor	**Significado**
LineaExpediente_	1	Para indicar que los objetos a relaccionar son una acción y una línea de expediente
Expediente_	2	Para indicar que **los** objetos a relaccionar son una acción y un expediente
Oportunidad_	3	Para indicar que los objetos a relaccionar son una acción y una oportunidad
Cartera_	4	Para indicar que los objetos a relaccionar son una acción y un efecto de cartera.
Rae_none	5	Por si la procedencia no esta identificada
raeLineaCuota [Nuevo 14.3.0.0]	6	Para indicar que los objetos a relaccionar son una acción y una línea de cuota.

IAccionEntidadProcedencia;		
Interfaz para comunicar información de la procedencia de la acción.		
Propiedades	**Tipo**	**Descripción**
IDEntidad	Currency	Id de la entidad de la que procede la accion
Entidad	RelacionAccion_Entidad	Tipo de entidad
IDAccion	Currency	Identificador de la accion
AditionalInfo	string	Texto informativo de la entidad de la que procede la acción, nombre de la entidad, tipo contable, serie y numero documento

procedure Relaciona(IDAccion: Integer; ID: Integer; Tipo: RelacionAccion_Entidad);
El método permite relacionar una entidad del ERP con una acción. Si la relación ya existe da una violación de “clave primaria”. • IDAccion: Identificador de la acción. • ID: Identificador de la entidad con la cual relacionaremos la acción. • RelacionAccion_Entidad: Naturaleza de la entidad referida en el parámetro anterior.

QuitarRelacion(IDAccion: Integer; ID: Integer; Tipo: RelacionAccion_Entidad);
El método permite eliminar la relación entre una entidad del ERP y una acción. Si la relación no existe no hace nada. • IDAccion: Identificador de la acción. • ID: Identificador de la entidad con la cual relacionaremos la acción. • RelacionAccion_Entidad: Naturaleza de la entidad referida en el parámetro anterior.

QuitarRelacionConAcciones (IDAccion: Integer; Tipo: RelacionAccion_Entidad);
El método permite eliminar todas las relaciones entre una entidad del ERP y todas las acciones asociadas. Si no existe ninguna relación no hace nada. • IDAccion: Identificador de la acción. • RelacionAccion_Entidad: Naturaleza de la entidad referida en el parámetro anterior.

GetEntidadProcedencia(IDAccion: Integer): IAccionEntidadProcedencia;
La función devuelve la interfaz IAccionEntidadProcedencia si la acción procede de alguna entidad. • IDAccion: Identificador de la acción. • IAccionEntidadProcedencia: Aporta información de la procedencia de la acción.

42.- OBJETO INMUEBLECLIENTE

Este objeto permite **indicar los inmuebles de un cliente**.

Interface Accion
procedure Iniciar(const CodCli: WideString); **function** AnadeInmueble(IDInmueble: integer): integer; **procedure** BorraInmueble(IDInmueble: integer); **function** ListaInmuebles: string; **procedure** Acabar;

procedure Iniciar;
Inicia el objeto InmuebleCliente para el cliente indicado. El cliente debe existir.

procedure Acabar;
Destruye los recursos usados.

AnadeInmueble(IDInmueble: integer): integer
Incluye el inmueble indicado en los inmuebles del cliente. El inmueble debe existir.

BorraInmueble(IDInmueble: integer)
Elimina el inmueble indicado de los inmuebles del cliente. El inmueble no es eliminado.

function ListaInmuebles: string;
Devuelve una lista separada por comas con los id de los inmuebles del cliente iniciado

43.- OBJETO INMUEBLEPROVEEDOR

Este objeto permite **indicar los inmuebles de un proveedor**.

Interface Accion
procedure Iniciar(const CodPro: WideString); **function** AnadeInmueble(IDInmueble: integer): integer; **procedure** BorraInmueble(IDInmueble: integer); **function** ListaInmuebles: string; **procedure** Acabar;

procedure Iniciar;
Inicia el objeto InmuebleProveedor para el proveedor indicado. El proveedor debe existir.

procedure Acabar;
Destruye los recursos usados.

AnadeInmueble(IDInmueble: integer): integer
Incluye el inmueble indicado en los inmuebles del proveedor. El inmueble debe existir.

BorraInmueble(IDInmueble: integer)
Elimina el inmueble indicado de los inmuebles del provedor. El inmueble no es eliminado.

function ListaInmuebles: string;
Devuelve una lista separada por comas con los id de los inmuebles del proveedor iniciado

44.- OBJETO CONTACTORELACION

Este objeto permite **crear, modificar, borrar y consultar las relaciones de un contacto con respecto a otra entidad** (cliente, proveedor, almacen, etc.)

Interface ContactoRelacion
Property AsString[const sCampo: WideString]: WideString; **property** AsFloat[const sCampo: WideString]: Double; **property** AsInteger[const sCampo: WideString]: Integer; **property** AsCurrency[const sCampo: WideString]: Currency; **property** AsBoolean[const sCampo: WideString]: WordBool; **property** AsVariant[const sCampo: WideString]: Variant; **property** Estado: EstadoMaestro **readonly**; **procedure** Iniciar; **procedure** Acabar; **procedure** Nuevo(IdContacto: Integer; TipoEntidad: ContactoRelacionEntidad; IdEntidad: String); **function** Guardar: Integer; **procedure** Editar(IdContactoRelacion: Integer); **procedure** Cancelar; **procedure** Borrar(IdContactoRelacion: Integer); **procedure** BorrarTodo(IdContacto: Integer);

Propiedades	Tipo	Descripción
AsString	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsFloat	Tabla(float)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsInteger	Tabla(integer)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsBoolean	Tabla(lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsVariant	Tabla(lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsCurrency	Tabla(lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
Estado	EstadoMaestro	Devuelve en que estado se encuentra la relación.

Métodos	Tipo	Descripción
Iniciar	Procedimiento	Pasa al estado estM_ACTIVO.
Acabar	Procedimiento	Cierra el gestor de relaciones. El estado queda como estM_NOACTIVO.
Nuevo	Procedimiento	Pasa al estado estM_NUEVO en el que se pueden asignar valores a los campos del registro. En **IdContacto** debe indicarse el identificador del contacto origen. En **TipoEntidad** debe informarse el tipo de entidad con la que se relaciona el contacto (Cliente, Proveedor, Representante, etc.). Es del tipo ContactoRelacionEntidad. En **IdEntidad** se indicará el código de la entidad. Ejemplo para asignar un contacto • a un cliente, el TipoEntidad es 1 y el IdEntidad es el CODCLI del cliente • a un proveedor, el TipoEntidad es 2 y el IdEntidad es el CODPRO del proveedor • a un cliente potencial, el TipoEntidad es 3 y el IdEntidad es el CODPOS del cliente potencial • a un representante, el TipoEntidad es 4 y el IdEntidad es el CODREP del representante • a un banco, el TipoEntidad es 5 y el IdEntidad es el CODBAN del banco • a un transportista, el TipoEntidad es 6 y el IdEntidad es el CODTRA del transportista • a un almacén, el TipoEntidad es 7 y el IdEntidad es el CODALM del almacén • a una dirección de entrega, el TipoEntidad es 8 y el IdEntidad es el IDDIRENT de la dirección de entrega • a una dirección de recepción, el TipoEntidad es 9 y el IdEntidad es el IDDIRENT de la dirección de recepción.
Editar	Procedimiento	Pasa al estado estM_EDICION en el que se pueden asignar valores a los campos del registro. En IdContactoRelacion debemos informar el identifcador de la relación.
Guardar	Función	Inserta o modifica la relación. Devuelve el identificador de la relación.
Cancelar	Procedimiento	Pasa al estado estM_ACTIVO, descartando el registro actual.
Borrar	Procedimiento	Borra la relación en curso.
BorrarTodo	Procedimiento	Borra todas las relaciones de un contacto.

45.- OBJETO OPORTUNIDADES

Este objeto permite **crear, modificar, borrar y consultar oportunidades**. Formado por una o más oportunidades.

Interface Oportunidades
Property Store: IConjuntoDatos;

Propiedades	Tipo	Descripción
Store	IConjuntoDatos	Devuelve una interfaz que permite gestionar las oportunidades. En este caso la propiedad Store, cumple con las interfaces IConjuntoDatos e IOportunidad.

46.- OBJETO CONJUNTODATOS

Este objeto **no debe instanciarse directamente**. Será devuelto por alguna otra interfaz.

Interface ConjuntoDatos
funtion Nuevo: IDatos; **funtion** Obtener(Id: Integer): IDatos; **procedure** Borrar(Datos: IDatos); **procedure** Guardar(Datos: IDatos);

Métodos	Tipo	Descripción
Nuevo	Función	Devuelve una nueva instancia del tipo IDatos.
Obtener	Función	Busca la instancia de IDatos que conicide con el ID.
Borrar	Procedimiento	Elimina el objeto que conincide con el parámetro IDatos.
Guardar	Procedimiento	Almacena la información del objeto parámetro IDatos.

47.- OBJETO DATOS

Este objeto **no debe instanciarse directamente**. Será devuelto por alguna otra interfaz.

Interface Datos
Property AsString[const sCampo: WideString]: WideString; **property** AsFloat[const sCampo: WideString]: Double; **property** AsInteger[const sCampo: WideString]: Integer; **property** AsBoolean[const sCampo: WideString]: WordBool; **property** IsClear[const sCampo: WideString]: WordBool; **procedure** Clear(const sCampo: WideString);

Propiedades	Tipo	Descripción
AsString	Tabla(string)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsFloat	Tabla(float)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsInteger	Tabla(integer)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsBoolean	Tabla(lógico)	Asigna/lee valores a/de los campos del registro nuevo/actual.
IsClear	Tabla(lógico)	Nos indica si la propiedad parámetro está vacía

Métodos	Tipo	Descripción
Clear	Procedimiento	Vacía la propiedad parámetro.

48.- OBJETO DATOSEXT

Este objeto **extiende IDatos y no debe instanciarse directamente**. Será devuelto por alguna otra interfaz.

Interface Datos
property Value[const sCampo: WideString]: Variant;

Propiedades	Tipo	Descripción
Value	Tabla()	Asigna/lee valores a/de los campos del registro nuevo/actual.

49.- OBJETO DATOSEXT2

Este objeto **extiende IDatosExt y no debe instanciarse directamente**. Será devuelto por alguna otra interfaz.

Interface DATOSEXT2 [Nuevo 14.0.0.0]
property AsCurrency[const sCampo: WideString]: Currency; **property** AsDateTime[const sCampo: WideString]: TDateTime;

Propiedades	Tipo	Descripción
AsCurrency	Tabla(Currency)	Asigna/lee valores a/de los campos del registro nuevo/actual.
AsDateTime	Tabla(TDateTime)	Asigna/lee valores a/de los campos del registro nuevo/actual.

50.- OBJETO OPORTUNIDAD

Este objeto **contiene los datos de la oportunidad y sus líneas.**

Interface Oportunidades
Property Datos**:** IDatos**;**
Property Lineas: IConjuntoLineas;

Propiedades	**Tipo**	**Descripción**
Datos	IDatos	Información sobre la cabecera de la oportunidad.
Lineas	IConjuntoLineas	Devuelve una interfaz que permite gestionar las líneas de una oportunidad. En este caso, los datos devueltos por los métodos de la interfaz IConjuntoLineas serán del tipo IOportunidad e IDatos.

51.- OBJETO CONJUNTODELINEAS

Este objeto **no debe instanciarse directamente**. Será devuelto por alguna otra interfaz.

Interface ConjuntoDatos
funtion Nueva: IDatos; **funtion** Linea(Index: Integer): IDatos; **procedure** Quitar(Datos: IDatos); **funtion** NumLineas: Integer; **funtion** Obtener(Id: Integer): IDatos; **procedure** Guardar(Datos: IDatos); **procedure** Vaciar;

Métodos	Tipo	Descripción
Nueva	Función	Devuelve una instancia del tipo IDatos.
Obtener	Función	Devuelve la instancia de IDatos que conicide con el ID.
Quitar	Procedimiento	Borra la línea que concida con el objeto parámetro.
Linea	Función	Devuelve la línea por índice.
NumLineas	Función	Devuelve el número de líneas
Guardar	Procedimiento	Guardar en la lista el objeto Datos.
Vaciar	Procedimiento	Elimina las líneas del conjunto.

52.- OBJETO LINEAOPORTUNIDAD

Este objeto **contiene la información de una línea de oportunidad**.

Interface LineaOportunidad
Property Datos: IDatos;

Propiedades	**Tipo**	**Descripción**
Datos	IDatos	Devuelve una interfaz que contiene la información de una línea de una oportunidad.

53.- OBJETO VINCULOSOPORTUNIDAD

Este objeto permite crear, modificar, borrar y consultar los vínculos de una oportunidad.

Interface VinculosOportunidad
Function Nuevo(IdOportunidad: Integer; Tipo: TipoVinculo): IVinculo; **Procedure** Borrar(Vinculo: IVinculo); **Function** ObtenerVinculos(IdOpoportunidad: Integer; Tipo: TipoVinculo): IIterador; **Procedure** Vaciar(IdOp: Integer; TipoVinculo: TipoVinculo); **Procedure** BorrarPorId(TipoVinculo: TipoVinculo; IdVinculo: Integer); **Procedure** Guardar(Vinculo: IVinculo);

Métodos	Tipo	Descripción
Nuevo	Función	Crea un vínculo en la oportunidad indicada y del tipo informado. Devuelve una instancia del Vínculo.
Borrar	Procedimiento	Borra el vínculo informado.
ObtenerVinculos	Función	Permite movermos por la lista de vínculos. Patrón iterador.
Vaciar	Procedimiento	Elimina los vínculos de determinado tipo de la oportunidad informada.
BorrarPorId	Procedimiento	Elimina el vínculo indicado.
Guardar	Procedimiento	Guarda el vínculo informado.

54.- OBJETO VÍNCULO

Este objeto contiene la información de un vínculo.

Interface Oportunidades
Property Datos**:** IDatos**;**
Property Tipo: TipoVinculo;

Propiedades	Tipo	Descripción
Datos	IDatos	Información sobre el vínculo.
Tipo	TipoVinculo	Tipo del vínculo.

55.- OBJETO VINCULOSGENERAL1302

Este objeto permite **crear, modificar, borrar y consultar los vínculos de cualquier tipo**. (Hasta version 13.02 inclusive, En versión 14.0 IVinculosGeneral1402)

Interface VinculosGeneral
Procedure Nuevo(const Vinculo: IVinculoGeneral); **Procedure Borrar(const Vinculo: IVinculoGeneral);** **Function ObtenerVinculos(const IDPadre: WideString; TipoPadre: TipoVinculoGeneral): IIterador;** **Procedure Vaciar(const IDPadre: WideString; TipoPadre: TipoVinculoGeneral);**

Métodos	Tipo	Descripción
Nuevo	Procedimiento	Crea un vínculo.
Borrar	Procedimiento	Borra el vínculo informado.
ObtenerVinculos	Función	Permite movermos por la lista de vínculos de determinado padre. Patrón iterador.
Vaciar	Procedimiento	Elimina los vínculos de determinado padre.

56.- OBJETO VINCULOSGENERAL1402

Extiende VinculosGeneral1302 y añade nuevos métodos.

Interface VinculosGeneral [Nuevo 14.0.0.0]
Function NuevoConId(const Vinculo: IVinculoGeneral): Integer; **Procedure Editar(const Vinculo: IVinculoGeneral);** **Function ObtenerVinculo(IdVinculo: Integer): IVinculoGeneral;** **Procedure BorrarPorId(IdVinculo: Integer);** **Procedure VaciarPorId(IdVinculo: Integer);**

Métodos	Tipo	Descripción
NuevoConId	Función	Crea un vínculo. Devuelve el identificador del vínculo creado.
Editar	Procedimiento	Edita el vínculo informado.
ObtenerVinculo	Función	Recupera la información del vínculo informado y devuelve una instancia.
BorrarPorId	Procedimiento	Elimina el vínculo indicado.
VaciarPorId	Procedimiento	Elimina los vínculos de determinado vínculo.

57.- OBJETO VINCULOSGENERAL1403

Extiende VinculosGeneral1402 y añade los siguientes métodos.

Interface VinculosGeneral [Nuevo 14.0.3.0]
function SeleccionarEntidad(const ParamOperVinc: IParametroOperacionVinculo): WideString; **procedure AbrirVinculo(const ParamOperVinc: IParametroOperacionVinculo; const Identificador: WideString);** **function ObtenerCaption(const ParamOperVinc: IParametroOperacionVinculo; const Identificador: WideString): WideString;** **function ObtenerListaTipoVinculo: IIterador;** **function ObtenerTipoVinculo**(const Entidad: WideString): TipoVinculoGeneral;

Métodos	Tipo	Descripción
SeleccionarEntidad	Función	Muestra la selección del tipo de vinculo informado. El tipo del vínculo se indica mediante la propiedad “TipoVinculo” de la interfaz IParametroOperacionVinculo.
AbrirVinculo	Procedimiento	Permite abrir el mantenimiento del elemento correspondiente con el tipo informado y su Identificador.
ObtenerCaption	Función	Obtener el Caption para un vínculo, con el Tipo de vinculo y su Identificador.
ObtenerListaTipoVinculo	Función	Obtener la lista de todos los Tipos de vinculos disponibles segun la gama actual.
ObtenerTipoVinculo	Función	Dado el nombre de una entidad (tabla) devuelve el tipo de vínculo enumerado.

58.- OBJETO VINCULOSGENERAL

Extiende VinculosGeneral1403 y añade los siguientes métodos.

Interface VinculosGeneral [Nuevo 14.0.4.0]
function ObtenerEntidad (TipoVinculo: TipoVinculoGeneral): WideString;

Métodos	Tipo	Descripción
ObtenerEntidad	Función	Dado el tipo de vínculo enumerado devuelve el nombre de una entidad (tabla).

59.- OBJETO PARAMETROOPERACIONVINCULO

Este objeto sólo debe utilizarse como parámetro en algunos métodos del objeto VínculosGeneral.

Interface ParametroOperacionVinculo [Nuevo 14.0.3.0]
property TipoVinculo: TipoVinculoGeneral

60.- OBJETO VÍNCULOGENERAL1302

Este objeto **contiene la información de un vínculo de cualquier tipo** (Hasta versión 13.02. En versión 14 se devuelve IVinculoGeneral).

Interface VinculoGeneral1302
Property TipoPadre: TipoVinculoGeneral;
Property TipoHijo: TipoVinculoGeneral;
Property IdPadre: WdieString;
Property IdHijo: WdieString;
Property A3DOC: WordBool;

Propiedades	**Tipo**	**Descripción**
TipoPadre	TipoVinculoGeneral	Tipo del vínculo padre. **A tener en cuenta:** Los tipos de vínculo "Carpeta", "Fichero" y "Exchange", no pueden utilizarse como vínculo padre.

TipoHijo	TipoVinculoGeneral	Tipo del vínculo hijo. --> **A tener en cuenta:** Los tipos de vínculo "Oportunidad" y "Acción" no pueden utilizarse como vínculos hijo. **[Nuevo 14.2.0.0]** Los elementos cuyo valor es igual o superior a 1000 no pueden utilizarse como vínculos hijo.
IdPadre	String	Identificador del padre. Hay que cuadrar el valor.
IdHijo	String	Identificador del hijo. Hay que cuadrar el valor.
A3Doc	Lógico	Indica si ha de subir a a3DOC.

61.- OBJETO VÍNCULOGENERAL

Extiende VinculoGeneral y **añade la propiedad Datos que permite leer/escribir en cualquier campo**.

Interface VinculoGeneral [Nuevo 14.0.0.0]
Property Datos: IDatos; **[Nuevo 14.0.0.0]**
Property Id: Integer; [Nuevo 14.0.0.0]

Propiedades	Tipo	Descripción
Datos	IDatos	Información sobre el vínculo.
Id	Integer	Identificador del vínculo.

62.- OBJETO SERVIR

Permite **servir líneas de documentos**. Debe usarse si hay doble unidad de stock.

Interface Servir
Procedure Servirlinea(ParametrosLinea: IServirLineaParametros)

Métodos	Tipo	Descripción
ServirLinea	Procedimiento	Sirve una línea utilizando los parámetros informados. El documento destino será el documento que cumpla esta interfaz.

63.- OBJETO SERVIRLINEAPARAMETROS

Permite **indicar la información de la línea a servir**. Esta interfaz debe implementarse desde fuera. A3ERPActiveX no ofrece una implementación de la misma.

Interface ServirLineaParametros
Property Linea: Integer; **Property** Bultos: Integer; **Property** Paquetes: Integer; **Property** UnidadesPrecio: Integer; **Property** UnidadesStock: Integer; **Property** NumeroSerie: String; **Property** Lote: String; **Property** Ubicacion: String; **Property** Almacen: String; **Property** FechaCaducidad: TDateTime;

Propiedades	Tipo	Descripción
Linea	Entero	Número de linea del documento.
Bultos	Entero	Bultos de la línea del documento.
Paquetes	Entero	Paquetes de la línea del documento.
UnidadesPrecio	Entero	Unidades de precio de la línea del documento.
UnidadesStock	Entero	Unidades de stock de la línea del documento.
NumeroSerie	Cadena	Número de serie de la línea del documento.
Lote	Cadena	Lote de la línea del documento.
Ubicacion	Cadena	Ubicación de la línea del documento.
Almacen	Cadena	Almacén de la línea del documento.
FechaCaducidad	Fecha	Fecha de caducidad de la línea del documento.

64.- OBJETO OPCIONESSERVIR

Permite **indicar determinadas opciones a la acción de servir**.

Interface OpcionesServir
Property MantenerAlmacen: Boolean

Propiedades	Tipo	Descripción
MantenerAlmacen	Lógico	Indica si debe respetarse el almancen al servir línea.

65.- OBJETO SERVIR2

Permite **servir documentos de forma completa**.

Interface Servir2
Procedure Iniciar(TipoOrigen: OrigenServir; IdOrigen: long); **Procedure** ServirCompletamente; **Procedure** Finalizar;

Métodos	Tipo	Descripción
Iniciar	Procedimiento	Prepara el proceso para servir completamente.
ServirCompletamente	Procedimiento	Sirve el documento completo desde el documento origen pasado por parámetro. El documento destino será el documento que cumpla esta interfaz.
Finalizar	Procedimiento	Finaliza la operación y aplica los cambios.

66.- OBJETO LINEDADETALLE

Permite indicar los detalles de una línea de documento.

Interface LineaDetalle
Procedure AnadirDetalle (ParametrosDetalle: ILineaDetalleParametros); **Procedure** CambiarDetalle(ParametrosDetalle: ILineaDetalleParametros);

Métodos	Tipo	Descripción
AnadirDetalle	Procedimiento	Permite añadir artículos con detalle.
CambiarDetalle	Procedimiento	Permte cambiar el detalle de un artículo con detalle.

67.- OBJETO LINEADETALLEPARAMETROS

Permite **indicar la información de la línea de detalle**. Esta interfaz debe implementarse desde fuera. A3ERPActiveX no ofrece una implementación de la misma.

Interface LineaDetalleParametros
Property Linea: Integer; **Property** Bultos: Integer; **Property** Paquetes: Integer; **Property** UnidadesPrecio: Integer; **Property** UnidadesStock: Integer; **Property** NumeroSerie: String; **Property** Lote: String; **Property** Ubicacion: String; **Property** Almacen: String;

```
Property FechaCaducidad: TDateTime;
Property PrcMedio: Double:
```

Propiedades	Tipo	Descripción
Linea	Entero	Número de la línea de detalle del documento.
Bultos	Entero	Bultos de la línea de detalle del documento.
Paquetes	Entero	Paquetes de la línea de detalle del documento.
UnidadesPrecio	Entero	Unidades de precio de la línea de detalle del documento.
UnidadesStock	Entero	Unidades de stock de la línea de detalle del documento.
NumeroSerie	Cadena	Número de serie de la línea de detalle del documento.
Lote	Cadena	Lote de la línea de detalle del documento.
Ubicacion	Cadena	Ubicación de la línea de detalle del documento.
Almacen	Cadena	Almacén de la línea de detalle del documento.
FechaCaducidad	Fecha	Fecha de caducidad de la línea de detalle del documento.
PrcMedio	Importe	Precio medio de la línea de detalle del documento.

68.- INTERFAZ IITERADOR

Interfaz que recorre una serie de elementos Idispatch que pueden implementar otras interfaces.

Métodos	Tipo	Descripción
HayMas	Bool	Devuelve si hay más elementos en la serie
Actual	IDispatch	Retorna el elemento actual

Ejemplo:

```
while (MiIterador.HayMas())
   MiLista.Add(MiIterador.Actual());
```

69.- INTERFAZ IPROGRAMAEXTERNO

Interfaz que cumple un elemento que alberga la configuración de un programa externo para una entidad y la posibilidad de realizar la llamada.

Propiedades	Tipo	Descripción
Descripcion	String	Descripción indicada en el mantenimiento de programas externos
Parametros	String	Lista, separada por puntos y comas, de campos que componen la clave de la tabla.
Ruta	String	Ruta indicada en el mantenimiento de programas externos.
Procedimiento	String	Procedimiento indicado en el mantenimiento de programas externos.

procedure Ejecutar(datos: IDatos);		
Realiza la llamada al programa externo.		
Parámetros:	datos, IDatos	Estructura IDatos con los valores de los campos de un registro de la tabla, para la cual se ha definido la llamada, sobre el cual se realiza la llamada.

70.- INTERFAZ IDATOS

Contenedor de datos tabulares, generalmente correspondientes a un registro de una tabla.

property AsString[Campo:String]: String		
A través de esta propiedad podemos asignar valor a los campos especificando que su tipo es string.		
Parámetros:	Campo	Nombre del campo
Valor retornado	String	Valor del campo

property AsInteger[Campo:String]: Integer		
A través de esta propiedad podemos asignar valor a los campos especificando que su tipo es entero.		
Parámetros:	Campo	Nombre del campo
Valor retornado	Integer	Valor del campo

property AsFloat[Campo:String]: Double		
A través de esta propiedad podemos asignar valor a los campos especificando que su tipo es double.		
Parámetros:	Campo	Nombre del campo
Valor retornado	Double	Valor del campo

property AsBoolean[Campo:String]: Boolean		
A través de esta propiedad podemos asignar valor a los campos especificando que su tipo es lógico.		
Parámetros:	Campo	Nombre del campo
Valor retornado	Lógico	Valor del campo

property IsClear[Campo:String]: Boolean [SOLO LECTURA]		
Con esta propiedad podemos conocer si un campo tiene valor asignado.		
Parámetros:	Campo	Nombre del campo
Valor retornado	Lógico	True si tiene valor indicado, False en caso contrario.

procedure Clear(Campo:String)		
Desasigna el valor de un determinado campo		
Parámetros:	Campo	Nombre del campo

71.- INTERFAZ IDATOSEXT: IDATOS

Extensión de IDatos que es capaz de devolver y asignar el contenido de un campo como valor Variant, Currency o TDateTime.

property Value[Campo:String]: OleVariant		
Permite el acceso y modificación de un campo mediante un valor Variant		
Parámetros:	**Parámetros:**	**Parámetros:**
Valor retornado	**Valor retornado**	**Valor retornado**

72.- INTERFAZ IDATOSEXT2: IDATOSEXT [NUEVO 14.0.0.0]

Extensión de IDatosExt que es capaz de devolver y asignar el contenido de un campo como valor Currency o TDateTime.

property AsCurrency[Campo:String]: Currency		
Permite el acceso y modificación de un campo mediante un valor Currency		
Parámetros:	Campo	Nombre del campo
Valor retornado	Currency	Valor del campo

property AsDateTime[Campo:String]: TDateTime		
Permite el acceso y modificación de un campo mediante un valor TDateTime		
Parámetros:	Campo	Nombre del campo
Valor retornado	TDateTime	Valor del campo

73.- INTERFAZ IDOCUMENTOCLONABLE [NUEVO 14.5.2.0]

Interfaz que cumple un elemento que permite realizar la clonación de un documento sobre el documento que estamos creando nuevo o editando
Los documentos a los que aplica son las ofertas, depósitos, pedidos, albaranes y facturas de compra y de venta.

- Si estamos realizando un nuevo documento, la clonación se realiza sobre ese documento.
- Si estamos editando un documento, la clonación se realiza sobre ese documento.

IDocumentoClonable		
Permite realizar la clonación de un documento sobre otro		
Procedimiento:	Clonar	Permite realizar la clonación de un documento sobre otro

Parametros	**Tipo**	**Descripción**
DocumentoOrigenTipo	String	Tipo del documento origen
DocumentoOrigenID	Currency	Identificador del documento origen
BorrarLineasDelDocumentoDestino	Booleano	Borrar las líneas del documento destino
ClonarVariablesDeDocumentos	Booleano	Clonar las variables de documentos

Ejemplo de uso (Delphi)

Clonar una factura de venta con identificador 3 sobre una nueva factura de compra.

```
procedure TestClonarDocumento();
begin
  var Documento: IFactura := coFactura.Create;
  Documento.Iniciar;
  try
    Documento.Nuevo(FormatDateTime('dd/mm/yyyy', Date), '1', False, False, False, True);
    var DocumentoClonable: IDocumentoConable := Documento as IDocumentoConable;
    DocumentoClonable.Clonar('FV', 3, True, True);
    Documento.Anade;
  finally
    Documento.Acabar;
  end;
end;
```

Clonar una factura de venta con identificador 3 sobre una factura de venta con el identificador indicado en ID.

```
procedure TestClonarDocumento(ID: Integer);
begin
  var Documento: IFactura := coFactura.Create;
  Documento.Iniciar;
  try
    Documento.Modifica(ID, False);
    var DocumentoClonable: IDocumentoConable := Documento as IDocumentoConable;
    DocumentoClonable.Clonar('FV', 3, True, True);
    Documento.Anade;
  finally
    Documento.Acabar;
  end;
end;
```

74.- INTERFAZ IDOCUMENTOCOPIABLE [NUEVO 14.5.2.0]

Interfaz que cumple un elemento que permite realizar la copia de un documento sobre el documento que estamos creando nuevo o editando
Los documentos a los que aplica son las ofertas, depósitos, pedidos, albaranes y facturas de compra y de venta.

- Si estamos realizando un nuevo documento, la clonación se realiza sobre ese documento.
- Si estamos editando un documento, la clonación se realiza sobre ese documento.

IDocumentoCopiable		
Permite realizar la copiar un documento sobre otro		
Procedimiento:	Copiar	Permite realizar la copia de un documento sobre otro

Parametros	**Tipo**	**Descripción**
DocumentoOrigenTipo	String	Tipo del documento
DocumentoOrigenID	Currency	Identificador del documento
BorrarLineasDelDocumentoDestino	Booleano	Borrar las líneas del documento destino
CopiarCondicionesFinancieras	Booleano	Copiar condiciones financieras del documento origen en el documento destino. Se copiarán del documento origen la forma de pago, documento de pago, descuento de pronto pago o recargo financiero y si éste afecta a la base imponible
CopiarLosPortes	Booleano	Copiar los portes del documento origen en el documento destino
CopiarPreciosYDescuentos	Booleano	Copiar precios y descuentos del documento origen en el documento destino
CopiarDescripcionesYObservacionesLineas	Booleano	Copiar descripciones y observaciones de las líneas del documento origen en el documento destino

Ejemplo de uso (Delphi)
Copiar una oferta de venta con identificador 3 sobre una nueva oferta de venta. *procedure TestCopiarDocumento();* *begin* *var DocumentoO: IOferta := coOferta.Create;* *DocumentoO.Iniciar;* *try* *DocumentoO.Nuevo(FormatDateTime('dd/mm/yyyy', Date), '1', False);* *var DocumentoCopiableO: IDocumentoCopiable := DocumentoO as IDocumentoCopiable;* *DocumentoCopiableO.Copiar('OV', 3, True, True, True, True, True);* *DocumentoO.Anade;* *finally* *DocumentoO.Acabar;* *end;*
Copiar una factura de venta con identificador 3 sobre una factura de venta con el identificador indicado en ID. *procedure TestClopiarDocumento(ID: Integer);* *begin* *var Documento: IFactura := coFactura.Create;* *Documento.Iniciar;* *try* *Documento.Modifica(ID, False);* *var DocumentoClonable: IDocumentoConable := Documento as IDocumentoConable;* DocumentoCopiableO.Copiar('OV', 3, True, True, True, True, True); *Documento.Anade;* *finally* *Documento.Acabar;* *end;* *end;*