ebimap

edicom business integrator mapping tool

Edicom Business Integrator Mapping tool
Anexo - Funciones EdiwinScript

edicom

connecting business

Título del documento:	EBIMAP - Funciones "Ediwin Script"
Nombre del fichero:	ES EBIMAP Anexo Funciones EdiwinScript 1.1.odt
Versión:	1.1
Estado:	VIGENTE
Fecha:	02/04/2012
Autor:	Francisco Belda

Revisión, Aprobación		
Revisado por:		Fecha:
Aprobado por:		Fecha:

Historial de cambios			
Versión	Fecha	Descripción de la acción	Páginas
1.0	21/11/2008	Versión inicial	Todas
1.1	02/04/2012	Correcciones ejemplos ABRIR LOOKUP, FINLOOKUP, GETCAMPOSQL y SUGIENTELOOKUP.	15 – 18
2.0	17/02/15	Noticia sobre contenido obsoleto. Ahora esta información debe consultarse en la propia aplicación, El contenido de este documento puede estar obsoleto.	todas

Índice de contenido

IMPORTANTE.

Las funciones EDIWINScript que se reflejan en este documento puede estar obsoletas, incompleta o ser funciones que no estén habilitadas en el EBIMAP que el usuario está usando en ese momento.

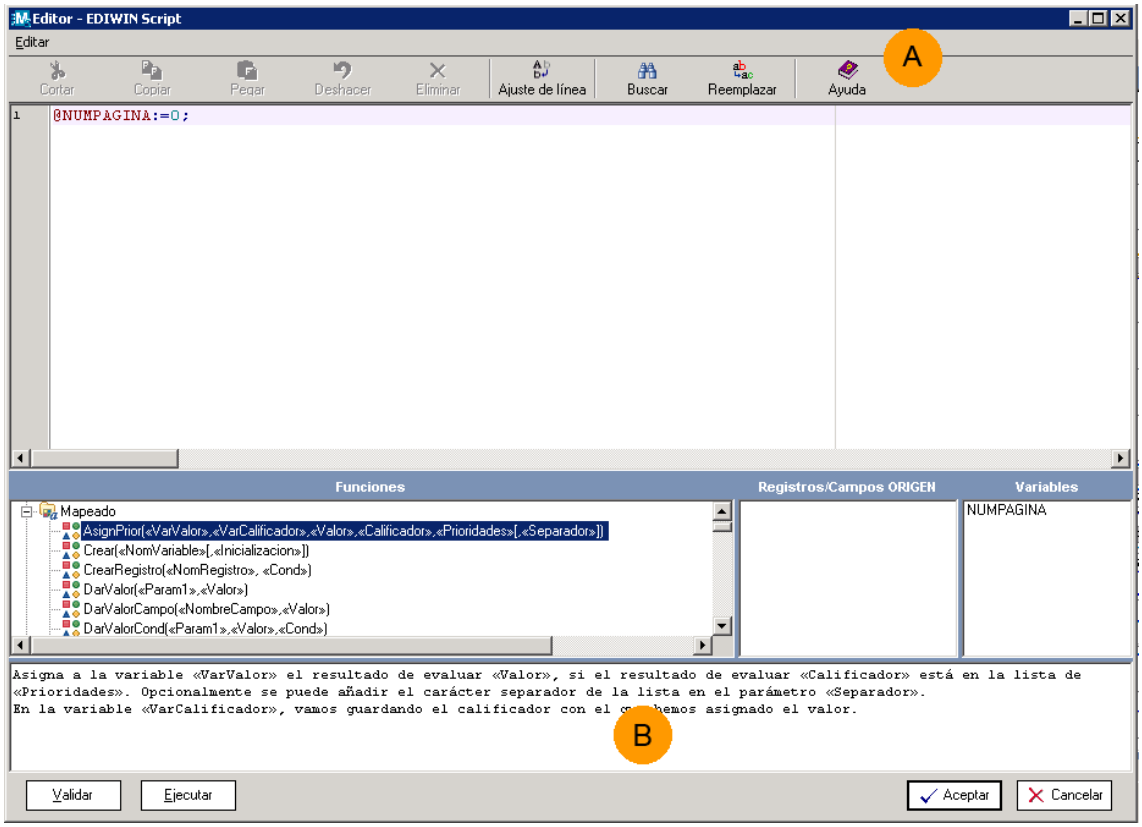
Para evitar problemas similares se aconseja utilizar la ayuda online y contextual que ofrece el propio EBIMap a través de la ventana de edición de scripts.

Utilizando estas opciones tendremos el contenido actualizado en todo momento con la versión del EBIMAP que se está utilizando.

La manera de acceder es, desde cualquier ventana del editor de EDIWIN Script.

- A) Menú principal, botón **Ayuda.** Nos lleva a una ayuda en formato HTML. Donde se han añadido ejemplos a la mayoría de funciones.

- B) Al seleccionar una función concreta. Esta misma ayuda aparece de forma contextual.



1 ANEXO. FUNCIONES “EDIWINSCRIPT”

Ediwin Script es un lenguaje Script que permite la realización de opciones avanzadas de mapeo como el análisis de datos y la asignación de valores a través de expresiones . En este anexo se muestran todas las funciones “Ediwin Script” organizadas según las funciones que realizan.

1.1 FUNCIONES MAPEADO

1.1.1 AsignPrior(«VarValor», «VarCalificador», «Valor», «Calificador», «Prioridades»[,«Separador»])

Asigna a la variable «VarValor» el resultado de evaluar «Valor», si el resultado de evaluar «Calificador» está en la lista de «Prioridades». Opcionalmente se puede añadir el carácter separador de la lista en el parámetro «Separador».

En la variable «VarCalificador», vamos guardando el calificador con el que hemos asignado el valor. “%s”

Ejemplo:

```
AsignPrior(@ORIG,@CALO,[UNH/G01/NAD. 3039],[UNH/G01/NAD. 3035],"DP PE
PR"," ")
```

1.1.2 Crear(«NomVariable»[,«Inicializacion»])

Crea la variable de nombre «NomVariable». Si la variable ya existe, se inicializa. Devuelve la cadena vacía.

Ejemplo:

```
Crear(@TOTAL, 80) {Crea la variable TOTAL con el valor 80}
Crear(@TOTAL) {Crea la variable TOTAL vacía}
```

1.1.3CrearRegistro(«NomRegistro», «Cond»)

Forzamos la creación de registro del interfaz destino «NomRegistro», si se cumple la condición «Cond»

Ejemplo:

```
CrearRegistro([UNH/G45MOA#2. 5004],MAYOR(@TOTDTO,0))
```

1.1.4DarValor(«Param1», «Valor»)

Asigna a la variable «Param1» el resultado de evaluar «Param2». Devuelve la cadena vacía.

Ejemplo:

```
DarValor(@TENGOMOA,0)
```

1.1.5DarValorCampo(«NombreCampo», «Valor», «Sobreescribir»)

Asigna al campo referenciado en «NombreCampo» el resultado de evaluar «Valor». Devuelve la cadena vacía.

Ejemplo:

```
@Campo:="[Cabecera. Referencia]"; DarValorCampo(@Campo,"123")
```

1.1.6DarValorCond(«Param1», «Valor», «Cond»)

Asigna a la variable «Param1» el resultado de evaluar «Param2» si se cumple «Param3».

Ejemplo:

```
DarValor(@TENGOMOA,0,1)
```

1.1.7FinalizarRegistro(«NomRegistro», «Cond»)

Forzamos la finalización de un registro del interfaz destino «NomRegistro», si se cumple la condición «Cond»

Ejemplo:

```
FinalizarRegistro([UNH/G45MOA#2. 5004],MAYOR(@TOTDTO,0))
```

1.1.8 GeneraInforme(«Informe», «Origen», «Destino», «TipoInforme», «Condición»[,«ContextScript»])

Si se cumple la condición, ejecuta generará el informe indicado en el parámetro «Informe».

«Informe». Nombre del informe a ejecutar. Si el nombre no tiene ruta absoluta, se asumirá que estará o en el carpeta

- `DirDatos()+'DOMINIOS\<Dominio_actual>\REPOSITORIO\INFORMES'`, o bien en
- `DirDatos()+'REPOSITORIO\INFORMES'`. Siempre se buscará primero en la carpeta `DOMINIOS`.

«Origen». Origen de datos del informe.

«Destino(opcional)». Destino de los datos del informe.

«TipoInforme». Tipo de informe a generar, a elegir entre. HTML4, EMF, EMF, PDF, JPEG, PREVIEW(Solo depuración)

«Condición». Condición de ejecución.

«ContextScript». Podemos pasar adicionalmente un script con parámetros de inicialización del contexto de ejecución.

1.1.9 HayDatos(«NomRegistro»)

Indica si el registro «NomRegistro», tiene datos o está vacío.

Ejemplo:

```
HayDatos([UNH/BGM. NODO])
```

1.1.10 HayMapError()

Indica si existe algún error durante la ejecución del mapeado actual. Si estamos en un script que no se encuentra inmerso en un mapeado devuelve 0.

1.1.11 Inc(«Variable»[,«Incremento»])

Incrementa una variable con el valor de «Incremento». Si se omite el «Incremento» se incrementa en 1.

Ejemplo:

```
• Inc(@TOTAL, 20) {Incrementa la variable TOTAL en 20}
• Inc(@TOTAL) {Incrementa en una unidad la variable TOTAL}
```

1.1.12 MAPEAR(«MAPA», «ORIGEN», «DESTINO», «CONDICIÓN» [,«FILTRO»] [,«PROPAGAERRORES»] [,«EBIMAPFORMAT»] [,«CONTEXTSCRIPT»])

Si se cumple la condición, ejecuta el mapa que se le pasa como parámetro.

«Mapa». Nombre del mapa a ejecutar. Si el nombre no tiene ruta absoluta, se asumirá que estará o en el carpeta

- DirDatos()+'DOMINIOS\dominio_actual\USUARIO', o bien en
- DirDatos()+'REPOSITORIO\USUARIO'.

Siempre se buscará primero en la carpeta DOMINIOS. Si el nombre del mapeado tiene una ruta relativa, esta se concatenará a la carpeta usuario del dominio o la carpeta usuario del repositorio.

«Origen». origen de datos del mapa

«Destino». destino de los datos del mapa

«Condición». condición de ejecución

«Filtro». Parámetro opcional. Este filtro se le aplicará al interfaz origen del mapeado. EL formato de este parámetro será.

```
REGISTRO1[condicionFiltro1],REGISTRO2[condicionFiltro2],etc.
```

Donde "REGISTROi", será el nombre del registro del interfaz que queramos filtrar y "condicionFiltroi" será la condición que queramos aplicar.

«PropagaErrores». Si vale 1, hace que los errores que se generan en el mapa que invocamos se propaguen al mapa actual.

«ebiMapFormat». Si queremos que el resultado del pmapeado se exprese en un fichero con el formato propietario XML ebiMapFormat.

«ContextScript». Podemos pasar adicionalmente un script con parámetros de inicialización del contexto de ejecución.

1.1.13 MAPEAR(«MAPA», «CONDICIÓN»[,«FILTRO»])

Si se cumple la condición, ejecuta el mapa que se le pasa como parámetro. El directorio de entrada y salida de datos, será el directorio de entrada del mapeado.

«Mapa». Nombre del mapa a ejecutar. Si el nombre no tiene ruta absoluta, se asumirá que estará o en el carpeta

- DirDatos()+'DOMINIOS\dominio_actual\USUARIO', o bien en
- DirDatos()+'REPOSITORIO\USUARIO'.

Siempre se buscará primero en la carpeta DOMINIOS. Si el nombre del mapeado tiene una ruta relativa, esta se concatenará a la carpeta usuario del dominio o la carpeta usuario del repositorio.

«Condición». Condición de ejecución.

«Filtro». Parámetro opcional. Este filtro se le aplicará al interfaz origen del mapeado.

EL formato de este parámetro será.

```
REGISTRO1[condicionFiltro1],REGISTRO2[condicionFiltro2]
```

Donde "REGISTROi", será el nombre del registro del interfaz que queramos filtrar y "condicionFiltroi" será la condición que queramos aplicar.

1.1.14 ObtenErroresMap(«TipoError», «DocumentoActual», «Warning»)

Indica los errores generados hasta el momento en el mapeado. Si indicamos alguna cadena en el parámetro «TipoError», se mostrarán únicamente los errores que contengan dicha cadena.

Si «Warning» vale 1 se devolverán también los warnings asociados al mapeado.

Por otra parte si el parámetro «DocumentoActual» vale 1, nos devolverá únicamente los errores asociados al documento que se está mapeando, en caso contrario se generarán todos los errores.

1.1.15 ObtenEtiquetaRegistro([«XPath»])

Obtiene la etiqueta fisica del registro en el origen de datos.

Se puede usar en los registros de tipo `$$ANY$$` para obtener la etiqueta fisica y poder mapear estos campos según la etiqueta

1.1.16 ObtenValorCampo(«NombreCampo»)

Devuelve el valor del campo referenciado en «NombreCampo».

Ejemplo:

```
@Campo:="["+@Registro+". Referencia]"; ObtenValorCampo(@Campo)
```

1.1.17 PreparaContexto(«Variables»)

Prepara el contexto para una llamada a Mapear o Validar.

Variables. Es una lista de las variables separada por ";", en la que se especifican las variables del origen que se pasan al destino o variables del origen que se convierten en otra en el destino

Ejemplo:

```
@Contexto := PreparaContexto("VarDestino=VarOrigen;Proveedor");
```

1.1.18 PropiedadInterfaz(«Registro/Campo»[,«Propiedad»])

Devuelve el valor de la «Propiedad» del «Registro/Campo» especificado.

Si el parámetro «Propiedad» no se especifica, devuelve una lista con todas las propiedades del registro/campo con el carácter ';' como separador.

Ejemplo:

```
PropiedadInterfaz('[CABECERA. NUMPED]','Tipo de datos');
PropiedadInterfaz('[CABECERA]','Maestro-Detalles')
```

1.1.19 RegistroActual([«XPath»])

Devuelve el nombre del registro actual. Si estamos en un registros virtuales nos devolverá el registro que ha forzado la aparición del mismo. Si el pasamos el parámetro opcional «XPath» con valor 1 devolveremos el XPath completo del registro actual.

Ejemplo:

```
RegistroActual(); {Si estamos en [xbrl:item] devolverá
[xbrli:xbrl/ifrs-
gp:AccrualBasedCapitalAdditionsForSecondaryBusinessSegment]
```

1.1.20 RegistroDisparador(«GrupoVirtual»,[«XPath»])

Requiere como parámetro el nombre/xpath de un "Grupo virtual". Devuelve el nombre del registro que ha forzado su disparo. Si pasamos el parámetro opcional «XPath» con valor 1 devolveremos el XPath completo del registro actual.

Ejemplo:

```
RegistroDisparador([xbrl:item],1); { Devolverá [xbrli:xbrl/ifrs-
gp:AccrualBasedCapitalAdditionsForSecondaryBusinessSegment]
```

1.1.21 SOBREESCRIBIR(«PARAM1», «VALOR», «COND»)

Asigna a «Param1» el resultado de evaluar «Param2», si y sólo si «Param3» vale 1. Devuelve «Param3».

«Param1» debe ser un campo del fichero. Se usa para los interfaces de entrada. Esta función tiene la característica de impedir que se cree un nuevo registro en el registro si ya existe un valor.

1.1.22 SYSCOUNT(«REGISTRO/CAMPO»)

Devuelve el número de registros leidos/escritos del tipo

Ejemplo:

```
@NumMOA := SysCount([UNH/G45MOA])
```

1.1.23 SYSCOUNTABS(«REGISTRO/CAMPO»)

Devuelve el número de registros absolutos leidos/escritos del tipo

Ejemplo:

```
@NumMOA:= SysCountAbs([UNH/G45MOA])
```

1.1.24 SYSNREGS()

Devuelve el número de registros que hay actualmente en el documento.

Ejemplo:

```
@Total:= SysNREGS();
```

1.1.25 SYSREGS(«ASIGNAR»)

Número de registros en el documento. Cada vez que se invoca esta función nos devuelve el número de registros generados, además de inicializar a 0 el contador de registros.

El parámetro opcional «NumReg» nos permite especificar un valor de inicialización del contador de registros.

Ejemplo:

```
@Contador:= SysREGS();
```

1.1.26 VALIDAR(«INTERFAZ», «FICHERO», «CONDICIÓN»[,«CONTEXTSCRIPT»])

Valida un origen de datos dado en «Fichero» mediante el interfaz «Interfaz» si se cumple «Condición».

Podemos pasar adicionalmente en «ContextScript» un script con parámetros de inicialización del contexto de ejecución.

Ejemplo:

```
Validar('EDI_INVOIC_D_93A_UN_EAN007','Factura. EDI',1)
```

1.2 BASE DE DATOS

1.2.1ABRIRLOOKUP(«NOMLOOKUP», «COND»)

Ejecuta el lookup «NomLookup» si se cumple «Cond»

Ejemplo:

```
AbrirLookup("REAL",1);
while No(FinLookup("REAL")) do
@INTER:=GetCampoSQL("REAL","CODINTERLOCUTOR","","X",1);
DarValorCond(@PEDREAL,1,[CABECERA. RECEPTOR]=@INTER);
SiguienteLookup("REAL");
endwhile;
```

1.2.2CONECTABD(«ALIAS», «USUARIO», «PASSWORD», «CONFIG»)

Conecta con una base de datos

«Alias» nombre del alias que vamos a utilizar

«Usuario» usuario de la base de datos

«Password» password del usuario

«Config» cadena de conexión

Ejemplo:

```
ConectaBD("BBDD","User","Password","ACCESS;C:\DB\DB. MDB; C:\DB\DB.
MDW")
```

1.2.3DesconectaBD(«Alias»)

Desconecta una base de datos

«Alias» nombre del alias que vamos a utilizar

Ejemplo:

```
DesconectaBD("BBDD")
```

1.2.4DesconectaBD(«Alias»)

Desconecta una base de datos

«Alias» nombre del alias que vamos a utilizar

Ejemplo:

```
DesconectaBD("BBDD")
```

1.2.5EjecutaPLSQL(«Alias», «Request»)

Esta función ejecuta procedimiento PLSQL, en base de datos del tipo ORACLE.

«Alias» nombre del alias que vamos a utilizar

«Request» Petición de ejecución del procedimiento almacenado, así como los parámetros de entrada del procedimiento, en formato XML.

Devuelve el resultado de la ejecución del procedimiento almacenado en formato XML.

Ejemplo:

```
EjecutaPLSQL("BBDD", @Request)
```

1.2.6EjecutaSQL(«Alias», «Sql»)

Ejecuta una sentencia SQL. Si es una SELECT, devuelve un DataSet en XML

«Alias» nombre del alias que vamos a utilizar

«Sql» Sentencia SQL a ejecutar

Ejemplo:

```
EjecutaSQL("BBDD", "SELECT * FROM ARTICULOS")
```

1.2.7 FinLookup(«NomLookup»)

Devuelve 0 si el lookup tiene filas disponibles y 1 si hemos llegado al final.

Ejemplo:

```
AbrirLookup("REAL",1);
while No(FinLookup("REAL")) do
@INTER:=GetCampoSQL("REAL","CODINTERLOCUTOR","","X",1);
DarValorCond(@PEDREAL,1,[CABECERA. RECEPTOR]=@INTER);
SiguienteLookup("REAL");
endwhile;
```

1.2.8 GetCampoSQL(«NomLookup», «Columna», «Defecto», «Tipo», «Cond»)

Obtiene el valor del campo «Columna» para la fila actual del lookup «NomLookup», si se cumple la condición «Cond».

Otros Parámetros

«Defecto». Valor por defecto, si no encontramos el campo.

«Tipo». Tipo de datos del campo. N (Numérico), X (Alfanumérico)

Ejemplo:

```
AbrirLookup("REAL",1);
while No(FinLookup("REAL")) do
@INTER:=GetCampoSQL("REAL","CODINTERLOCUTOR","","X",1);
SiguienteLookup("REAL");
endwhile;
```

1.2.9 ParamSQL(«NomLookup», «Parametro», «Valor», «Tipo», «Cond»)

Establece el valor «Valor» en el campo «Parametro» de la fila actual del lookup «NomLookup», si se cumple la condición «Cond».

Otros Parámetros

«Tipo». Tipo de datos del campo. N (Numérico), X (Alfanumérico)

Ejemplo:

```
ParamSQL("REAL","DOMINIO","EDI4","X",1);
AbrirLookup("REAL",1);
while No(FinLookup("REAL")) do
@INTER:=GetCampoSQL("REAL","CODINTERLOCUTOR","","X",1);
SiguienteLookup("REAL");
endwhile;
```

1.2.10 SIGUIENTELOOKUP(«NOMLOOKUP»)

Pasa a la siguiente fila del lookup y devuelve 1 si no hemos llegado al final.

Ejemplo:

```
AbrirLookup("REAL",1);
while No(FinLookup("REAL")) do
@INTER:=GetCampoSQL("REAL","CODINTERLOCUTOR","","X",1);
SiguienteLookup("REAL");
endwhile;
```

1.3 NUMÉRICAS

1.3.1ABS(«NUM»)

Devuelve el valor absoluto de Num.

Ejemplo:

```
Abs(-5) devuelve 5
Abs(5) devuelve 5
```

1.3.2DISTNUM(«PARAM1», «PARAM2»)

Realiza comparación numérica. Devuelve 1 si «Param1» y «Param2» son distintos y 0 en caso contrario.

1.3.3DIV(«PARAM1», «PARAM2»)

Devuelve el resultado de dividir «Param1» por «Param2»

1.3.4DIVENT(«PARAM1», «PARAM2»)

Devuelve el resultado de la división entera de «Param1» entre «Param2».

1.3.5IgualNum(«Param1», «Param2»)

Realiza comparación numérica.

Devuelve 1 si «Param1» y «Param2» son iguales y 0 en caso contrario.

Ejemplo:

```
IgualNum(1,2) Devuelve 0
```

1.3.6Inverso(«Param1»)

Devuelve el inverso binario de «Param1». El parámetro debe ser un número entero.

1.3.7MayorIgualNum(«Param1», «Param2»)

Realiza comparación numérica.

Devuelve 1 si «Param1» es mayor o igual que «Param2» y 0 en caso contrario.

1.3.8MayorNum(«Param1», «Param2»)

Realiza comparación numérica.

Devuelve 1 si «Param1» es mayor que «Param2» y 0 en caso contrario.

1.3.9MenorIgualNum(«Param1», «Param2»)

Realiza comparación numérica.

Devuelve 1 si «Param1» es menor o igual que «Param2» y 0 en caso contrario.

1.3.10 MenorNum(«Param1», «Param2»)

Realiza comparación numérica.

Devuelve 1 si «Param1» es mayor o igual que «Param2» y 0 en caso contrario.

1.3.11 ModEnt(«Param1», «Param2»)

Devuelve el resto de la división entera de «Param1» entre «Param2».

1.3.12 Mult(«Param1», «Param2»)

Devuelve el producto de los dos parámetros.

1.3.13 NumeroEnLetras(«Numero»[,«Idioma»])

Dado un numero obtenemos la representación verbal de la parte entera.

Ejemplo:

```
NumeroEnLetras(21234) {Devuelve "veintiún mil doscientos treinta y cuatro"}
NumeroEnLetras(21234, "PT") {Devuelve "vinte e um mil duzentos e trinta e quatro"}
```

1.3.14 REDONDEAR(«PARAM1»)

Devuelve el resultado de redondear «Param1».

1.3.15 RESTAR(«PARAM1», «PARAM2»)

Devuelve el resultado de restar «Param2» de «Param1».

1.3.16 RND(«PARAM1»)

Devuelve un valor aleatorio entre 0 y «Param1».

1.3.17 SIGUIENTECONTADOR(«NOMCONTADOR»[,«COND»])

Recupera el valor de un contador numérico

«NomContador» es el nombre del contador. Incrementa el contador en una unidad. Si le añadimos el parámetro «Cond» sólo se incrementará el contador si la condición es 1

1.3.18 SUMAR(«PARAM1», «PARAM2»)

Devuelve la suma de los dos parámetros.

1.3.19 TRUNCAR(«PARAM1»)

Devuelve el resultado de truncar «Param1».

Ejemplo:

```
Truncar(7. 123) {Devuelve 7}
```

1.3.20 VALORCONTADOR(«NOMCONTADOR»)

Recupera el valor de un contador numérico, sin incrementar su valor

1.4 LISTA DE EQUIVALENCIA

1.4.1AnadeCampoLista(«Lista», «Campo», «Longitud», «EsClave»)

Añade un Campo de nombre «Campo» y longitud «Longitud» a una lista de equivalencias de nombre «Lista».

El parámetro «EsClave» indica que el campo será clave en la Lista de Equivalencias si tiene valor 1. Se utiliza junto con la función “CrearLista()”

Ejemplo:

```
CrearLista("LISTA1",1);
AnadeCampoLista("LISTA1","NOMBRE",35,0)
```

1.4.2AnadeFilaLista(«Lista», «Valor1», «Valor2»,. . .,«ValorN»)

Añade los valores indicados en una fila de la lista de equivalencias indicada.

Podemos tener tantos parámetros «ValorI» como campos tengamos en la lista de equivalencias.

Ejemplo:

```
AnadeFilaLista("LISTA1","PEDRO", "ALTO");
{Añade una fila a la LISTA1 y en el primer campo introduce el valor
"PEDRO" y en el segundo "ALTO"
```

1.4.3AnadeFilaListaCond(«Condicion», «Lista», «Valor2»,. . .,«ValorN»)

Añade los valores indicados en una fila de la lista de equivalencias indicada, si se cumple la condición.

Podemos tener tantos parámetros «ValorI» como campos tengamos en la lista de equivalencias.

El parámetro «Valores» puede ser una lista de valores. Ver ejemplo AnadeFilaLista()

Ejemplo:

```
AnadeFilaLista(@EsAlto, "LISTA1","PEDRO", "ALTO");
{Si @EsAlto es igual a "1", añade una fila a la LISTA1 y en el primer
campo introduce el valor "PEDRO" y en el segundo "ALTO"}
```

1.4.4CrearLista(«Lista», «Temporal»)

Crea dinámicamente una Lista de Equivalencias vacía y sin campos, de nombre el que se le indica como primer parámetro. Se admiten rutas en el nombre de la lista. Si no se incluye ruta, la lista se creará en el directorio del mapeado. El parámetro dos, «Temporal», indica si la Lista existirá sólo durante la ejecución del mapeado, valor 1, o permanecerá, valor 0.

Ejemplo:

```
CrearLista("LISTA1",1); AnadeCampoLista("LISTA1","NOMBRE",35,0)
```

1.4.5CrearListaIN4(«Lista», «InterfazIN4», «NombreRegistro», «Temporal»[,«Directorio»])

Crea dinámicamente una Lista de Equivalencias a partir de un registro de un interfaz, cargando los datos que éste contenga en la lista. Se admiten rutas en el nombre de la lista. Si no se incluye ruta, la lista se creará en el directorio del mapeado. El parámetro «Temporal», indica si la Lista existirá sólo durante la ejecución del mapeado, valor 1, o permanecerá, valor 0.

Ejemplo:

```
CrearListaIN4("LISTA1","EDI_TEXTOL_90_4_AE_AECOM_1. in4","BGM",0)
```

1.4.6CrearListaLookUp(«Lista», «LookUp», «Temporal»)

Crea dinámicamente una Lista de Equivalencias a partir de un lookup. Cargando los datos que éste contenga en la lista. Se admiten rutas en el nombre de la lista. Si no se incluye ruta, la lista se creará en el directorio del mapeado. El parámetro «Temporal», indica si la Lista existirá sólo durante la ejecución del mapeado, valor 1, o permanecerá, valor 0.

1.4.7CrearListaV3(«FicheroEquivalenciaV3»)

Crea dinámicamente una Lista Temporal de Equivalencias de mismo nombre que el nombre de fichero sin extensión que se le indica como parámetro. Debe indicarse junto al nombre del fichero el path completo del mismo.

1.4.8EliminaFilaLista(«Lista»[,«Cond»])

Elimina la fila actual de una lista de equivalencias, opcionalmente podemos especificar una condición «Cond».

Requiere la ejecución previa de EquivPrepara y EquivEjecuta para restringir el ambito del borrado

Ejemplo:

```
EquivPrepara("articulos");
{Nos posicionamos al inicio de la lista}
EquivEjecuta("articulos");
{Ejecutamos la consulta sobre la lista de artículos}
{Recorremos los resultados de la consulta}
while No(EquivFinal("articulos")) do
@idCab :=EquivObtenCampo("articulos","idref");
EliminaFilaLista("articulos", (@idCab > 840000));
EquivSiguiente("articulos");
endwhile;
{Elimina de la lista "artículos" todas las filas cuyo campo "idref" se
mayor que 840000}
```

1.4.9 EliminaFilasLista(«Lista»[,«Cond»])

Ejecuta la consulta de «Lista», tomando como valores y campos clave, los parámetros pasados con la función EquivPrepara.

Requiere la ejecucion previa de EquivPrepara y EquivEjecuta para restringir el ambito del borrado

Ejemplo:

```
EquivPrepara("articulos", "8400034", "idref", 1); {Cond.  1. El campo
"idref" = "8400034"}
EquivPrepara("articulos", "2007", "referencia"); {Cond.  2. El campo
"referencia" = "2007"}
EquivEjecuta("articulos"); {Ejecutamos la consulta sobre la lista de
articulos
EliminaFilasLista("articulos"); {Eliminamos los artículos que cumplen
con la consulta}
Elimina de la lista "articulos" todas las filas cuyo campo idref=
"8400034" y referencia = "2007"}
```

1.4.10 Equiv(«Valores», «Lista», «CamposClave», «CampoResultado»)

Convierte el valor «Valores» mediante la lista de equivalencia «Lista».

Utilizaremos como campo de búsqueda «CampoClave», y devolveremos el valor de «CampoResultado».

Tanto el parámetro «Valores» como «CamposClave», pueden ser una lista de valores separados por el caracter #.

1.4.11 EQUIV(«VALOR», «LISTA», «TIPOCONVERSION»)

Convierte en valor «Valor» mediante la lista de equivalencia «Lista». El «TipoConversion» indica si la conversión es normal(0) o inversa(1).

Ejemplo:

```
{Si tenemos una lista LISTA1 con los valores 1111|2222}
Equiv("1111",LISTA1,0) {Devuelve 2222}
Equiv("2222",LISTA1,1) {Devuelve 1111}
```

1.4.12 EQUIVCOPIA(«LISTAORIGEN», «LISTADESTINO»)

Copia la lista de equivalencias «ListaOrigen» en «ListaDestino»

Ejemplo:

```
EquivCopia("Lista1","Lista2")
```

1.4.13 EQUIVEJECUTA(«LISTA»)

Ejecuta la consulta de «Lista», tomando como valores y campos clave, los parámetros pasados con la función EquivPrepara.

Ejemplo:

```
EquivPrepara("articulos", "8400034", "idref", 1); {Cond.  1. El campo
"idref" = "8400034"}
EquivPrepara("articulos", "2007", "referencia"); {Cond.  2. El campo
"referencia" = "2007"}
EquivEjecuta("articulos"); {Ejecutamos la consulta sobre la lista de
articulos}
{Recorremos los resultados de la consulta}
while No(EquivFinal("articulos")) do
@idCab:= EquivObtenCampo("articulos","descripcion");
EquivSiguiente("articulos");
endwhile;
```

1.4.14 EQUIVEJECUTA(«LISTA», «CAMPORESULTADO»)

Ejecuta la consulta de «Lista», tomando como valores y campos clave, los parámetros pasados con la función EquivPrepara() y devuelve el valor del «CampoResultado» de la lista de equivalencia «Lista».

Ejemplo:

```
EquivPrepara("articulos", "8400034", "idref", 1); {Cond.  1. El campo
"idref" = "8400034"}

EquivPrepara("articulos", "2007", "referencia"); {Cond.  2. El campo
"referencia" = "2007"}

@descripcion := EquivEjecuta("articulos", "descripcion"); {Ejecutamos
la consulta sobre la lista de artículos}
```

1.4.15 EquivFinal(«Lista»)

Indica si hemos llegado al final de la búsqueda en la lista «Lista». Ver ejemplo en función EquivEjecuta

1.4.16 EquivObtenCampo(«Lista», «CampoResultado»)

Obtenemos el valor del campo «CampoResultado» de la lista «Lista». Previamente se debe ejecutar **EquivEjecuta**(«Lista»).

Ver ejemplo en función **EquivEjecuta**()

1.4.17 EquivOrdena(«Lista», «CamposOrdenación», [«Duplicados»])

Ordena una lista de equivalencia por los campos «CamposOrdenación» (Nombre de campos de la ordenación separados por punto y coma). Tras llamar a esta función la lista de equivalencia quedará ordenada.

De esta manera podemos recorrer la lista con **EquivSiguiente()** y **EquivFinal()**.

El parámetro «Duplicados» indica si se permiten valores duplicados de los campos de ordenación en la lista ordenada. El valor por defecto es 1.

Ver ejemplo en función **EquivEjecuta**()

1.4.18 EquivOrdenaF(«Lista», «CamposOrdenación», [«Duplicados»])

Ordena una lista de equivalencia por los campos «CamposOrdenación» (Nombre de campos de la ordenación separados por punto y coma). Tras llamar a esta función la lista de equivalencia quedará ordenada. Si la lista está filtrada, se mantiene

De esta manera podemos recorrer la lista con EquivSiguiente y EquivFinal.

El parámetro «Duplicados» indica si se permiten valores duplicados de los campos de ordenación en la lista ordenada. El valor por defecto es 1.

Ver ejemplo en función **EquivEjecuta()**

1.4.19 EquivPrepara(«Lista»[,«CaseSensitive»])

Prepara una consulta sobre la lista «Lista», posicionandose en el primer elemento de la lista.

El parámetro «Lista» indica sobre que lista queremos trabajar.

Ejemplo:

```
EquivPrepara("articulos");{Nos posicionamos al inicio de la lista}
EquivEjecuta("articulos");{Ejecutamos la consulta sobre la lista de
articulos}
{Recorremos los resultados de la consulta}
while No(EquivFinal("articulos")) do
@idCab:=EquivObtenCampo("articulos","idref");
@ref:=EquivObtenCampo("articulos","referencia");
EquivSiguiente("articulos");
endwhile;
```

1.4.20 EquivPrepara(«Lista», «Valor», «CampoClave»[,«Primero»][,«CaseSensitive»])

Prepara una consulta sobre la lista «Lista», posicionandose en el primer elemento de la lista.

Además podemos establecer condiciones para la consulta usando los parámetros «CampoClave» y «Valor»

«Primero» es condicional y limpia las condiciones de la consulta.

Ejemplo:

```
EquivPrepara("articulos", "8400034", "idref", 1); { Cond.  1. El campo
"idref" = "8400034"}
EquivPrepara("articulos", "2007", "referencia"); { Cond.  2. El campo
"referencia" = "2007"}
EquivEjecuta("articulos");{Ejecutamos la consulta sobre la lista de
articulos}
{Recorremos los resultados de la consulta}
while No(EquivFinal("articulos")) do
@idCab :=EquivObtenCampo("articulos","descripcion");
EquivSiguiente("articulos");
endwhile;
```

1.4.21 EquivSiguiente(«Lista»)

Pasa al siguiente registro de la «Lista». Devuelve 0 si hemos llegado al final y 1 en caso contrario. Ver ejemplo en función **EquivEjecuta**()

1.4.22 EquivVacia(«Lista»)

Vacía la «Lista» de elementos.

Ejemplo:

```
EquivVacia("artículos")
```

1.4.23 ExisteLista(«Param1»)

Devuelve 1 si existe la Lista de Equivalencias que se pasa como parámetro, y 0 en caso contrario.

1.4.24 ModificaCampoLista(«Lista», «Campo», «Valor»[,«Cond»])

Actualiza el campo indicado de la lista. Debemos llamar anteriormente a la función **EquivEjecuta**()

1.4.25 RefrescaLista(«Lista»[,«Cond»])

Refresca una lista de equivalencia. El efecto de refrescar una lista es que la cierra físicamente, y vuelve a cargarla del disco, por si ha sido alterada por otra aplicación o proceso.

Se admite un segundo parámetro «Cond»; si su valor es 0 no se realiza el refresco de la lista si por contra «Cond» vale 1 se realiza el refresco.

1.5 TEXTO

1.5.1Asc(«Param1»)

Devuelve el código ASCII del carácter «Param1».

Ejemplo:

```
Asc("A")
{devuelve el código ASCII que es 65}
```

1.5.2CambiaExtensionFichero(«Fichero», «NuevaExtensión»)

Cambia la extensión del fichero por la nueva extensión.

Ejemplo:

```
CambiaExtensionFichero("C:\Temp\EOrders123. EDI",". BAK")
{devuelve "C:\Temp\EOrders123. BAK"}
```

Nota. No modifica el nombre del fichero en disco, para modificar el nombre del fichero en disco usar la función RenombrarF.

1.5.3Capitalize(«Param1»)

Devuelve la cadena «Param1» tras capitalizarla. Convierte la primera letra de una cadena a mayúsculas y el resto a minúsculas

Ejemplo:

```
Capitalize("hola MUNDO")
{devuelve "Hola mundo"
```

1.5.4Chr(«Param1»)

Devuelve el carácter representado por el código ASCII «Param1».

Ejemplo:

```
Chr("65")
{devuelve el carácter "A"}
```

1.5.5Concat(«Cad1»,. . .,«CadN»)

Devuelve el resultado de concatenar «Cad1». . . «CadN»

Ejemplo:

```
Concat("A","B","C","D")
{devuelve "ABCD"
```

1.5.6DerCad(«Param1», «Param2»)

Devuelve los últimos «Param2» caracteres de «Param1»

Ejemplo:

```
DerCad("Hola",2)
{devuelve "la"}
```

1.5.7Dist(«Param1», «Param2»)

Realiza comparación entre cadena. Devuelve 1 si «Param1» y «Param2» son distintos y 0 en caso contrario.

Ejemplo:

```
Dist("A","B")
{Devuelve 1}
```

1.5.8 EnCad(«Param1», «Param2»)

Comprueba la existencia de la cadena «Param1» en la cadena «Param2».

Ejemplo:

```
EnCad("AAA","BBB")
{Devuelve 0}
```

1.5.9 Entrecomilla(«Param1»)

Pone entre comillas simples «Param1». Si el IEAD es Edifact se incluirán los caracteres escape correspondientes

Ejemplo:

```
Entrecomilla("A")
{Devuelve `A´}
```

1.5.10 Entrecomillad(«Param1»)

Pone entre comillas dobles «Param1».

Ejemplo:

```
Entrecomillad("A")
{Devuelve "A"}
```

1.5.11 EstaEn(«Elemento», «Conjunto»)

Devuelve la existencia de «Param1» en la lista «Param2».

Ejemplo:

```
EstaEn([BGM. 1001],"/380/381/383/")
```

1.5.12 ExtraeExtensionFichero(«Fichero»)

Devuelve la extensión del fichero sin ruta incluido el punto.

Ejemplo:

```
ExtraeExtensionFichero("C:\Temp\EOrders123. EDI")
{devuelve ". EDI"}
```

1.5.13 ExtraeNombreFichero(«Fichero»)

Devuelve el nombre del fichero sin ruta.

Ejemplo:

```
ExtraeNombreFichero("C:\Temp\EOrders123. EDI")
{devuelve "EOrders123. EDI"}
```

1.5.14 ExtraeRutaFichero(«Fichero»)

Devuelve únicamente la ruta del fichero.

Ejemplo:

```
ExtraeRutaFichero("C:\Temp\EOrders123. EDI")
{devuelve "C:\Temp\"}
```

1.5.15 FormatFloat(«Formato», «Valor»)

FormatFloat formatea el valor dado en «Valor» usando la cadena de formateo proporcionada en «Formato».

0 Si en la cadena de formateo hay un 0, en la posición del dígito a ser formateado, si este tiene valor se conservará, y en caso contrario, se colocará un 0.

Si en la cadena de formateo hay una #, en la posición del dígito a ser formateado, si este tiene valor se conservará, y en caso contrario, se dejará vació.

. Punto decimal. El primer carácter '. ' en la cadena de formateo determina la posición del separador decimal.

, Separador de miles. Si la cadena de formateo, contiene 1 o más caracteres ',' el valor formateado tendrá un separador de miles, entre cada tres dígitos, a la izquierda del separador decimal

A continuación se muestra una serie de resultados para **`FormatFloat()`**.

«Valores»	**1234**	**-1234**	**0. 5**	**0**
FormatFloat(0)	1234	-1234	1	0
FormatFloat(0. 00)	1234. 00	-1234. 00	0. 50	0. 00

FormatFloat(#. ##)	1234	-1234 . 5		
FormatFloat(#,##0. 00)	1,234. 00	-1,234. 00	0. 50	0. 00

1.5.16 FORMATLOT(«PARAM1», «PARAM2», «PARAM3», «PARAM4»)

Formatea el valor «Param4» según el formato LOT nativo.

«Param1» indica el Tipo ('N','X').

«Param2» indica el tamaño.

«Param3» indica el número de decimales en el caso de numéricos.

Nota. Esta función está obsoleta. Para formatear número utilizar FormatFloat

Ejemplo:

```
Format("X",4,0,"AA")
{devuelve una cadena formada por dos A y dos espacios en blanco:”  AA”}
```

1.5.17 IGUAL(«PARAM1», «PARAM2»)

Realiza comparación entre cadena.

Devuelve 1 si «Param1» y «Param2» son iguales y 0 en caso contrario.

Ejemplo:

```
Igual("A","a")
{devuelve 0}
```

1.5.18 IZQCAD(«PARAM1», «PARAM2»)

Devuelve los primeros «Param2» caracteres de «Param1»

1.5.19 LONG(«CADENA»)

Devuelve la longitud de la cadena «Param1».

1.5.20 MAYOR(«PARAM1», «PARAM2»)

Realiza comparación entre cadena.

Devuelve 1 si «Param1» es mayor que «Param2» y 0 en caso contrario.

1.5.21 MAYORIGUAL(«PARAM1», «PARAM2»)

Realiza comparación entre cadena.

Devuelve 1 si «Param1» es mayor o igual que «Param2» y 0 en caso contrario.

1.5.22 MAYUSCULAS(«PARAM1»)

Devuelve «Param1» tras convertirlo a mayúsculas.

1.5.23 MENOR(«PARAM1», «PARAM2»)

Realiza comparación entre cadena

Devuelve 1 si «Param1» es menor que «Param2» y 0 en caso contrario.

1.5.24 MENORIGUAL(«PARAM1», «PARAM2»)

Realiza comparación entre cadena.

Devuelve 1 si «Param1» es menor o igual que «Param2» y 0 en caso contrario.

1.5.25 MINUSCULAS(«CADENA»)

Devuelve «Param1» tras convertirlo en minúsculas.

1.5.26 OBTENVALORCADENA(«CADENA», «CARACTERSEPARADOR», «INDICE»)

Cadena es una cadena de valores separada por el carácter separador «CaracterSeparador», enumerados del 0 hasta n. Esta función devuelve el elemento "indice" de la cadena. Si indice es menor que cero o mayor al número de elementos de la cadena devuelve vacío.

Ejemplo:

```
ObtenValorCadena("Pedro#Juan#Luís","#",1)
{devuelve "Juan"}
```

1.5.27 POSX(«CADENAORIGEN», «CADENAABUSCAR»)

Devuelve la posición de «CadenaABuscar» en «CadenaOrigen».

Ejemplo:

```
PosX("Número$Pedido","$") = 7
```

1.5.28 QUITARCARS(«CADENA», «CARACTER»)

Devuelve la cadena «Param1» sin los elementos de «Param2».

«Param1» es la cadena inicial.

«Param2» es la cadena de caracteres a eliminar.

También podemos eliminar un carácter poniendo en «Param2» su código ASCII precedido del carácter #. (Ej. #32)

Ejemplo:

```
QuitarCars("ababab","b")
{devuelve bbb}
QuitarCars("ababab","c")
{devuelve ababab}
```

1.5.29 QuitarCeros(«Cadena», «Param2», «Param3»)

Devuelve la cadena «Cadena» eliminando los ceros que contenga.

«Param2» indica si se han de quitar los ceros a la izquierda ("I"), derecha ("D") o ambas ("A")

«Param3» indica si el quitar ceros es Numérico ("N") o Alfanumérico ("X")

1.5.30 Recortar(«Param1»)

Devuelve la cadena «Param1» eliminando primero los espacios blancos al principio y final de la cadena.

1.5.31 RecortarDer(«Param1»)

Devuelve la cadena «Param1» eliminando primero los espacios blancos al final de la cadena.

1.5.32 RecortarIzq(«Param1»)

Devuelve la cadena «Param1» eliminando primero los espacios blancos al principio de la cadena.

1.5.33 Str(«expresión»[,«RellenaConCeros»])

Rodea de comillas simples «expresión»

1.5.34 StrD(«expresión»)

«Rodea de comillas dobles «expresión»

1.5.35 SubCad(«Cadena», «Inicio», «Longitud»)

Devuelve la subcadena contenida en «Cadena» que empieza en la posición «Inicio» y tiene de longitud «Longitud».

1.5.36 SustituyeCad(«CadenaOrigen», «PatrónAntiguo», «PatrónNuevo»)

Devuelve «CadenaOrigen», sustituyendo en ella toda ocurrencia de «PatrónAntiguo», por «PatrónNuevo».

1.5.37 Traduce(«cadena»[,«Idioma»])

Traduce la cadena «cadena» por defecto al idioma en el que se ha lanzado la ejecución. O en el idioma indicado en el parametro «Idioma».

Ejemplo:

```
Traduce("Factura", "EN")
{Devuelve "Invoice"}
```

1.5.38 Val(«Cadena»)

Convierte «Cadena» de string a número. Si «Cadena» no es numerico se devuelve cero.

1.6 ENTRADA/SALIDA

1.6.1 BorraDir(«Directorio» [,«Cond»])

Borra un directorio y su contenido, tanto ficheros como subdirectorios.

Ademas se puede añadir una condición para el borrado con el parametro opcional «Cond»

Ejemplo:

```
BorraDir("C:\EDIWIN4\DOMINOS\TMP",1)
```

1.6.2 BorrarAntiguos(«Mascara», «Dias», «Recursivo»[,«Cond»])

Borra los ficheros que cumplan la máscara «Mascara» y cuya antigüedad sea mayor al número de días «Dias».

El parámetro «Recursivo» indica si se borrarán los ficheros en todos los subdirectorios

Además podemos añadir una condición para la borrado con el parametro opcional «Cond»

Ejemplo:

```
BorrarAntiguos("C:\Temp\*. tmp",10,1,1)
```

1.6.3BorrarF(«Mascara», «Cond»)

Borra los ficheros que cumplan la máscara «Mascara», si se cumple la condición «Cond»

Ejemplo:

```
BorrarF("C:\Temp\*. tmp",1)
```

1.6.4CalculaHash(«Datos», «Algoritmo», «FormatoSalida»)

Calcula el HASH «Datos» del tipo que le indiquemos en «Algoritmo». El parámetro «FormatoSalida» indica si el resultado lo queremos en hexadecimal("HEX") o en MIME base 64("MIME64").

Por defecto el valor de «Algoritmo» es "Message Digest 5" y el valor de «FormatoSalida» es "HEX".

Otros algoritmos			
"Message Digest 4"	320"	1"	"Square"
"Message Digest 5"	"Haval-128"	"Sapphire II-128"	"Tiger"
"Ripe Message Digest	"Haval-160"	"Sapphire II-160"	"XOR-16"
128"	"Haval-192"	"Sapphire II-192"	"XOR-32"
"Ripe Message Digest	"Haval-224"	"Sapphire II-224"	"CRC-16 CCITT"
160"	"Haval-256"	"Sapphire II-256"	"CRC-16 Standard"
"Ripe Message Digest	"Secure Hash	"Sapphire II-288"	"CRC-32"
256"	Algorithm"	"Sapphire II-320"	
"Ripe Message Digest	"Secure Hash Algorithm	"Snefru-256"	

1.6.5ComprimeZIP(«MascaraOrigen», «FicheroZIP», «Cond»)

Esta función comprime todos los archivos que cumplan «MascaraOrigen» y los almacena en el fichero comprimido «FicheroZIP», si se cumple la condición «Cond».

Ejemplo:

```
ComprimeZIP("C:\TEMP\IN\*. TXT", "C:\TEMP\OUT. ZIP", 1)
```

1.6.6ConvierteBase64(«CadenaOrigen», «Tipo», «Cond»)

Devuelve como resultado la conversión de «CadenaOrigen» a/de Base64. El valor de «Tipo» puede ser E=Encode(De Texto a Base64) o D=Decode(De Base64 a Texto).

Ejemplo:

```
@B64:=ConvierteBase64(@Texto,"E",1)
{Nos devuelve en la variable @B64 el contenido de @Texto codificado en
base64. }
```

1.6.7 ConvierteBase64(«FicheroOrigen», «FicheroDestino», «Tipo», «Cond»)

Convierte «FicheroOrigen» a/de Base64 y lo vuelca sobre «FicheroDestino». EL valor de «Tipo» puede ser E=Encode(De Texto a Base64) y D=Decode(De Base64 a Texto).

Ejemplo:

```
ConvierteBase64("C:\a. txt","C:\b. txt","E",1)
{Convierte el fichero C:\a. txt a BASE64 y lo escribe en C:\b. txt}
```

1.6.8 ConvierteUnicodeToAnsi(«CadenaOrigen», «Codificación», «GenEntities», «Cond»)

Devuelve el resultado de convertir la cadena «CadenaOrigen» en formato UNICODE (el formato interno de los datos -registros/campos- de ebiMap) a ANSI en función de la codifición indicada por el parámetro «Codificación».

Los valores admitidos por «Codificación» son:

Valores Codificación Unicode Admitidos			
1=ISO_8859_1	5=ISO_8859_5	9=ISO_8859_9	13=UTF_16_BE
2=ISO_8859_2	6=ISO_8859_6	10=ISO_8859_10	14=UTF_16_LE
3=ISO_8859_3	7=ISO_8859_7	11=US_ASCII	
4=ISO_8859_4	8=ISO_8859_8	12=UTF_8	

El parámetro «GenEntities» si toma el valor "1" indica que cuando un carácter UNICODE(tenemos aproximadamente 64000) no se pueda representar en la codificación indicada,

lo representemos en forma de "entitie". &#CódigoUnidode; . En caso contrario «GenEntities» no vale "1", estos caracteres se sustituirán por el carácter de subrayado “_”

Ejemplo:

```
ConvierteUnicodeToAnsi([Articulo. Descripcion],7,1,1)
{Devolverá la representación ANSI de [Articulo. Descripcion] respecto a
la codificación ISO_8859_7 (que representa el alfabeto griego)}
```

1.6.9 CopiarF(«Mascara», «DirDestino», «Cond»)

Copia los ficheros que cumplan la mascara «Mascara» al directorio «DirDestino», si se cumple la condición «Cond»

Ejemplo:

```
CopiarF("I:\Entrada\*. TXT", "C:\Entrada\",1)
```

1.6.10 CreaDir(«Directorio»[,«Cond»])

Crea un directorio dado. Si no existe toda la ruta crea todos los directorios necesarios. Debe cumplirse la condición «Cond»

Ejemplo:

```
CreaDir("C:\EDIWIN4\DOMINOS\TMP",1)
```

1.6.11 DescomprimeZIP(«FicheroZIP», «DirDestino», «Cond»)

Descomprime el contenido del fichero comprimido «FicheroZIP» en el directorio «DirDestino»,si se cumple la condición «Cond».

Ejemplo:

```
DescomprimeZIP("C:\TEMP\IN. ZIP", "C:\TEMP\OUT\", 1)
```

1.6.12 DirDatos()

Devuelve el directorio de datos (EDIWIN)

1.6.13 DirIn()

Devuelve el directorio de entrada del mapeado o validación actuales. Si no se tiene interfaz origen se devuelve DirDatos().

1.6.14 DirOut()

Devuelve el directorio de salida del mapeado o validación actuales. Si no se tiene interfaz destino se devuelve DirIn().

1.6.15 EscribeFicheroFisico(«FicheroDestino», «Datos»)

Escribe sobre el fichero «FicheroDestino» el valor de «Datos».

Ejemplo:

```
EscribeFicheroFisico("C:\a. txt",@Datos)
```

1.6.16 EscribirF(«NomFich», «Datos», «Cond»)

Escribe en el fichero «NomFich» el texto «Datos» si se cumple la condición «Cond».

Ejemplo:

```
EscribirF(DirDatos()+"Logs\LogDebug. Log",Fecha("dd/mm/yyyy hh:nn")+"
Inicio mapeado",1);
```

1.6.17 ExisteFichero(«NombreFichero»)

Devuelve 1 si existe el fichero, 0 en caso contrario. Si el nombre no tiene ruta, se buscará en el directorio de salida del mapeado.

1.6.18 ExtraerAdjuntos(«FicheroOrigen»)

Extrae los ficheros adjuntos de un mensaje MIME, en el directorio con el siguiente formato. <NomFichIn>_000. IN

Ejemplo:

```
ExtraerAdjuntos("C:\Temp\entrada. eml")
```

1.6.19 FichDefinicion()

Devuelve el fichero de definición si el script se ejecuta en alguno de los siguientes ámbitos. Mapeado o Multimapeado (MA4/MM4). Validación de interfaces(IN4). Generación de informes(INF).

1.6.20 FichOrigen(«ConRuta»)

Devuelve el nombre de fichero de datos del primer registro del interfaz origen. Si parámetro «ConRuta» es un 1, Indica que queremos el nombre de fichero con la ruta incluida en caso contrario nos devuelve el nombre del fichero sin la ruta.

Este nombre de fichero irá variando en función a la máscara origen utilizada.

Por ejemplo dado un interfaz origen con un IEAD_LOT cuya mascara de fichero es *. LOT y ejecutar un mapeado sobre un directorio con F1. LOT, F2. LOT y F3. LOT.

La función FichOrigen(), en cada momento nos devolverá el fichero que se esté recorriendo en ese instante.

1.6.21 GetFileDate(«File», «Format»)

Obtiene la fecha y hora del fichero «File» con el formato de fecha «Format»

Ejemplo:

```
GetFileDate("C:\a. txt", "dd/mm/yyyy hh:nn")
```

1.6.22 GetFileSize(«File»)

Obtiene el tamaño del fichero «File»

Ejemplo:

```
GetFileSize("C:\a. txt")
```

1.6.23 HTMLDecode(«Datos»)

Convierte «Datos» codificados en formato HTML

1.6.24 HTMLEncode(«Datos»)

Convierte «Datos» a la codificación HTML

1.6.25 LeeFicheroFisico(«FicheroOrigen», «MaxNumBytes»)

Lee los «MaxNumBytes» de «FicheroOrigen». Si «MaxNumBytes» es 0 lee el fichero completo.

Ejemplo:

```
LeeFicheroFisico("C:\a. txt",1024)
```

1.6.26 ListaDir(«Directorio», «Lista», «Temporal»[,«Cond»])

Crea una lista temporal llamada «Lista» y la carga con los nombres de los ficheros del directorio «Directorio».

La lista consta de cuatro campos. NombreFichero, Longitud, Fecha y Tipo. Donde Tipo sera FiC si se trat de un fichero o DIR si es un directorio.

Debe cumplirse la condición «Cond»

Ejemplo:

```
ListaDir("C:\EDIWIN4\DOMINOS\TMP","Temporales",1)
```

1.6.27 MoverF(«Mascara», «DirDestino», «Cond»)

Mueve los ficheros que cumplan la máscara «Mascara» al directorio «DirDestino», si se cumple la condición «Cond»

Ejemplo:

```
MoverF("I:\Entrada\*. TXT", "C:\Entrada\",1)
```

1.6.28 ReadAllText(«File» [,«Encoding»])

Devuelve el contenido de un fichero como una cadena de texto.

Parámetros:

«File». String. Nombre y ruta del fichero a leer. Requerido.

«Encoding». String. Codificación usada para leer el fichero. Por defecto es "ISO-8859-1". Opcional.

1.6.29 RenombrarF(«FicheroOrigen», «FicheroDestino», «Cond»)

Renombra el fichero «FicheroOrigen» a «FicheroDestino, si se cumple la condición «Cond»

Ejemplo:

```
RenombrarF("I:\Entrada\Fich. TXT", "C:\Entrada\FichCopia. TXT",1)
```

1.6.30 URLDecode(«Datos»)

Convierte «Datos» codificados en formato URL

1.6.31 URLEncode(«Datos»)

Convierte «Datos» a la codificación URL

1.6.32 WriteAllText(«File», «Text» [, «Append», «Encoding»])

Escribe texto a un fichero (con una codificación).

Parámetros:

«File». String. Fichero en el que se escribirá el texto. Requerido.

«Text». String. Texto que se escribirá en el fichero. Requerido.

«Append». Boolean. Añadir texto a un fichero existente o sobrescribir el contenido. Por defecto es "0". Opcional.

«Encoding». String. Codificación usada para escribir el fichero. Por defecto es "ISO-8859-1". Opcional.

1.7 ACCESO A EDIWIN

1.7.1 ActualizaComunicacion(«IDMessage», «Campos»)

Actualiza los campos de un intercambio en la tabla de comunicaciones

«IDMessage». Identificador del intercambio a actualizar

«Campos». Campos a actualizar

Ejemplo:

```
ActualizaComunicacion(@IDMessage,'ORIGENPROT='juan@edicom.
es',RECUPERADO='+Entrecomilla(@Fecha)
+',DESTINOPROT='+Entrecomilla(@Destino)')
```

1.7.2 BorraComunicacion(«IDMessage»)

Borra un intercambio de la tabla de comunicaciones

«IDMessage». Identificador del intercambio a borrar

Ejemplo:

```
BorraComunicacion(@IDMessage)
```

1.7.3 BuscarDesc(«RefArticulo»)

Devuelve la descripción de la referencia interna de artículo que se pasa como parámetro de la lista de artículos de Ediwin

Ejemplo:

```
BuscarDesc("123456")
```

1.7.4 BuscarEAN(«RefArticulo»)

Devuelve el valor EAN de la referencia interna de artículo que se pasa como parámetro

1.7.5 BuscarRef(«EanArticulo»)

Dado un valor EAN, devuelve la referencia interna usada para el artículo

1.7.6 CheckDigit(«Code», «Odd», «Even»[,«Modulus»] [,«UseMod»][,«UseLuhn»][,«Inverse»])

Calcula el digito de control de un código.

Parámetros:

«Code». Código sobre el que se calculara el digito de control

«Odd». Peso de los dígitos impares (de derecha a izquierda). Requerido

«Even». Peso de los dígitos pares (de derecha a izquierda). Requerido

«Modulus». Modulo del digito de control. Opcional. Por Defecto. 10

«UseMod». Usar modulo en cálculos intermedios. Opcional. Por Defecto. 0

«UseLuhn». Usar Luhn en cálculos intermedios. Opcional. Por Defecto. 0

«Inverse». Usar modulo inverso. Opcional. Por Defecto. 0

Ejemplo:

```
{EAN-13, EAN-8, UPCEAN-14, ITF-14, SCC-14, DUN-14:
CheckDigit(@Codigo, 3, 1, 10, 0, 0, 1)
{SIREN.
CheckDigit(@Codigo, 2, 1, 10, 0, 1, 0)
```

1.7.7 ConexionFTP(«Condn», «Comando», «Servidor», «Puerto», «Usuario», «Password», «TipoServidor», «ModoPasivo», «CertificadoCliente», «DirLocal», «DirRemoto», «MascaraOrigen», «MascaraSinc», «ModoTransfer», «CambioEstado», «FicheroLog», «LogDebug», «Reintentos», «TimeOut», «ProxyServer», «ProxyPort», «ProxyUser», «ProxyPassword», «ProxyType»)

Realiza una conexión FTP si se cumple la condición «Cond».

El resultado de la función será una cadena que comenzará por OK y en caso contrario comenzará por ERROR.

El significado de los parámetros es:

- «Comando». Indica la acción que queremos realizar durante la conexión
- «Servidor» y «Puerto». Dirección IP o nombre del servidor y el puerto de conexión (por defecto. 21).

- «DirLocal» y «DirRemoto». Son los directorios de trabajo tanto en ambos extremos de la transferencia.
- «TipoServidor». No utilizado. En todo caso se autodetecta el tipo de servidor al establecer la conexión FTP.
- «FicheroLog». Si aparece se generará un log. Si «LogDebug» es 1, entonces en el log se incluirá todas las tramas enviadas y recibidas.
- «ModoPasivo». Indica que queremos entrar en modo pasivo.
- «ModoTransfer». ftASCII. 0, ftBinary. 1
- «CambioEstado». REN (Renombramos los ficheros a BAK), DEL (Borramos los ficheros transferidos)
- «Reintentos». Numero de reintentos
- «ProxyServer, ProxyPort, ProxyUser». Configuración de PROXY (Opcional)
- «ProxyType»:
  - fpcmNone. 0
  - fpcmUserSite. 1
  - fpcmSite. 2
  - fpcmOpen. 3
  - fpcmUserPass. 4
  - fpcmTransparent. 5
  - fpcmHttpProxyWithFtp. 6

Lista de Comandos:

GET

Descargar ficheros desde el servidor FTP

- Param1. Mascara que deben cumplir los ficheros a descargar
- Param2. Si toma valor significa que para cada fichero a transferir se comprobará previamente si existe un fichero de sincronización que cumpla la mascara definida

DGET

Descargar ficheros desde el servidor FTP (recursiva). Este comando se comporta como GET pero además buscara ficheros que cumplan la mascara dentro de los subdirectorios

- Param1. Mascara que deben cumplir los ficheros a descargar
- Param2. Si toma valor significa que para cada fichero a transferir se comprobará previamente si existe un fichero de sincronización que cumpla la mascara definida

PUT

Subir ficheros al servidor FTP. Si no existe el fichero lo crea y si existe lo sobrescribe

- Param1. Máscara que deben cumplir los ficheros a subir al servidor

DPUT

Subir ficheros al servidor FTP (recursiva). Este comando se comporta como PUT pero además buscara ficheros que cumplan la mascara dentro de los subdirectorios

- Param1. Máscara que deben cumplir los ficheros a subir al servidor

APPEND

Subir ficheros al servidor FTP. Si no existe el fichero lo crea y si existe añade el contenido del fichero a subir a fichero que reside en el servidor

- Param1. Máscara que deben cumplir los ficheros a subir al servidor

DAPPEND

Subir ficheros al servidor FTP (recursiva). Si no existe el fichero lo crea y si existe añade el contenido del fichero a subir a fichero que reside en el servidor.

Este comando se comporta como **APPEND** pero además buscara ficheros que cumplan la mascara dentro de los subdirectorios

- Param1. Máscara que deben cumplir los ficheros a subir al servidor

NLST

Muestra un listado con los nombres de los ficheros que cumplen la mascara en el directorio remoto

- Param1. Mascara de búsqueda
- Param2. Nombre del fichero donde se almacenará el listado en el cliente

LIST

Muestra un listado con los ficheros que cumplen la mascara en el directorio remoto (Muestra ademas información de atributos, tamaño, etc. . .)

- Param1. Máscara de búsqueda

- Param2. Nombre del fichero donde se almacenará el listado en el cliente

Ejemplo:

```
09-05-06 12:13PM 141312 IEPE_HTTP. doc
09-05-06 2:00PM <DIR> AAA
```

REN

Renombrar un directorio o un fichero en el servidor

- Param1. Nombre actual del fichero/directorio
- Param2. Nombre nuevo del fichero/directorio

1.7.8 ConexionHTTP(«Cond», «Comando», «URL», «Texto», «FicheroPetición», «FicheroRespuesta», «FicheroLog», «LogDebug», «Usuario», «Contraseña», «FicheroPFX», «ContraseñaPFX», «TipoSSL», «Reintentos», «TimeOut», «ProxyServer», «ProxyPort», «ProxyUser», «ProxyPassword»)

Realiza una conexión HTTP si se cumple la condición «Cond».

El resultado de la función será una cadena que comenzará por OK y en caso contrario comenzará por ERROR.

El significado de los parámetros es:

- «Comando». Indica la acción que queremos realizar durante la conexión.
- «URL». Con estos parámetros indicamos la dirección URL a la que queremos acceder.
- «FicheroLog». Si aparece se generará un log. Si «LogDebug» es 1, entonces en el log se incluirá todas las tramas enviadas y recibidas.
- «Reintentos». Numero de reintentos
- «ProxyServer», «ProxyPort», «ProxyUser». Configuración de PROXY (Opcional)
- «ProxyType»:
  - fpcmNone 0
  - fpcmUserSite 1
  - fpcmSite 2

- fpcmOpen 3
- fpcmUserPass 4
- fpcmTransparent 5
- fpcmHttpProxyWithFtp 6

Lista de Comandos:

GET

Solicita un recurso HTTP referenciado por la dirección URL.

- FicheroRespuesta. Fichero donde se almacenará la respuesta HTTP.

HEAD

Solicita la cabecera de un recurso HTTP referenciado por la dirección URL.

- FicheroRespuesta. Fichero donde se almacenará la respuesta HTTP.

POST

Envia datos a un recurso HTTP referenciado por la dirección URL.

- Texto. Datos de la petición HTTP.
- FicheroPeticion. Fichero con los datos de la petición HTTP
- FicheroRespuesta. Fichero donde se almacenará la respuesta HTTP.

Nota. Podemos utilizar indistintamente los parámetros Texto y FicheroPetición para establecer los datos

1.7.9 ConexionTCP(«Condn», «Servidor», «Puerto», «FichEnvio», «FichRespuesta», «FicheroLog», «LogDebug», «Reintentos», «TimeOut»)

Realiza una conexión TCP si se cumple la condición «Cond».

El resultado de la función será una cadena que comenzará por OK y en caso contrario comenzará por ERROR.

El significado de los parámetros es:

«Servidor» y «Puerto»,. Dirección IP o nombre del servidor y el puerto de conexión (por defecto. 21).

«FichEnvio». Fichero con los datos de la petición TCP

«FichRespuesta». Fichero donde se almacenará la respuesta TCP.

«FicheroLog». Si aparece se generará un log. Si «LogDebug» es 1, entonces en el log se incluirá todas las tramas enviadas y recibidas.

«Reintentos». Número de reintentos

«TimeOut». Tiempo máximo de espera para recibir la respuesta

1.7.10 CONVERSIONCODLOT(«NOMCAMPO», «TIPOCONVERSION»)

Esta función se utiliza para realizar conversión de códigos en ficheros LOT.

El parámetro «NomCampo» Indica el campo que queremos convertir.

El parámetro «TipoConversion» puede tomar los valores {PO,ART} para puntos operacionales o artículos respectivamente.

Sólo actúa en el ámbito de una importación o exportación LOT desde EDIWIN4.

Habitualmente realizaremos esta llamada desde una función llamada LOT de un Interfaz EDIFACT.

Esta función no esta pensada para utilizarse en un mapeado, sino para utilizarla en la validación de un interfaz. Para realizar conversión de códigos en mapeados, tenemos funciones como. **`ConvertirPO()`**, **`ConvertirPOSII()`**, **`BuscarEAN()`** y **`BuscarREF()`**.

1.7.11 CONVERTIRPO(«CODSII»)

En Ediwin se puede asociar a un Punto Operacional un Código de interlocutor interno. Esta función devuelve el PO asociado a un código del SII. El parámetro «CodSII», es el código del SII.

1.7.12 CONVERTIRPOSII(«CODPO»)

En Ediwin se puede asociar a un Punto Operacional un Codigo de interlocutor interno. Esta función devuelve el código del SII asociado a un PO. El parámetro «CodPO», es el código del PO de la libreta de direcciones de EDIWIN.

1.7.13 DOMINIO([«NUEVODOMINIO»])

Devuelve el dominio activo. Si pasamos como parámetro «NuevoDominio» cambiamos a este dominio.

Ejemplo:

```
Dominio()
{Devuelve el dominio activo}
```

1.7.14 EEACL(«Param1», «Param2», «Param3»)

Devuelve datos de una estructura administrativa de cliente.

«Parm1» contiene el punto operacional de quien pide, «parm2» de quien recibe y la variable «parm3» uno de los siguiente valores. "Código a quien se factura", "Código quien paga", "Código copia factura", "Código relación factura".

Ejemplo:

```
EEACL("1111111111116","2222222222222","Código quien paga")
```

1.7.15 EEAPR(«Param1», «Param2»)

Devuelve datos de una estructura administrativa de proveedor.

«Param1» contiene el punto operacional de "a quien se pide" y la variable «Param2» uno de los siguientes valores. "Código a quien se paga".

Ejemplo:

```
EEAPR("1111111111116","Código a quien se paga")
```

1.7.16 EnviaMail(«Condición», «Host», «From», «To», «Asunto», «Texto»[,«FicherosAdjuntos», «FicheroHTML», «FicheroLog», «UsuarioESMTP», «PasswordESMTP», «Reintentos», «TimeOut», «ResolucionDNS», «ProxyServer», «ProxyPort», «ProxyUser», «ProxyPassword»])

Envía un correo electrónico por SMTP si se cumple la condición «Cond».

El resultado de la función será una cadena que comenzará por OK y en caso contrario comenzará por ERROR.

El significado de los parámetros es:

«Host». Dirección IP. Si «ResolucionDNS» vale uno podemos especificar un nombre de servidor.

«From». Remitente del mensaje

«To». Destinatario del mensaje

«Asunto». Asunto del mensaje. Si en el asunto indicamos "NoEncode" el envío SMTP se hace del texto sin ninguna codificación.

«Text». Cuerpo del mensaje. Se puede pasar el contenido de un fichero pasando como texto "@file=NombreFichero".

«FicheroHTML». Fichero con el texto alternativo del mensaje

«FicherosAdjuntos». Lista de ficheros adjuntos del mensaje

«FicheroLog». Si aparece se generará un log en fichero especificado

«Reintentos». Número de reintentos

«TimeOut». Tiempo máximo de espera para recibir la respuesta

«UsuarioESMTP», «PasswordESMTP». Usuario y contraseña (Solo ESMTP)

«ProxyServer», «ProxyPort», «ProxyUser». Configuración de PROXY (Opcional).

1.7.17 ExtraeContenido(«ID», «Tipo», «Nombre de fichero»[,«Alias», «IdVolumen»])

Extrae el contenido del documento o intercambio, con identificador ID, a un fichero.

«ID». Identificador del documento o intercambio

«Tipo». Tipo del documento a extraer (DOCUMENTO,INTERCAMBIO)

«Nombre de fichero». Nombre con el que se extraerá el contenido. Si el nombre no tiene ruta, se extraerá en el directorio de salida del mapeado

«Alias». Alias de la base de datos donde esta el documento o intercambio

«IdVolumen». Identificador del volumen, en el caso que el documento o intercambio este almacenado en un volumen.

1.7.18 ExtraeDocumento(«Origen», «Destino», «Número de referencia», «Dominio», «Nombre de fichero»[,«Alias», «IdVolumen»])

Dado un origen, un destino, un número de referencia y un dominio, extrae el contenido del documento a un fichero

«Origen». Origen del documento

«Destino». Destino del documento

«Número de referencia». Número de referencia del documento

«Domino». Dominio en el cual se encuentra el documento

«Nombre de fichero». Nombre con el que se extraerá el documento. Si el nombre no tiene ruta, se extraerá en el directorio de salida del mapeado

«Alias». Alias de la base de datos donde esta el documento o intercambio

«IdVolumen». Identificador del volumen, en el caso que el documento o intercambio este almacenado en un volumen

En el caso que alguno de los parámetros (origen, destino, número de referencia o dominio) este vacío, estos se omitirán de la búsqueda.

1.7.19 ExtraerComunicacion(«IDMessage», «Fdata», «Fcontrol»)

Extrae los datos de un intercambio de la tabla de comunicaciones. Devuelve un DataSet en xml.

«IDMessage». Identificador del intercambio del que queremos extraer sus datos

«Fdata». Nombre del fichero donde queremos que nos exporte el fichero de datos

«Fcontrol». Nombre del fichero donde queremos que nos exporte el fichero de control, si lo tiene

Ejemplo:

```
@DataSet := ExtraerComunicacion(@IDMessage,@DIRTEMP+'Com_1.
edi',@DIRTEMP+'Com_1. ctr')
```

1.7.20 FuncionPO(«Param1», «Param2»)

Devuelve el PO asociado a una función de un código del SII. Se pueden ver todos los códigos de funciones en la liberta de ediwin, apartado SII

«Param1» es el código del SII.

«Param2» es la función.

Ejemplo:

```
FuncionPO("1111","PRI");
{devuelve el PO asociado a un código SII cuya función es PRI}
```

1.7.21 FuncionPOSII(«Param1», «Param2»)

Devuelve el código del SII asociado a una función de un PO.

«Param1» es el código del SII.

«Param2» es la función.

Ejemplo:

```
{Si tenemos un PO=1111111111116 y asociado un SII=1111 con la función
PRI entonces}
FuncionPOSII("1111","PRI)
{Devuelve 1111111111116}
```

1.7.22 InsertaComunicacion(«Dominio», «Iepe», «Buzon», «IdExterno», «Tipo», «Situacion», «OrigenProt», «DestinoProt», «Destino», «Fdata», «Fcontrol»)

Inserta un intercambio en la tabla de comunicaciones. Devuelve el identificador del intercambio.

«Dominio». Dominio del intercambio, si no se le pasa utilizará el del acceso a ediwin

«Iepe». Nombre de la Iepe

«Buzon». Nombre del buzón de comunicaciones

«IdExterno». Identificador externo si procede

«Tipo». Tipo del intercambio (E:Entrada, S:Salida, A:Autack)

«Situacion». Situación del intercambio (DEP,ENV,RCP,RH1,RH2,RH3,REC,TRA)

«OrigenProt». Origen del protocolo del intercambio

«DestinoProt». Destino del protocolo del intercambio

«Destino». Destino del documento

«Fdata». Nombre del fichero que contiene el intercambio a insertar

«Fcontrol». Nombre del fichero de control del intercambio

Ejemplo:

```
@IDMessage :=
InsertaComunicacion(@DOMINIO,@NOMIEPE,@BUZON,'','E','REC','','','',@D
IRENTRADA+'Genral_1. edi','')
```

1.7.23 MiPO()

Devuelve el punto operacional propio y EDI de la estación.

1.7.24 PO(«Param1», «Param2»)

Devuelve datos de un punto operacional.

«Param1» contiene el punto operacional y «Param2» uno de los siguientes valores. "Nombre", "Domicilio", "Poblacion", "CP" ,"Pais", "NIF", "Regmercantil", "Padre"

1.7.25 UsuarioEdiwin()

Devuelve el Usuario de Ediwin con el que nos hemos validado. Si esta función se invoca desde fuera de EDIWIN devuelve la cadena vacia.

Ejemplo:

```
@User:=UsuarioEdiwin() -
{Almacena sobre la variable @User el valor "SUPERVISOR"}
```

1.7.26 ws_AnyadeCabecera(«Cabecera»)

Añade cabeceras a la petición SOAP. Debe llamarse primero a **ws_PreparaPeticion()**

Ejemplo:

```
ws_AnyadeCabecera('Content-Type. text/xml');
ws_AnyadeCabecera('SOAPAction. Ue4Server');
```

1.7.27 ws_AnyadeNamespace(«Prefijo», «Nombre»)

Añade “namespaces” a la petición SOAP

Ejemplo:

```
ws_AnyadeNamespace('xsi','http://www. w3. org/2001/XMLSchema-
instance')
```

1.7.28 ws_AnyadeParametro(«Nombre», «Tipo», «Valor»[,«Attributos»])

Añade los parámetros de una petición SOAP

Los tipos pueden ser los siguientes. X(string),N(int),R(double),F(date) cualquier otro tipo pasa directamente a la petición

Ejemplo:

```
ws_AnyadeParametro('Dominio','X','EDI4');
ws_AnyadeParametro('Usuario','string','SUPERVISOR', 'xsi:nil="true"')
```

1.7.29 ws_DameAtributoParametroXml(«Xml», «Parámetro»,

«Atributo»)

Devuelve el valor del atributo «Atributo» del elemento «Parámetro» del fichero «Xml»

1.7.30 ws_DameParametroXml(«Xml», «Parámetro»)

Devuelve el valor del elemento «Parámetro» del fichero «Xml»

1.7.31 ws_DamePeticion()

Esta función devuelve la petición SOAP formateada

Previamente deberá llamarse a **ws_PreparaPeticion** y **ws_AnyadeParametros**

1.7.32 ws_PreparaPeticion(«TargetNamespace», «PrefijoTargetNamespace», «Funcion», «Encoding»)

Esta función prepara los valores para hacer una llamada SOAP a un servidor

La petición debe completarse con los ws_AnyadeParametro y obtener la petición con ws_DamePeticion

El parámetro «Encoding» es opcional indica donde situar el EncodingStyle (0:No lo pone, 1:Evelope 2:Función)

Ejemplo:

```
{para registrar una sesión en ediwinS la función sería:
ws_PreparaFuncion('Ue4Server','','Te4Server_RegistraSesion')
```

1.8 OPERADORES LÓGICOS

1.8.1Eval(«Param1»)

Devuelve la evaluación de la condición «Param1».

Ejemplo:

```
Eval(Igual("A","A"))
{Devuelve 1. }
```

1.8.2Inv(«Param1»)

Devuelve el inverso binario de «Param1». El parámetro debe ser un número entero.

1.8.3No(«Param1»)

Devuelve NO lógico de «Param1»

1 equivale a verdadero y 0 a Falso.

1.8.4O(«Param1», «Param2»)

Devuelve el O lógico de «Param1» y «Param2».

1 equivale a verdadero y 0 a Falso.

1.8.5OO(«Param1», «Param2»)

Devuelve el O aritmético de «Param1» y «Param2». Los parámetros deben ser números enteros.

1.8.6Y(«Param1», «Param2»)

Devuelve el Y lógico de «Param1» y «Param2».

1 equivale a verdadero y 0 a Falso.

1.8.7YY(«Param1», «Param2»)

Devuelve el Y aritmético de «Param1» y «Param2». Los parámetros deben ser números enteros.

1.9 FECHA/HORA

1.9.1Fecha(«FormatoFecha»)

Devuelve la fecha de hoy en el formato especificado en el parámetro «FormatoFecha».

Para el «FormatoFecha» se pueden usar los caracteres "/" , "-" , ":" como separadores y los caracteres "y" (año), "m" (mes), "d" (día), "h" (hora), "n" (minutos)y "s" (segundos).

Ejemplo:

```
fecha("dd/mm/yyyy hh:nn")
{devuelve 05/08/1996 14:53}
```

1.9.2FechaEDI(«FechaEDI», «CalifEDI»)

Devuelve la fecha «Fecha» formateada de acuerdo al calificador EDI («CalifEDI») que se le pasa.

El formato del parámetro «Fecha» debe ser . "dd/mm/yyyy hh:nn:ss" o "ddmmyyyyhhnnss".

1.9.3FechaFormat(«FormatoFecha», «Fecha»)

Devuelve la fecha «Fecha» con el formato «FormatoFecha».

El formato del parámetro «Fecha» debe ser. "dd/mm/yyyy hh:nn:ss" o "ddmmyyyyhhnnss".

Para el «FormatoFecha» se pueden usar los caracteres "/" , "-" , ":" como separadores y los caracteres "y" (año), "m" (mes), "d" (día), "h" (hora), "n" (minutos)y "s" (segundos).

Ejemplo:

```
FechaFormat("dd-mm-yyyy","05/08/1996")
{devuelve 05/08/1996}
```

1.9.4FechaIEDI(«CalifEDI», «FechaEDI»)

Devuelve la fecha con formato dd/mm/yyyy hh:nn:ss.

«FechaEDI» es una fecha EDI y «CalifEDI» es el calificador de formato de la fecha.

1.9.5GetTimeZone()

Obtiene la zona horaria en formato +/-HH:MM

1.9.6JulianDate([«Fecha»])

Devuelve la fecha en formato Juliano.

El parámetro «Fecha» es opcional, en caso de llamar a la función sin parámetro, esta tomará la fecha y hora actual.

El formato del parámetro «Fecha» debe ser. "dd/mm/yyyy hh:nn:ss" o "ddmmyyyyhhnnss".

1.9.7NumSemana(«Fecha»)

Dada una «Fecha», devuelve la semana del año a la que pertenece.

El formato del parámetro «Fecha» debe ser . "dd/mm/yyyy hh:nn:ss" o "ddmmyyyyhhnnss".

1.9.8 NumSemanaISO(«Fecha»)

Dada una «Fecha», devuelve la semana del año a la que pertenece.

Se considera que la primera semana del año es aquella que tiene el primer jueves del año, según dicta la especificación ISO-8601.

El formato del parámetro «Fecha» debe ser . "dd/mm/yyyy hh:nn:ss" o "ddmmyyyyhhnnss".

1.9.9 PrimerLunes(«Semana», «FormatoFechaDestino»,[«Año»])

Devuelve la fecha del primer lunes de la semana especificada en «Semana», tomando como base el año actual. Devuelve la fecha en el formato especificado con «FormatoFechaDestino».

El parámetro «Año», es opcional, si no se pasa, se tomará el año en curso.

Ejemplo:

```
PrimerLunes(2,"dd/mm/yyyy",2004)
{devuelve 05/01/2004}
```

1.9.10 PrimerLunesISO(«Semana», «FormatoFechaDestino», [«Año»])

Devuelve la fecha del primer lunes de la semana especificada en «Semana», tomando como base el año actual. Devuelve la fecha en el formato especificado con «FormatoFechaDestino».

El parámetro «Año», es opcional, si no se pasa, se tomará el año en curso. Se considera que la primera semana del año es aquella que tiene el primer jueves del año, según dicta la especificación ISO-8601.

Ejemplo:

```
PrimerLunes(2,"dd/mm/yyyy",2004)
{devuelve 05/01/2004
```

1.9.11 RestarFechaDias(«FechaBase», «NumDias», «FormatoFechaDestino»)

Resta a la fecha especificada en «FechaBase» el número de días indicados en «NumDias» y devuelve la fecha en el formato especificado con «FormatoFechaDestino».

El formato del parámetro «FechaBase» debe ser . "dd/mm/yyyy hh:nn:ss" o "ddmmyyyyhhnnss".

1.9.12 RestarFechas(«FechaFin», «FechaOrigen»)

Resta a la fecha especificada en «FechaFin», la fecha especificada en «FechaOrigen». Devuelve el número de días entre las dos fechas.

El formato de los parámetro «FechaFin» y «FechaOrigen» debe ser . "dd/mm/yyyy hh:nn:ss" o "ddmmyyyyhhnnss".

1.9.13 SumarFecha(«FechaBase», «NumDias», «FormatoFechaDestino»)

Suma a la fecha especificada en «FechaBase», el número de días indicados en «NumDias» y devuelve la fecha en el formato especificado con «FormatoFechaDestino».

El formato del parámetro «FechaBase» debe ser . "dd/mm/yyyy hh:nn:ss" o "ddmmyyyyhhnnss".

Ejemplo:

```
SumarFecha("1/1/2004",5,"dd/mm/yyyy")
{devuelve 06/01/2004}
```

1.9.14 ValidDate(«FechaAValidar»,[«FormatoFecha», «SeparadorFecha»[,«FormatoHora», «SeparadorHora»]])

Comprueba si la fecha que recibe como parámetro es válida o no.

Devolverá 1 si la fecha es correcta 0 en caso contrario.

El formato de la fecha por defecto es . "dd/mm/yyyy hh:nn:ss" o "ddmmyyyyhhnnss", y los separadores "/" para la fecha y ":" para la hora.

Ejemplo:

```
ValidDate("33/1/2004")
{devuelve 0}
ValidDate("05-06-1968 12:15","dd-mm-yyyy","-","hh. nn",". ")
{devuelve 1}
```

1.10 ESTRUCTURAS DE CONTROL

1.10.1 For(«Inicio», «Condición», «Incremento», «Cuerpo»)

Ejecuta una vez el «Inicio», si la «Condición» se cumple ejecuta el parámetro «Cuerpo» y finalmente ejecutamos el parámetro «Incremento». Posteriormente se vuelve a comprobar la «Condición» hasta que no se cumpla.

El parámetro «Cuerpo» puede ser un conjunto de comandos si va entre comillas dobles.

Ejemplo:

```
for (@i:=1, @i <= 10, @i:=@i+1, CrearRegistro([UNH]))
```

1.10.2 SALIR(«RESULTADO»)

Finaliza la ejecución de un script, opcionalmente se puede pasar el parámetro «Resultado» que será el valor que devuelva el script al finalizar.

1.10.3 SI(«COND», «SCRIPTENTONCES»[,«SCRIPTSINO»])

Si la condición «Cond» se evalúa a 1, evaluamos «ScriptEntonces», en caso contrario evaluamos «ScriptSino».

Se evaluará única y exclusivamente «ScriptEntonces» o «ScriptSino». Se devuelve el resultado del script ejecutado.

Ambos scripts pueden ser un conjuto de instucciones separadas por punto y coma siempre y limitemos cada script con comillas dobles.

Ejemplo:

```
Si(0,"@R:=0;@R:=Sumar(@R,1)","@R:=0;@R:=Restar(@R,1)")
{Devuelve -1}
```

1.10.4 SII(«COND», «VALORENTONCES», «VALORSINO»)

Si la condición «Param1» se evalúa a 1, devuelve el resultado de evaluar «Param2», en caso contrario el resultado de evaluar «Param3».

Independientemente de «Param1» se evalúan ambos, «Param2» y «Param3» aunque la función devuelve el resultado de sólo uno de ellos.

1.10.5 TIPORECORRIDO(«PARAM1»)

Indica el modo de recorrer el interfaz origen. 0 = Preorden (por defecto), 1 = Preorden con filtros

1.11 VARIAS

1.11.1 CALLDLL(«NOMBREDLL», «NOMBREFUNCION», «COND», «PAR1»,. . .,«PARN»)

Realiza una llamada a la función «NombreFuncion» de la dll «NombreDLL». Debe cumplirse la condición «Cond». La función de la DLL debe tener una lista de parámetros de entrada y otra de salida.

Los parámetros de salida se concatenarán en un string, el cual será el resultado de **`CallDLL()`**.

Las funciones de la DLL deben seguir la siguiente estructura

```
function «NombreFuncion»(Request,Response:TStringList):boolean;
```

Ejemplo:

```
CallDLL("ENCODE. DLL","EncodeB64",1,"C:\TMP\Fichero.
TXT","C:\TMP\Fichero. B64"
```

1.11.2 CodEAN(«Param1»)

Devuelve el código EAN calculado (incluye el digito de control) para «Param1».

Ejemplo:

```
CodEan("111111111111")
{devuelve 1111111111116}
```

1.11.3 DC(«Param1»)

Devuelve el dígito de control para «Param1».

Ejemplo:

```
DC("111111111111")
{Devuelve 6}
```

1.11.4 Ejecutar(«Aplicación», «Parámetros», «Espera», «SegTimeOut», «MatarSiTimeOut», «Cond»)

Ejecuta la aplicación «Aplicación» con los parámetros «Parámetros». Si «Espera» es 1 devolvemos el Código de terminación. Se le puede poner un timeout en segundos «SegTimeOut». Si vence este timeout, el código de terminación será 258. Además si «MatarSiTimeOut» es 1 y hemos añadido un TimeOut, la aplicación lanzada finalizará.

Ejemplo:

```
Ejecutar("CALC. EXE","",1,10,1,1)
{Ejecutará la calculadora y la cerrará a los 10 segundos
```

1.11.5 EvaluaScript(«CodigoScript»[,«Condicion»])

Evaluamos el script «CodigoScript» y devolvemos el resultado del mismo. Opcionalmente podemos pasar un segundo parámetro «Condicion». Si condición es distinto de "1" el script no se ejecutará.

Ejemplo.

```
@Script:="@Res:=Mult(3,4);Muestra(@Res)"
EvaluaScript(@Script);
```

1.11.6 Muestra(«Texto»)

Muestra una ventana con «Texto». Para la ejecución del mapeado. Solo muestra la ventana en caso de ejecutarse el mapeado en modo interactivo.

Ejemplo:

```
Muestra("El total es. "+@Total)
```

1.11.7 ParamCount()

Devuelve el número de parámetros pasados al programa en la línea de comando. Por ejemplo, si ejecutamos la aplicación con Map4. exe A B C ParamCount() devuelve 3 parámetros

1.11.8 ParamStr(«Indice»)

Devuelve el parámetro de la línea de comando que corresponde con el índice, o una cadena vacía si el índice es mayor que el número de parámetros. Por ejemplo, el índice “2” devuelve el segundo parámetro de la llamada. Por ejemplo, si ejecutamos la aplicación con Map4. exe A B C ParamStr(2) devuelve B

Si indicamos como parametro el “0” devuelve la ruta completa y el nombre del programa que se esta ejecutando. Por ejemplo, ParamStr(0) en el caso anterior devuelve *C:\EDIWIN4\BMAP4. EXE*

1.11.9 TraduceSegmento(«Segmento» [,«Formato»], «NombreCampo»)

Devuelve el valor del campo «NombreReg. NombreCampo» en el segmento de servicio «Segmento».

El parámetro «Formato» puede tomar el valor EDIFACT o X12, según el tipo de segmento. Por defecto será EDIFACT.

Los campos repetidos dentro de un segmento/registro se especifican con su número de repetición, ejemplo. UNB/0007. 1 para indicar el calificador del destino del intercambio.

Ejemplo:

```
TraduceSegmento([UNH/$CONTROL$. ENV_SEGSERV],'EDIFACT',[UNB. 0004])
```