DEFINICIÓN DEL FORMATO DE INTERCAMBIO ESTÁNDAR DE BASES DE DATOS DE CONSTRUCCIÓN FIEBDC-3/2024.

PRESENTACIÓN.

La siguiente DEFINICIÓN del Formato de Intercambio ESTÁNDAR de Bases de Datos de CONSTRUCCIÓN, añade a la versión anterior las especificaciones indicadas desde el Acta 28 hasta el Acta 30 de la Comisión Técnica Permanente, posteriormente refrendadas/modificadas en las respectivas ASAMBLEAS de la Asociación FIEBDC (Actas 18/6/2021, 17/6/2022 y 15/6/2023).

Entrada en vigor el día 1 de enero de 2024, como VERSIÓN 3/2024.

Para facilitar su lectura, se indica en letra de color azul aquellos párrafos nuevos o modificados respecto a la especificación anterior FIEBDC-3/2020.

Este documento se pone a disposición de usuarios y empresas, con la única condición de que cualquier implementación informática del presente formato debe recoger tanto la entrada como la salida de datos.

Este formato pretende abarcar toda la INFORMACIÓN contenida en las actuales bases de datos de CONSTRUCCIÓN. No todos los desarrolladores de bases de datos necesitarán utilizar todas las posibilidades del formato, así como tampoco todos los programas de mediciones y presupuestos harán uso de toda la INFORMACIÓN suministrada.

Se prevé, además, dentro del propio formato, la posibilidad de ampliación; manteniendo en lo posible la compatibilidad entre versiones en el caso de tratar nuevos contenidos que se prevean en un futuro.

FORMATO FIEBDC-3. ESPECIFICACIÓN.

Toda la INFORMACIÓN necesaria para reconstruir completamente una base de datos u obra en soportes físico y lógico distintos a aquellos en los cuales se produjo la INFORMACIÓN es el objetivo del formato FIEBDC, Formato de Intercambio ESTÁNDAR de Bases de Datos de CONSTRUCCIÓN.

La INFORMACIÓN de una base de datos, obra o certificación se dispondrá en un archivo en formato FIEBDC, con la extensión ".BC3".

La única limitación de tamaño del archivo será la máxima que permita el soporte físico utilizado para su transporte. Si se utiliza algún tipo de compresor de archivos, se deberá incluir en el mismo soporte el descompresor o utilizar un formato autodescomprimible.

El juego de caracteres a emplear en los campos CODIGO será el definido por MS-DOS 6.0, incluyendo A-Z, a-z, 0-9, ñ, Ñ, < - > (ASCII-45), < . > (ASCII-46), < $ > (ASCII-36), < # > (ASCII-35), < %> (ASCII-37), < & > (ASCII-38), < _ > (ASCII-95). Excluyendo cualquier otro carácter como espacio, tabulador, etc.

El fin de línea será el ESTÁNDAR de los archivos MS-DOS (ASCII-13 y ASCII-10). El fin de archivo se marcará según el mismo ESTÁNDAR (ASCII-26). El único carácter de control adicional que se permitirá será el tabulador (ASCII-9).

Cada archivo estará compuesto de registros, zonas de texto entre el carácter de principio de registro < ~ > (ASCII-126) y el siguiente principio de registro o fin de archivo. Los archivos deberán contener registros completos, es decir, la división de archivos se deberá realizar al

comienzo de un registro (carácter < ~ >).

Cada registro estará compuesto de campos separados por caracteres < | > (ASCII-124). Todo campo con INFORMACIÓN tendrá que finalizar con el separador de campos y el registro deberá contener todos los separadores de campos anteriores, aunque no contengan INFORMACIÓN. No es necesario disponer de finalizadores de los campos posteriores al último con INFORMACIÓN.

Cada campo estará compuesto de subcampos separados por caracteres < \ > (ASCII-92). El separador final, entre el último dato de un campo y el fin de campo es opcional.

El primer campo de cada registro es la cabecera de registro, una letra mayúscula que identifica el tipo de registro.

Se ignorará cualquier INFORMACIÓN entre el último separador de campos de un registro (carácter < | >) o el comienzo del archivo y el comienzo del siguiente registro (carácter < ~>).

Se ignorarán los caracteres blancos (32), tabuladores (9) y de fin de línea (13 y 10), delante de los separadores < ~ >, < | > y < \ >.

No se podrán actualizar parcialmente campos de segundo orden (subcampos). Deberá actualizarse la INFORMACIÓN completa de un campo en cualquiera de los registros

La disposición de registros dentro de un archivo es completamente libre, pero se garantizará la lectura secuencial de los mismos para evitar ambigüedades en las sustituciones de INFORMACIÓN.

Los campos vacíos se considerarán SIN INFORMACIÓN, no con INFORMACIÓN nula, esto permite producir archivos de actualización que contengan únicamente la INFORMACIÓN en alguno de sus campos y, por supuesto, el CODIGO de referencia.

Para anular un campo numérico deberá aparecer explícitamente el valor 0 (cero).

Para anular un campo alfanumérico deberá aparecer explícitamente el ROTULO NUL.

Con la finalidad de poder recoger varias certificaciones a la vez, el nombre del archivo que contiene la certificación (a origen) se tiene que llamar igual que el del presupuesto añadiéndole (concatenándole) “#certificacionNNN” donde NNN sería el número de la certificación. Se recomienda que el archivo se coloque en la misma carpeta que la del presupuesto, ya que permitiría importar a la vez un presupuesto y todas sus certificaciones (o solo las que se seleccionen). El archivo que contiene la certificación es igual que el de un presupuesto y solo se diferencia en el registro ~V, en el que pondrá que es una certificación y el número y la fecha de la misma. Este contiene toda la información (~D, ~M, ~C, ~T), y no tan solo los registros de las unidades certificadas (~D, ~M).

Con la finalidad de permitir la transferencia desde las bases de datos en Internet a las aplicaciones en ordenadores locales utilizando FIEBDC, se indica que al hacer drag&drop en el icono del FIE sobre una aplicación, la aplicación en el evento drop recibe una url que finaliza con el id_concepto con la extensión BC3, a la cual el programa tendrá que hacer una petición que le devolverá un BC3.”

CONVENIO DE NOTACIÓN.

[a]	Indica nada o "a"
{a}	Indica cero o más ocurrencias de "a"
<a>	Indica una o más ocurrencias de "a"
(<DD>c)	Tamaño máximo en número de caracteres del campo

Todos los valores numéricos irán sin separadores de miles y con el carácter punto '.' entre la parte entera y la decimal.

~V. REGISTRO TIPO PROPIEDAD Y VERSIÓN.

Este registro se utiliza para documentar la procedencia y el formato de los archivos y, cuando exista, se dispondrá al comienzo del primer archivo.

~V | [PROPIEDAD_ARCHIVO] | VERSION_FORMATO [\ DDMMAAAA] | [PROGRAMA_EMISION] | [CABECERA] \ { ROTULO_IDENTIFICACION \ } | [JUEGO_CARACTERES] | [COMENTARIO] | [TIPO INFORMACIÓN] | [NÚMERO CERTIFICACIÓN] | [FECHA CERTIFICACIÓN] | [URL_BASE] |

PROPIEDAD_ARCHIVO: Redactor de la base de datos u obra, fecha…

VERSION_FORMATO: VERSION del formato del archivo, la actual es FIEBDC-3/2016.

DDMMAAAA: DD representa el día con dos dígitos, MM el mes y AAAA el año. Si la fecha tiene 6 o menos dígitos, el año se representará con dos dígitos (AA), interpretándose con el criterio “80/20”. Esto es, cualquier año que sea igual o superior a 80 corresponderá al siglo XX y cualquier año que sea menor de 80 corresponderá al siglo XXI. Si la fecha tiene menos de 5 dígitos representa mes y año únicamente (MMAA), si tiene menos de tres, solo el año (AA). Si se identifica la fecha con un número impar de dígitos, se completará con el carácter cero por la izquierda. Para representar una fecha sin un día o mes específico, se utilizará un doble cero en cada caso.

Ejemplos:

12062000	12 de junio de 2000
120699	12 de junio de 1999
00061281	junio de 1281
061281	6 de diciembre de 1981
401	abril de 2001

PROGRAMA_EMISION: Programa y/o empresa que genera los ficheros en formato BC3.

CABECERA: Título general de los ROTULOS_IDENTIFICACION.

ROTULO_IDENTIFICACION: Asigna secuencialmente títulos a los valores definidos en el campo PRECIO del registro ~C, y los conjuntos de campos de números de decimales del registro ~K, que tal como se indica en su ESPECIFICACION, puede representar distintas épocas, ámbitos geográficos, etc., estableciéndose una relación biunívoca entre ambos. Véanse los anexos 5 (Ámbitos territoriales) y 6 (Divisas).
En el caso de que en el registro ~V existan más campos ROTULO_IDENTIFICACION que campos PRECIO en el registro ~C o que conjuntos de campos de decimales en el registro ~K, se entenderá que el PRECIO y los conjuntos de campos de decimales de dicho resto serán iguales al último definido.

JUEGO_CARACTERES: Asigna si el juego de caracteres a emplear es el definido para D.O.S., cuyos identificadores serán 850 ó 437, o es el definido para Windows, cuyo identificador será ANSI. En caso de que dicho campo esté vacío se interpretará, por omisión, que el juego de caracteres a utilizar será el 850 por compatibilidad con versiones anteriores.

COMENTARIO: Contenido del archivo (base, obra...).

TIPO INFORMACIÓN: Índice del tipo de información a intercambiar.
Se definen los siguientes tipos:

1 Base de datos.
2 Presupuesto.
3 Certificación (a origen).
4 Actualización de base de datos.

NÚMERO CERTIFICACIÓN: Valor numérico indicando el orden de la certificación (1, 2, 3...) Solo tiene sentido cuando el tipo de información es Certificación.

FECHA CERTIFICACIÓN: Fecha de la certificación indicada en el campo número certificación. Solo tiene sentido cuando el tipo de información es Certificación. La fecha se definirá con el mismo formato que el campo DDMMAAAA de este registro.

URL_BASE: Url a partir de la cual se encontrarán los documentos y los gráficos.

~K. REGISTRO TIPO COEFICIENTES.

Indica el número de decimales en cada campo numérico. Cuando el campo numérico aparece con signo negativo, indica número máximo de decimales. En caso contrario indica número exacto de decimales.

Cuando en un archivo FIEBDC aparece una cantidad con más cifras decimales de las correspondientes según el registro ~K, se debe especificar como error, ya que lo que se exporta debe coincidir con el registro ~K.

Cuando en un archivo FIEBDC no exista este registro, se recomienda aplicar los decimales por defecto establecidos en el formato.

~K | { DN \ DD \ DS \ DR \ DI \ DP \ DC \ DM \ DIVISA \ } | CI \ GG \ BI \ BAJA \ IVA | { DRC \ DC \ \ DFS \ DRS \ \ DUO \ DI \ DES \ DN \ DD \ DS \ DSP \ DEC \ DIVISA \ } | [n] |

Este registro incluye el campo 1 por compatibilidad con versiones anteriores del formato, aunque los programas deben leer el campo 3 por ser más completo y en su defecto el campo 1.

Conceptos previos

Unidad de obra: Cualquier elemento simple o elemento compuesto, con o sin costes indirectos, que se utiliza en un presupuesto.
Elemento compuesto: Todo elemento constructivo que contiene una descomposición y que no es ni raíz ni capitulo.
Elemento simple: Todo elemento constructivo que no contiene una descomposición y que no es ni raíz ni capitulo.

Definiciones

DN	Decimales del campo número de partes iguales de la hoja de mediciones. Por defecto 2 decimales.
DD	Decimales de dimensiones de las tres magnitudes de la hoja de mediciones. Por defecto 2 decimales.
DS	Decimales de la línea de subtotal o total de mediciones. Por defecto 2 decimales.
DR	Decimales de rendimiento y factor en una descomposición. Por defecto 3 decimales.
DI	Decimales del importe resultante de multiplicar rendimiento x precio del concepto. Por defecto 2 decimales
DP	Decimales del importe resultante del sumatorio de los costes directos del concepto. Por defecto 2 decimales
DC	Decimales del importe total del concepto. (CD+CI). Por defecto 2 decimales
DM	Decimales del importe resultante de multiplicar la medición total del concepto por su precio. Por defecto 2 decimales
DIVISA	Es la divisa expresada en el mismo modo que las abreviaturas utilizadas por el BCE (Banco Central Europeo), que en su caso deberán coincidir con las del registro ~V. En el Anexo 6 se indican las actuales.

CI Costes Indirectos, expresados en porcentaje.
GG Gastos Generales de la Empresa, expresados en porcentaje.
BI Beneficio Industrial del contratista, expresado en porcentaje.
BAJA Coeficiente de baja o alza de un presupuesto de adjudicación, expresado en porcentaje.
IVA Impuesto del Valor Añadido, expresado en porcentaje.

DRC Decimales del rendimiento y del factor de rendimiento de un presupuesto, y decimales del resultado de su multiplicación. Por defecto 3 decimales.
DC Decimales del importe de un presupuesto, de sus capítulos, subcapítulos, etc. y líneas de medición (unidades de obra excluidas), y decimales de los importes resultantes de multiplicar el rendimiento (o medición) total del presupuesto, sus capítulos, subcapítulos, etc. y líneas de medición (unidades de obra excluidas) por sus precios respectivos. Por defecto 2 decimales.
DFS Decimales de los factores de rendimiento de las unidades de obra y de los elementos compuestos. Por defecto 3 decimales.
DRS Decimales de los rendimientos de las unidades de obra y de los elementos compuestos, y decimales del resultado de la multiplicación de dichos rendimientos por sus respectivos factores. Por defecto 3 decimales.
DUO Decimales del coste total de las unidades de obra. Por defecto 2 decimales.
DI Decimales de los importes resultantes de multiplicar los rendimientos totales de los elementos compuestos y/o elementos simples por sus respectivos precios, decimales del importe resultante del sumatorio de los costes directos de la unidad de obra y decimales de los costes indirectos. Decimales de los sumatorios sobre los que se aplican los porcentajes. Por defecto 2 decimales.
DES Decimales del importe de los elementos simples. Por defecto 2 decimales.
DN Decimales del campo número de partes iguales de la hoja de mediciones. Por defecto 2 decimales.
DD Decimales de dimensiones de las tres magnitudes de la hoja de mediciones. Por defecto 2 decimales.
DS Decimales del total de mediciones. Por defecto 2 decimales.
DSP Decimales de la línea de subtotal de mediciones. Por defecto 2 decimales.
DEC Decimales del importe de los elementos compuestos. Por defecto 2.
DIVISA Es la divisa expresada en el mismo modo que las abreviaturas utilizadas por el BCE (Banco Central Europeo), que en su caso deberán coincidir con las del registro ~V. En el Anexo 6 se indican las actuales.

n Es el número de la opción de la función BdcGloParNumero que se refiere al concepto divisa.

Si un importe viene con un número mayor de decimales del estipulado en el registro ~K, se debe redondear al número de decimales indicado en dicho registro ~K (criterio <5 queda igual y >= 5 suma), y las operaciones que se hagan del importe serán con este valor redondeado.

Para relacionar una determinada divisa con su convención de decimales, deberá coincidir el orden en que aparezcan las diferentes divisas en el registro ~K, con el orden indicado en el registro ~V o con el orden de las opciones de la función BdcGloOpcNumero, en función de su caso. Si en el registro ~V hay más ROTULOS que divisas en el registro ~K, los precios sin su correspondiente divisa tomarán la última divisa definida en el registro ~K.

Ejemplo

Ejemplo de una base de precios que facilita importes de 2 ámbitos territoriales en 2 divisas (Se indican en negrita los campos afectados):

~V | PROPIEDAD_ARCHIVO | VERSION_FORMATO \ DDMMAA | PROGRAMA_EMISION | [**Precios de diferentes ámbitos territoriales en diferentes divisas**] \ { **B-eur** \ **T-eur** \ **B-usd** \ **T-usd** } | JUEGO_CARACTERES | [TIPO INFORMACIÓN] | [NÚMERO CERTIFICACIÓN] | [FECHA CERTIFICACIÓN

] |

~C | { CODIGO \ } | UNIDAD | RESUMEN | { **120** \ **108** \ **102,8** \ **92,52** } | { FECHA \ } | TIPO |

~K | { DN \ DD \ DS \ DR \ DI \ DP \ DC \ DM \ DIVISA \ } | CI \ GG \ BI \ BAJA \ IVA | { DRC \ DC \ DFS \ DRS \ DUO \ DI \ DES \ DN \ DD \ DS \ DSP \ DEC \ **eur** \ DRC \ DC \ DFS \ DRS \ DUO \ DI \ DES \ DN \ DD \ DS \ DSP \ DEC \ **eur** \ DRC \ DC \ DFS \ DRS \ DUO \ DI \ DES \ DN \ DD \ DS \ DSP \ DEC \ **usd** \ } [n] |

Esquema aclaratorio

Concepto	***Importe***
Capítulo raíz (##)	DC (2)

Concepto interviniendo en una línea de descomposición:

Concepto	***Factor***	***Rendimiento***	***FR*** Factor*Rend	***Precio***	***Importe*** FR*Precio
Presupuesto unitario	DRC(3)	DRC(3)	DRC(3)	DC(2)	DC(2)
Capítulo (#)					DC(2)*
Unidad de obra	DFS(3)	DS(2)	DS(2)	DUO(2)	DC(2)
Elemento compuesto	DFS(3)	DRS(3)	DRS(3)	DEC(2)	DI(2)
Elemento simple	DFS(3)	DRS(3)	DRS(3)	DES(2)	DI(2)

* El importe del Capítulo (#) será el sumatorio de los importes de sus componentes.

Cálculo de precio de una unidad de obra:

Unidad de Obra	**Coste directo**	**Coste indirecto**	***Precio*** Coste directo + Coste indirecto
Elemento compuesto	DI(2)	DI(2)	DUO(2)
Elemento simple	DI(2)	DI(2)	DUO(2)

Mediciones	***Cantidad***	***Dimensiones***	***Subtotal***	***Total***
Líneas de medición	DN(2)	DD(2)	DSP(2)	DS(2)

Se ha añadido los decimales que se ponen por defecto según el formato después de cada término.

En la carpeta RegistroK_ejemplo que acompaña a este documento se encuentra un ejemplo de presupuesto.

~C. REGISTRO TIPO CONCEPTO.

Este registro contiene la INFORMACION básica de un concepto de cualquier tipo, material, auxiliar, partida, capítulo, entidad, documento, etc., tanto en su VERSION paramétrica como DEFINICION tradicional.

~C | CODIGO { \ CODIGO } | [UNIDAD] | [RESUMEN] | { PRECIO \ } | { FECHA \ } | [TIPO] |

CODIGO: CODIGO del concepto descrito. Un concepto puede tener varios CODIGOs que actuarán como sinónimos, este mecanismo permite integrar distintos sistemas de clasificación.

Puede tener un máximo de 20 caracteres.

Para distinguir el concepto tipo raíz de un archivo, así como los conceptos tipo capítulo, se ampliará su CODIGO con los caracteres '##' y '#' respectivamente; quedando dicha NOTACION reflejada obligatoriamente en el registro tipo ~C, siendo opcional en los restantes registros del mismo concepto.
Las referencias a un CODIGO con y sin # y/o ##, se entienden únicas a un mismo concepto.

Únicamente puede haber un concepto raíz en una base de datos u obra.

Si un código cuenta con un carácter '#' intercalado, se entenderá que corresponde al conjunto CODIGO_ENTIDAD # CODIGO_CONCEPTO definido en el registro de Tipo Relación Comercial (registro ~O) o en la función BdcComercCodigo.

UNIDAD: Unidad de medida. Existe una relación de unidades de medida recomendadas, elaborada por la Asociación de Redactores de Bases de Datos de CONSTRUCCION. Véase el Anexo 7 sobre Unidades de Medida

RESUMEN: Resumen del texto descriptivo. Cada soporte indicará el número de caracteres que admite en su campo resumen. Se recomienda un máximo de 64 caracteres.

PRECIO: Precio del concepto. Un concepto puede tener varios precios alternativos que representen distintas épocas, ámbitos geográficos, etc., definidos biunívocamente respecto al campo [CABECERA \ {ROTULO_IDENTIFICACION\} del registro ~V. Cuando haya más de un precio se asignarán secuencialmente a cada ROTULO definido; si hay más ROTULOS que precios, se asignará a aquellos el último precio definido. En el caso que el concepto posea descomposición, este precio será el resultado de dicha descomposición y se proporcionará, de forma obligatoria, para permitir su comprobación. En caso de discrepancia, tendrá preponderancia el resultado obtenido por la descomposición, tal como se indica en el registro Tipo Descomposición, ~D, y complementariamente se podría informar al usuario de dicha situación. Esto se aplica también a los conceptos tipo capítulo y concepto raíz de una Obra o Presupuesto. Como excepción a esta regla está el intercambio de mediciones no estructuradas (véase la descripción del registro Tipo Mediciones, ~M).

FECHA: Fecha de la última actualización del precio. Cuando haya más de una fecha se asignarán secuencialmente a cada precio definido, si hay más precios que fechas, los precios sin su correspondiente fecha tomarán la última fecha definida.

Las fechas se definirán en el formato DDMMAAAA: DD representa el día con dos dígitos, MM el mes y AAAA el año. Si la fecha tiene 6 o menos dígitos, el año se representará con dos dígitos (AA), interpretándose con el criterio “80/20”. Esto es, cualquier año que sea igual o superior a 80 corresponderá al siglo XX y cualquier año que sea menor de 80 corresponderá al siglo XXI. Si la fecha tiene menos de 5 dígitos representa mes y año únicamente (MMAA), si tiene menos de tres, solo el año (AA). Si se identifica la fecha con un número impar de dígitos, se completará con el carácter cero por la izquierda. Para representar una fecha sin un día o mes específico, se utilizará un doble cero en cada caso.

Ejemplos:

12062000	12 de junio de 2000
120699	12 de junio de 1999
00061281	junio de 1281
061281	6 de diciembre de 1981
401	abril de 2001

TIPO: Tipo de concepto. Inicialmente se reservan los siguientes tipos:

0 Sin clasificar
1 Mano de obra
2 Maquinaria y medios auxiliares
3 Materiales

4 Componentes adicionales de residuo
5 Clasificación de residuo

También se permite (y aconseja) utilizar la clasificación indicada por el BOE y la CNC en índices y fórmulas polinómicas de revisión de precios, así como los aconsejados por la Asociación de Redactores de Bases de Datos de la Construcción. En el Anexo 4 aparecen los tipos actualmente vigentes.

~D. REGISTRO TIPO DESCOMPOSICIÓN.

Este registro contiene la descomposición de un concepto en otros a través de una o dos cantidades. El mismo registro lo emplearemos para definir la descomposición de un concepto tipo unidad de obra en conceptos tipo materiales, mano de obra, maquinaria y auxiliares y para la descomposición de un concepto tipo capítulo en conceptos tipo unidad de obra o subcapítulo.

Los redactores de un presupuesto, de un banco de precios o el exportador a formato FIEBDC de un programa de presupuestos deben ser conscientes de no aplicar dos veces el concepto de costes indirectos en una misma unidad de obra, ya que el formato permite la posibilidad de disponer los costes indirectos tanto en el registro ~K como en la descomposición de las unidades de obra.

Si se define como concepto derivado un concepto que no posee partes variables en su definición (no es un paramétrico) ni es un capítulo, entonces, un concepto derivado sólo puede contener en su descomposición a conceptos derivados.

~D | CODIGO_PADRE | < CODIGO_HIJO \ [FACTOR] \ [RENDIMIENTO] \ > | < CODIGO_HIJO \ [FACTOR] \ [RENDIMIENTO] \ {CODIGO_PORCENTAJE ; } \ > |

Este registro incluye el campo 2 por compatibilidad con versiones anteriores del formato, aunque los programas deben leer el campo 3 por ser más completo y en su defecto el campo 2.

CODIGO_PADRE: CODIGO del concepto descompuesto.

CODIGO_HIJO: CODIGO de cada concepto que interviene en la descomposición.

FACTOR: Factor de rendimiento, por defecto 1.0

RENDIMIENTO: Número de unidades, rendimiento o medición, por defecto 1.0

Cuando un capítulo interviene en una línea de descomposición, este no se ve afectado ni por el factor ni por el rendimiento. Estos campos se intercambian con el valor por defecto 1.0 en dicha línea de descomposición.

Cuando el CODIGO_HIJO incluye el carácter ‘%’ o el carácter ‘&’ es un porcentaje sobre las líneas anteriores de la descomposición. El código de los porcentajes tiene tres partes:

1) Prefijo, que forma una máscara indicando sobre qué elementos se aplica el porcentaje. Si el prefijo es nulo, el porcentaje se aplica a todas las líneas anteriores.

2) Un carácter, que puede ser ‘&’ (porcentaje acumulable), o ‘%’ (porcentaje no acumulable).

3) Una serie de caracteres libres que permite diferenciar un porcentaje de otro.

Ejemplo: OP%N0001

OP: Sobre todas las líneas anteriores cuyo código comience por OP.
%: Porcentaje no acumulable
N0001: Código diferenciador.
El rendimiento será el porcentaje que se aplica sobre las líneas anteriores a la actual y que queden afectadas por la máscara.

Ejemplo de una línea de descomposición: O%N0001 \\0.03\

Esta línea representa un porcentaje del 0.03 por uno (3%) de todas las líneas anteriores a la actual, incluso porcentajes, cuyo código comience por O y cuyo texto estará en la definición del código 'O%N0001'.

Ejemplo: ~C | O%N0001 | % | Medios auxiliares |

A efectos de cálculo de precios compuestos los porcentajes acumulables y no acumulables se comportan de la misma forma. La diferencia entre ellos únicamente se manifiesta en el cálculo de cantidades de simples que hay en un presupuesto, para ello se consideraran los acumulables (‘&’) como porcentajes de perdidas, roturas u otros casos que impliquen una mayor cantidad de los recursos en líneas superiores. Los no acumulables (‘%’) se pueden referir a pequeño material u otros casos que no impliquen una mayor necesidad de recursos en líneas superiores.

La existencia del factor en líneas de descomposición y el uso casi nulo que se ha hecho de los porcentajes acumulables (‘&’) hace que éstos se mantengan por razones históricas, pero se desaconseja su uso.

CODIGO_PORCENTAJE: CODIGOS de concepto porcentaje que se aplican a esta línea de descomposición. Tiene codificación libre ya que el código no se utiliza para determinar sobre que líneas aplicar el porcentaje separados con el carácter < ; > (ASCII-59). Cuando en un descompuesto aparece explícitamente el código porcentaje en las líneas sobre las que se aplica estas determinarán de forma unívoca la aplicación del porcentaje y será preciso que en el descompuesto aparezca otra línea con el código porcentaje en el campo CODIGO_HIJO. En este caso no se aplican las reglas sobre composición del código de los porcentajes y si las que se usan para el resto de los conceptos. El porcentaje se aplicará a la suma de las líneas que la preceden y que lleven el CODIGO_PORCENTAJE. No puede aparecer un concepto porcentaje en la posición CODIGO_PORCENTAJE después de aparecer en CODIGO_HIJO. Se pueden aplicar más de un concepto porcentaje sobre la misma línea.

En el caso que el importe de un concepto se pueda obtener mediante el campo precio del registro ~C y también mediante el registro ~D, será el importe obtenido de este último el que primará sobre el importe del primero.

~Y. REGISTRO TIPO AÑADIR DESCOMPOSICIÓN.

Con este registro se pueden añadir líneas de descomposición, el registro tipo ~D cambia la descomposición completa. Para añadir conceptos nuevos a una base de datos, además de definir los registros ~C, ~T, ~L, ~D… deberíamos posicionar los nuevos conceptos en el capítulo o capítulos donde queramos situarlos, para ello, es necesario un registro que nos permita añadir una o varias líneas de descomposición por cada capítulo donde queramos posicionar un nuevo concepto.

~Y | CODIGO_PADRE | < CODIGO_HIJO \ [FACTOR] \ [RENDIMIENTO] \ > | < CODIGO_HIJO \ [FACTOR] \ [RENDIMIENTO] \ { CODIGO_PORCENTAJE ; } \ >|

Todos los campos tienen el mismo significado que en el registro tipo ~D.

~R. REGISTRO TIPO DESCOMPOSICIÓN DE RESIDUOS.

Este registro sirve para asignar los componentes que generan residuo a un concepto y adicionalmente para proporcionar propiedades que dependen de la relación padre-hijo.

~ R | CODIGO_PADRE | { TIPO_DESCOMPOSICION \ CODIGO_HIJO \ { PROPIEDAD \ VALOR \ [UM] \ } | } |

CODIGO_PADRE: Código del concepto, contenido en la base de datos.

TIPO_DESCOMPOSICION: Tipo de componentes en los que se puede descomponer un concepto. Inicialmente se definen los siguientes:

- 0 Residuo de componente de colocación.
- 1 Residuo de componente de demolición.
- 2 Residuo de componente de excavación.
- 3 Residuo de componente de embalaje.

CODIGO_HIJO: Código del concepto que interviene como componente de CODIGO_PADRE en la relación TIPO_DESCOMPOSICION, contenido en la base de datos.

PROPIEDAD: Información técnica de CODIGO_HIJO relacionado con CODIGO_PADRE. En el cálculo del residuo utilizaremos dos propiedades:

- r Rendimiento.
- rp Factor de residuo. Porcentaje total de residuo de colocación. Se utilizará rp cuando CODIGO_HIJO corresponda a un residuo de colocación.

VALOR: Valor alfabético o numérico de la propiedad.

UM: Unidad de medida. En el caso de que los valores de la propiedad sean numéricos se indicará de acuerdo con el Sistema Internacional de Unidades de Medida (ver anexo 7).

Cálculo de residuos.

Para calcular los residuos en conceptos descritos como discretos utilizaremos los registros ~R y ~X. Para conceptos descritos con descripciones paramétricas compiladas en DLL utilizaremos BdcNumProp, BdcPropValString, BdcNumComponentes, BdcCodigoComponente, BdcNumPropComponente y BdcComponentePropValString.

Un elemento (material, elemento unitario…) puede generar uno o varios residuos.
Cada residuo debe tener definido una clasificación LER, un volumen (m3) y una masa (kg).
Un residuo, aparte de por LER, también se puede clasificar por otros sistemas de clasificación.
El residuo obtenido se calcula en la unidad de medición del elemento.

Residuo de un elemento simple.

El residuo de un elemento simple se obtiene de valores directos obtenidos del campo CODIGO_IT del registro ~X (propiedades: ler, v, m).

donde,

LER del residuo de un elemento.

Clasificación LER del residuo (lista europea de residuos, según Orden MAM/304/2002). El LER del residuo de un elemento puede ser un valor directo escogido de la lista LER o bien el listado acumulado de valores LER obtenidos de sus componentes. En los elementos simples el LER es siempre un valor directo.

[ ].ler = valor directo, o
si [ ].ler = nulo, [ ].ler = acumulación de LER de sus componentes

Volumen del residuo de un elemento.

Volumen que ocupa el elemento en el espacio.
El Volumen puede ser un valor directo o bien calcularse a partir de la suma del volumen de cada componente por la cantidad de cada componente en el elemento (Qobra). En los elementos simples el volumen es siempre un valor directo.
Cálculo del volumen de un elemento (m3 por ua, unidad de medición):

[ ].v = valor directo, o
si [ ].v = nulo, [ ].v = suma ([C].v * [ ].[C].Qobra (ua))

donde,
- C = componente
- Qobra (cantidad de cada componente en el elemento) = Rendimiento*Factor – Residuo

Masa del residuo de un elemento.

La masa puede ser un valor directo o bien calcularse a partir de la suma de la masa de cada componente por la cantidad de cada componente en el elemento (Qobra). En los elementos simples la masa es siempre un valor directo.
Cálculo de la masa de un elemento (kg por ua, unidad de medición):

[ ].m = valor directo, o
si [ ].m = nulo, [ ].m = suma ([C].m * [ ].[C].Qobra (ua))

donde,
- C = componente
- Qobra = Rendimiento*Factor – Residuo

Residuo de un elemento compuesto.

El residuo de un elemento compuesto se obtiene en el registro ~R del sumatorio de los residuos de sus 'componentes de residuo'.

Los 'componentes de residuo' inicialmente definidos son:

- 'Componentes de colocación': Su residuo es el material que se tira en el proceso de formación de la obra.
  Estos componentes deben existir en el registro ~D.
  Se tipifican en el registro ~C con el TIPO: 0, 1, 2 o 3.
  Para calcular el residuo de un 'componente de colocación' (C), aparte del 'Rendimiento' y el 'Factor', que se obtienen del registro ~D, necesitamos una nueva variable que es el 'Factor de residuo' (rp), que si existe se obtiene del registro ~R.
  El residuo de un 'componente de colocación' (C) es:
  [ ].[C].Residuo = [ ].[C].Rendimiento*Factor * [ ].[C].Factor de Residuo
  donde, Factor de Residuo es el porcentaje total de residuo de colocación.

- 'Componentes adicionales de residuo':
  - Componentes de demolición: Su residuo es el que se genera de la actividad de derribo de un elemento constructivo.
  - Componentes de excavación: Su residuo es el que se genera de la actividad de excavación de un elemento constructivo.
  - Componentes de embalaje: Su residuo es el que se genera del embalaje en el que vienen envueltos los materiales de un elemento constructivo.

  Estos componentes no aparecen incluidos en el registro ~D.
  Se tipifican en el registro ~C con el TIPO: 4.
  Para calcular el residuo de un 'componente de residuo' (R) se necesita su rendimiento, que se obtiene del registro ~R.

El residuo de un ‘componente adicional de residuo’ (R) es:

[ ].[R].Residuo = [ ].[R].Rendimiento

Es decir, el residuo de un ‘componente adicional de residuo’ (R) en un elemento es la cantidad de componente de embalaje, excavación y demolición en el elemento.

Esquema-ejemplo del cálculo del residuo de un elemento compuesto:

Padre: “Demolición de pared y posterior construcción de pared de ladrillo con mortero…”

Hijo1: “Demolición 1 …….” (~C tipo 4) = LER (010407)
Masa unit (7) * rendimiento (3) = masa
Volumen unit (20) * rendimiento (3) = volumen
Hijo2: “Demolición 2 …….” (~C tipo 4) = LER (010502)
Masa unit (8) * rendimiento (1.5) = masa
Volumen unit (21) * rendimiento (1.5) = volumen
Hijo3: “Ladrillo …….” (~C tipo 3) = LER (010301)
Masa unit (9) * rendimiento * factor * rp (1.3) = masa
Volumen unit (22) * rendimiento * factor * rp (1.3) = volumen
Hijo4: “Mortero …….” (~C tipo 3) = LER (010702)
Masa unit (10) * rendimiento * factor * rp (1.5) = masa
Volumen unit (23) * rendimiento * factor * rp (1.5) = volumen
Hijo5: “Embalaje 1 …….” (~C tipo 4) = LER (010801)
Masa unit (11) * rendimiento (7) = masa
Volumen unit (24) * rendimiento (7) = volumen
Hijo6: “Embalaje 2 …….” (~C tipo 4) = LER (010903)
Masa unit (12) * rendimiento (8.2) = masa
Volumen unit (25) * rendimiento (8.2) = volumen

Donde los valores ‘LER’, ‘masa unit’ y ‘volumen unit’ se obtienen del registro ~X.
Los ‘hijos’, ‘rp’ y los ‘rendimientos’ de los ~C tipo 4 se obtienen del registro ~R.

Su expresión sería:

~R | Padre | 1 \ Hijo1 \ r \ 3 \ \ | 1 \ Hijo2 \ r \ 1.5 \ \ | 0 \ Hijo3 \ rp \ 1.3 \ \ | 0 \ Hijo4 \ rp \ 1.5 \ \ | 3 \ Hijo5 \ r \ 7 \ \ | 3 \ Hijo6 \ r \ 8.2 \ \ | |

~X | Hijo1 | ler \ 010407 \ m \ 7 \ v \ 20 \ |
~X | Hijo2 | ler \ 010502 \ m \ 8 \ v \ 21 \ |
~X | Hijo3 | ler \ 010301 \ m \ 9 \ v \ 22 \ |
~X | Hijo4 | ler \ 010702 \ m \ 10 \ v \ 23 \ |
~X | Hijo5 | ler \ 010801 \ m \ 11 \ v \ 24 \ |
~X | Hijo6 | ler \ 010903 \ m \ 12 \ v \ 25 \ |

Clasificación de los residuos.

Un residuo, aparte de por LER, también se puede clasificar por otros sistemas de clasificación, como, por ejemplo:

- Tipo de residuo, según Directiva 1999/31/CE: inerte, no peligroso, peligroso.
- Fracciones mínimas, según Real Decreto 105/2008: hormigón, tejas y materiales cerámicos, madera, plástico, envases de papel y cartón, etc.

Una vez obtenido el LER del residuo, para obtener la clasificación según otras categorías utilizaremos los registros ~C y ~D. Para identificar a las categorías y a los conceptos incluidos en dichas categorías en el registro ~C se usará el TIPO de concepto: 5. Para cada clasificación este valor se totaliza transformado en m3 y kg.

Ejemplo:

Información previa:

Por una parte:
~X | | m \ masa \ kg \ v \ volumen \ m3 \ ler \ Código ler de la lista europea de residuos \ \ ce \ Coste Energético \ MJ \ GWP-total \ Emisión de CO2eq \ kg \ |

Por otra:
~C | TMA# | | Clasificaciones de Medio Ambiente | | | 0 |
~D | TMA# | ZZ# \ \ \ ZR# \ \ \ ZC# \ \ \ |

Donde:
~C | ZZ# | | Separación selectiva por códigos LER (Lista Europea de residuos) específicos | | | 0 |
~D | ZZ# | ZZ18971# \ \ 0 \ ZZ18974# \ \ 0 \ ZZ18975# \ \ 0 \ ZZ19003# \ \ 0 \ … Etc.|
Donde:
~C | ZZ18971# | | 150102 (envases de plástico) | | | 5 |
~C | ZZ18974# | | 170101 (hormigón) | | | 5 |
~C | ZZ18975# | | 170102 (ladrillos) | | | 5 |
~C | ZZ19003# | | 150103 (envases de madera) | | | 5 |
Etc.
~C | ZR# | | Separación selectiva según límites RD 105/2008 | | | 0 |
~D | ZR# | ZR19019# \ \ 0 \ ZR19020# \ \ 0 \ ZR20316# \ \ 0 \ ZR20318# \ \ 0 \ … Etc.|
Donde:
~C | ZR19019# | | 170201 (madera) | | | 5 |
~C | ZR19020# | | 170203 (plástico) | | | 5 |
~C | ZR20316# | | 170103 (tejas y materiales cerámicos) | | | 5 |
~C | ZR20318# | | 170101 (hormigón) | | | 5 |
Etc.
~C | ZC# | | Separación selectiva mínima por tipo de residuo según Directiva 1999/31/CE | | | 0 |
~D | ZC# | ZC19012# \ \ 0 \ ZC19013# \ \ 0 \ ZC19014# \ \ 0 \ |
Donde:
~C | ZC19012# | | inertes | | | 5 |
~C | ZC19013# | | peligrosos | | | 5 |
~C | ZC19014# | | no peligrosos | | | 5 |

Caso concreto: Elemento unitario P6143-AX1Q de la base de datos BEDEC Construcción del ITeC:

~C | P6143-AX1Q | m2 | Tabique apoyado divisorio 5cm ladrillo hueco sencillo, 240x115x50mm,LD,I UNE-EN 771-1,p/revestir,mort.albañilería,M10 | 0 | | EU |

~D | P6143-AX1Q | A0D-0007 \ 1 \ 0.285 \ A0E-000A \ 1 \ 0.071 \ A0F-000T \ 1 \ 0.57 \ B011-05ME \ 1 \ 0.0019 \ B0F13-0LM2 \ 1.02 \ 31.86 \ B07L-1PY2 \ 1 \ 0.014 \ C17A-00JM \ 1 \ 0.071 \ A%AUX001 \ 1 \ 0.025 \ |

~R | P6143-AX1Q | 0 \ B0F13-0LM2 \ rp \ 0.01960784 \ \ | |
…
~R | B0F13-0LM2 | 3 \ ZF1683246 \ r \ 0.0069 \ \ | 3 \ ZF1683248 \ r \ 0.00038 \ \ | |
~X | B0F13-0LM2 | m \ 1.9 \ v \ 0.00138 \ ler \ ZZ18975 \ ce \ 4.7292042864213 \ GWP-total \ 0.42835178848868 |
~X | B07L-1PY2 | m \ 1000 \ v \ 0.5128205128 \ ler \ ZZ18974 \ ce \ 830.72644326233 \ GWP-total \ 187.16315888945 \ |
…
Donde:
~C | ZF1683246 | m2 | film polietileno 25 micras | | | 4 |
~X | ZF1683246 | m \ 0.02275 \ v \ 0.000025 \ ler \ ZZ18971 \ |

~C | ZF1683248 | u | palet madera 120x120cm,27 kg Qd<=1000kg | | | 4 |
~X | ZF1683248 | m \ 27 \ v \ 0.144 \ ler \ ZZ19003 \ |
…
~D | ZZ18971 | ZR19020# \ \ \ ZC19014# \ \ \ |
~D | ZZ19003 | ZR19019# \ \ \ ZC19014# \ \ \ |
~D | ZZ18975 | ZC19012# \ \ \ ZR20316# \ \ \ |
~D | ZZ18974 | ZC19012# \ \ \ ZR20318# \ \ \ |
…

~T. REGISTRO TIPO TEXTO.

Este registro contiene el texto descriptivo de un concepto

~T | CODIGO_CONCEPTO | TEXTO_DESCRIPTIVO |

CODIGO_CONCEPTO: CODIGO del concepto descrito

TEXTO_DESCRIPTIVO: Texto descriptivo del concepto sin limitación de tamaño. El texto podrá contener caracteres fin de línea (ASCII-13 + ASCII-10) que se mantendrán al reformatearlo.

~P. REGISTRO TIPO DESCRIPCIÓN PARAMÉTRICA.

Este registro contiene la descripción paramétrica, bien en formato tradicional bien en formato API para DLL, que incluye la DEFINICION de parámetros, descomposiciones, comentario de ayuda a la selección de parámetros, resúmenes, textos, pliegos, claves e INFORMACION comercial, en función de tablas, expresiones y variables, para una familia de conceptos.

Este registro puede adoptar dos formas:

~P | | [DESCRIPCION_PARAMETRICA] | [NOMBRE.DLL] |

Cuando CODIGO_FAMILIA está lleno, o bien DESCRIPCION_PARAMETRICA está llena, o bien DESCRIPCION_PARAMETRICA está vacía. En este último caso se accede a la descripción paramétrica de la familia a través del archivo NOMBRE.DLL.

~P | CODIGO_FAMILIA | [DESCRIPCION_PARAMETRICA] |

Cuando CODIGO_FAMILIA está vacío, se refiere al paramétrico global.

Si DESCRIPCION_PARAMETRICA está llena, el paramétrico global se establece a partir de ésta. Si DESCRIPCION_PARAMETRICA está vacía y NOMBRE.DLL está lleno, se establece a partir de éste. Si DESCRIPCION_PARAMETRICA y NOMBRE.DLL están llenos a la vez, tan solo es válida DESCRIPCION_PARAMETRICA.

CODIGO_FAMILIA: CODIGO del concepto tipo familia descrito. Si se utiliza un modelo de codificación dependiente de los parámetros (ver Anexos 2 y 3), este código debe poseer un carácter '$' en su séptima posición, y los conceptos en los que se deriva tendrán como código los seis primeros caracteres de este más un carácter adicional por cada parámetro que posea.

DESCRIPCION_PARAMETRICA: Ver Anexo 2.

NOMBRE.DLL: Ver Anexo 3.

~L, ~Q, ~J. REGISTRO TIPO PLIEGOS.

Este registro contiene las diferentes secciones y textos del pliego de condiciones de un concepto. El pliego de condiciones se estructura de forma jerárquica con el Sistema de Clasificación por Codificación y de forma facetada en varias secciones de distinto contenido.

Secciones de los pliegos.

Cuando el primer campo del registro ~L está vacío, el registro define los CODIGOs de las SECCIONES de cada pliego y sus ROTULOs correspondientes. Este registro es único para una base de datos u obra.

~L | | < CODIGO_SECCION_PLIEGO \ [ROTULO_SECCION_PLIEGO] \ > |

CODIGO_SECCION_ PLIEGO: CODIGO que define cada SECCION o faceta del pliego.

ROTULO_SECCION_PLIEGO: DEFINICION del ROTULO asociado a cada CODIGO correspondiente de cada SECCION o faceta del pliego.

Ejemplo de las secciones de los pliegos definidas para la Base de Datos de CONSTRUCCION de la Comunidad de Madrid y la Base de Datos de CONSTRUCCION de la Comunidad Valenciana, indicando CODIGO y ROTULO de la SECCION:

~L | | DES \ DESCRIPCION Y COMPLEMENTOS AL TEXTO
\ PRE \ REQUISITOS PREVIOS A LA EJECUCIÓN
\ COM \ COMPONENTES
\ EJE \ EJECUCION Y ORGANIZACION
\ NOR \ NORMATIVA
\ CON \ CONTROL Y ACEPTACION
\ SEG \ SEGURIDAD E HIGIENE
\ VAL \ CRITERIOS DE VALORACION Y MEDICION
\ MAN \ MANTENIMIENTO
\ VAR \ VARIOS \ |

Modelo 1 de textos de los pliegos.

Cuando el primer campo del registro ~L no está vacío, identifica a un concepto determinado. Puede haber un registro de este tipo por cada concepto de una base de datos u obra.

~L | CODIGO_CONCEPTO | { CODIGO_SECCION_PLIEGO \ TEXTO_SECCION_PLIEGO \ } | { CODIGO_SECCION_PLIEGO \ ARCHIVO_TEXTO_RTF \ } | { CODIGO_SECCION_PLIEGO \ ARCHIVO_TEXTO_HTM \ } |

CODIGO_CONCEPTO: CODIGO del concepto descrito, contenido en la base de datos.

CODIGO_SECCION_PLIEGO: DEFINICION del CODIGO asociado a cada pliego.

TEXTO_SECCION_PLIEGO: Texto asignado a cada faceta o SECCION del pliego de condiciones del concepto.

El pliego de condiciones de cada concepto estará dividido con caracteres '\' en varias secciones o facetas, pensadas para imprimirse juntas o por separado.

Los fines de línea de cada SECCION del pliego se tratarán como en el REGISTRO TIPO TEXTO.

ARCHIVO_TEXTO_RTF: Es el nombre del archivo que contiene el texto en formato RTF asignado a cada SECCION del pliego del concepto. Dicho archivo deberá ubicarse en el mismo

directorio donde se hallen el/los archivos con extensión BC3 que incluyen su referencia.

ARCHIVO_TEXTO_HTM: Es el nombre del archivo que contiene el texto en formato HTM asignado a cada SECCION del pliego del concepto. Dicho archivo deberá ubicarse en el mismo directorio donde se hallen el/los archivos con extensión BC3 que incluyen su referencia.

Modelo 2 de textos de los pliegos.

Otra opción permite asignar el Pliego mediante párrafos de texto asociados a conceptos, utilizando el siguiente esquema de registros, como forma alternativa a la anterior:

~Q | < CODIGO_CONCEPTO \ > | { CODIGO_SECCION_PLIEGO \ CODIGO_PARRAFO \ { ABREV_AMBITO ; } \ } |

~J | CODIGO_PARRAFO | [TEXTO_PARRAFO] | | [ARCHIVO_PARRAFO_RTF] | [ARCHIVO_PARRAFO_HTM] |

CODIGO_CONCEPTO: CODIGO del concepto descrito, contenido en la base de datos. Será único para cada registro ~Q.
Este registro es de sustitución de la INFORMACION, no es de acumulación.

CODIGO_SECCION_PLIEGO: DEFINICION del CODIGO asociado a cada pliego. Corresponde al definido en el registro de cabecera de pliego ~L.

CODIGO_PARRAFO: CODIGO del texto asociado a cada sección del pliego.

ABREV_AMBITO: Identificador del ámbito geográfico de la sección del pliego. Se define en un registro propio.

TEXTO_PARRAFO: Texto que define el contenido de los pliegos que se asocian a un concepto y se identifica con CODIGO_PARRAFO.

TEXTO_PARRAFO_RTF: Texto que define el contenido de los pliegos que se asocian a un concepto y se identifica con CODIGO_PARRAFO, con formato RTF, de forma opcional, siendo siempre obligatorio el campo TEXTO_PARRAFO.

ARCHIVO_PARRAFO_RTF: Es el nombre del archivo en formato RTF que define el contenido de los pliegos que se asocian a un concepto y se identifica con CODIGO_PARRAFO. Dicho archivo deberá ubicarse en el mismo directorio donde se hallen el/los archivos con extensión BC3 que incluyen su referencia.

ARCHIVO_PARRAFO_HTM: Es el nombre del archivo en formato HTM que define el contenido de los pliegos que se asocian a un concepto y se identifica con CODIGO_PARRAFO. Dicho archivo deberá ubicarse en el mismo directorio donde se hallen el/los archivos con extensión BC3 que incluyen su referencia.

~W. REGISTRO TIPO ÁMBITO GEOGRÁFICO.

Establece el ámbito geográfico correspondiente a los Pliegos de Condiciones asociados a la Base de Datos. No necesariamente deberá corresponder al campo CABECERA definido en el registro ~V.

~W | < ABREV_AMBITO \ [AMBITO] \ > |

ABREV_AMBITO: Nombre abreviado que identifica el territorio geográfico al que se refiere. (Comunidad Autónoma, Provincia, Región, Comarca, Localidad, etc.). El identificador < * >

(ASCII - 42) indica AMBITO_GENERAL, y representa todo el territorio nacional.

AMBITO: Nombre completo del territorio geográfico.

Existe una relación de abreviaturas recomendadas, elaborada por la Asociación de Redactores de Bases de Datos de CONSTRUCCIÓN, que se puede consultar en el Anexo 5.

~G. REGISTRO TIPO INFORMACIÓN GRÁFICA.

Este registro contiene el/los archivos gráficos asociados a un concepto. Todos los archivos externos pueden ubicarse en el mismo directorio donde se hallen el/los archivos con extensión BC3 que incluyen su referencia o bien en la url que se indique.

~G | CODIGO_CONCEPTO | < ARCHIVO_GRAFICO.EXT \ > | [URL_EXT] |

CODIGO_CONCEPTO: CODIGO del concepto descrito en la base de datos y contenido en ella.

ARCHIVO_GRAFICO.EXT: Nombre del archivo que contiene la INFORMACION gráfica. Se usarán como referencia programas estandarizados de uso general, para chequear y verificar el contenido del fichero. Estos programas serán:

Ficheros tipo ráster:	Extensión .BMP, .PCX:	Windows 3.1
	Extensiones .GIF, .JPG, .PNG:	MS Internet Explorer 5.5
	Extensión .TIF:	Paint Shop Pro 4.0
Ficheros vectoriales:	Extensión .WMF:	Windows 3.1
	Extensión .DXF:	Autocad 12 Windows

URL_EXT: es un campo opcional. En el caso de no estar vacía, es la url a añadir a la URL_BASE para encontrar el gráfico. URL_BASE se define en el registro ~V. El comportamiento es: primero se busca el gráfico en el directorio local y si no está se busca en URL_BASE + URL_EXT + ARCHIVO_GRAFICO.EXT.

~E. REGISTRO TIPO ENTIDAD.

Define las entidades suministradoras de documentación técnica, tarifas de precios y especificaciones de los conceptos contenidos en la Base de Datos.

~E | CODIGO_ENTIDAD | [RESUMEN] | [NOMBRE] | { [TIPO] \ [SUBNOMBRE] \ [DIRECCIÓN] \ [CP] \ [LOCALIDAD] \ [PROVINCIA] \ [PAIS] \ { TELEFONO ; } \ { FAX ; } \ { PERSONA_CONTACTO ; } \ } | [CIF] \ [WEB] \ [EMAIL] \ |

CODIGO_ENTIDAD: CODIGO del SCc que define a la entidad (empresa, organismo, etc.).

RESUMEN: Nombre abreviado de la entidad

NOMBRE: Nombre completo de la entidad.

TIPO: Se definen los siguientes:

- C Central.
- D Delegación.
- R Representante.

SUBNOMBRE: Nombre de la delegación o representante en caso de que sea distinto de la central. Normalmente estará vacío.

DIRECCIÓN \ CP \ LOCALIDAD \ PROVINCIA \ PAIS: Dirección postal de la entidad con todos sus datos, existiendo una dirección por cada subcampo tipo, de forma ordenada y secuencial.

TELEFONO: Números de teléfono de la entidad, de forma ordenada y secuencial respecto al subcampo tipo, separados con el carácter < ; > (ASCII-59). Se indicará con nueve caracteres numéricos, incluido el prefijo de la provincia.

FAX: Números de fax de la entidad, con las mismas especificaciones que el campo anterior.

PERSONA_CONTACTO: Nombre de las personas de contacto con la entidad y cargo que desempeña, podrá haber varias asociadas a cada subcampo tipo, de forma que estén separadas por el carácter ASCII-59.

CIF: Código de Identificación Fiscal de la empresa.

WEB: Página web de la empresa.

MAIL: Dirección de correo electrónico de la empresa.

~O. REGISTRO TIPO RELACIÓN COMERCIAL.

Este registro establece los vínculos entre los conceptos de una Base de Datos General (BDG) con los productos comerciales de una Base de Datos Específica (BDE), y/o viceversa.

Así una Base de Datos (BD) podrá contener CONCEPTOS genéricos de una BDG, CONCEPTOS referentes a productos comerciales de una BDE, o ambas a la vez.

~O | CODIGO_RAIZ_BD # CODIGO_CONCEPTO | | < CODIGO_ARCHIVO \ CODIGO_ENTIDAD # CODIGO_CONCEPTO \ > |

CODIGO_RAIZ_BD # CODIGO_CONCEPTO: Identificador de un concepto de una BD, donde:

> CODIGO_RAIZ_BD: Se refiere a la identificación del CODIGO de la entidad que elabora la BD. Este CODIGO debe ser facilitado por la entidad que elabora la BD, para evitar ambigüedades. Se recomienda que éste sea el propio CIF de la entidad.
>
> CODIGO_CONCEPTO: Se refiere a un concepto que pertenece a CODIGO_RAIZ_BD, y empleado por ésta en su sistema de clasificación por codificación.

CODIGO_ARCHIVO: Se refiere al nombre del archivo que, de existir, indica el lugar donde se encuentra la INFORMACION referente a CODIGO_ENTIDAD # CODIGO_CONCEPTO. Sin embargo, si dicho CODIGO_ARCHIVO no existe, entonces indica que CODIGO_ENTIDAD # CODIGO_CONCEPTO se encuentra en la misma BD.

CODIGO_ENTIDAD # CODIGO_CONCEPTO: Identificador de un concepto de una BD, donde:

> CODIGO_ENTIDAD: Se refiere a la identificación del CODIGO de la entidad a la que se le asocia INFORMACION. Este CODIGO debe ser facilitado por la entidad que elabora la BD, de acuerdo con su sistema de clasificación, para evitar ambigüedades. Se recomienda que éste el propio CIF de la entidad.
>
> CODIGO_CONCEPTO: Se refiere a un concepto que pertenece a CODIGO_ENTIDAD, y empleado por la entidad que elabora la BD en su sistema de clasificación por codificación.

Cuando CODIGO_CONCEPTO se refiera a un producto comercial, dicho CODIGO deberá ser facilitado por el fabricante, y CODIGO_ENTIDAD#CODIGO_CONCEPTO no podrá coincidir nunca con la designación de CODIGO_RAIZ_BD, CODIGO_ENTIDAD o CODIGO_CONCEPTO, cuando éste se refiere a un concepto genérico. Al tener dicho producto comercial un tratamiento de CONCEPTO, éste puede utilizar todos los registros existentes en el formato para especificar su INFORMACION asociada (precio, INFORMACION gráfica, etc.). Para poder utilizar los registros mencionados, el código identificador del concepto será CODIGO_ENTIDAD#CODIGO_CONCEPTO.

~X. REGISTRO TIPO INFORMACIÓN TÉCNICA.

Este registro contiene la ESPECIFICACION de otros datos referentes al concepto como, por ejemplo, peso específico o nominal, características físicas, cuantías geométricas, propiedades físico-mecánicas, etc.

Estos datos podrían emplearse en otras utilidades, como el cálculo de los coeficientes de transmisión térmica, aislamiento acústico, etc.

El registro tipo INFORMACION Técnica puede adoptar dos formas:

Si el primer campo está vacío, éste sirve como diccionario de términos de INFORMACION Técnica a los cuales se les podrá asociar una descripción y una unidad de medida.

~X | | < CODIGO_IT \ DESCRIPCION_IT \ UM \ > |

Si el primer campo identifica a un concepto, la INFORMACION que se especificará a continuación serán la/las parejas de términos de INFORMACION técnica con sus respectivos valores.

~X | CODIGO_CONCEPTO | < CODIGO_IT \ VALOR_IT \ > |

CODIGO_IT: CODIGO de la INFORMACION Técnica descrita.
Se definen los siguientes:

ce	Coste energético (MJ) ce también se puede facilitar diferenciando energía primaria renovable y no renovable. Sus expresiones serían: PERT Uso total de energía primaria renovable (MJ) PENRT Uso total de energía primaria no renovable (MJ)
GWP-total	Emisión de CO_2 (kg). Como CODIGO_IT también se admite eCO2 por compatibilidad con versiones anteriores
ler	Código ler de la lista europea de residuos
m	Masa del elemento (kg)
v	Volumen (m3)
d	Densidad (kg/m3)

DESCRIPCION_IT: Texto descriptivo de la INFORMACION Técnica, sin limitación de tamaño.

UM: En el caso que los valores de la INFORMACION Técnica sean valores numéricos, se indicará su Unidad de Medida, de acuerdo con el Sistema Internacional de Unidades de Medida.

CODIGO_CONCEPTO: CODIGO del concepto descrito, contenido en la base de datos. Será único para cada registro ~X.

VALOR_IT: Valor alfabético o numérico de la INFORMACION Técnica.

Cálculo de coste energético y emisión de CO_2.

Para calcular el coste energético y emisión de CO_2 en conceptos descritos como discretos utilizaremos los registros ~R y ~X. Para conceptos descritos con descripciones paramétricas compiladas en DLL utilizaremos BdcNumProp y BdcPropValString.

El coste energético unitario de un elemento compuesto se obtiene del sumatorio del coste energético de los componentes de la justificación de precios (que se obtiene de rendimiento * coste energético unitario del componente). El coste energético de los elementos simples es un valor directo.

La emisión de CO_2 unitaria de un elemento compuesto se obtiene del sumatorio del coste energético de los componentes de la justificación de precios (que se obtiene de rendimiento * emisión de CO_2 unitaria del componente). La emisión de CO_2 de los elementos simples es un valor directo.

Para el cálculo del coste energético y de la emisión de CO_2 de un elemento compuesto tendríamos que aplicar el siguiente cálculo (Se usan todos los decimales. Los cálculos y los totales se realizan sin redondeos):

[ ].ce = suma [C].ce
[ ].eCO2 = suma [C].eCO2
donde,
C = componente
Coste energético y emisión de CO2 de un componente de la justificación de precios [C] en un elemento compuesto:
[ ].[C].ce = [C]. ce * [ ].[C].Cantidad
[ ].[C].eCO2 = [C]. eCO2 * [ ].[C].Cantidad
donde [ ].[C].Cantidad = [ ].[C].Rendimiento * [C].Factor

Ejemplo:

Para describir la UM del coste energético y de la emisión de CO2 pondríamos:
~X| | ce \ coste energético \ MJ \ eCO2 \ emisión de CO2 \ kg \ |
Para definir el 'ce' y 'eCO2' de los elementos simples pondríamos:
~X | B5221FM0 | ce \ 5.4 \ eCO2 \ 0.41 \ |

~M. REGISTRO TIPO MEDICIONES.

En este registro figuran las mediciones (cantidades), en que interviene un concepto de un presupuesto en la descomposición de otro de mayor rango.

En el intercambio de archivos de presupuestos, deberá figurar siempre este registro, exista o no desglose de mediciones.

Cuando se trate de intercambiar una relación de registros ~M que recogen un listado de mediciones no estructurado, no es necesario la disposición de un CODIGO raíz ni los registros ~D complementarios. El operador indicará en estos casos cual es el destino de la medición.

~M | [CODIGO_PADRE \ ] CODIGO_HIJO | { POSICION \ } | MEDICION_TOTAL | { TIPO \ COMENTARIO { # ID_BIM } \ UNIDADES \ LONGITUD \ LATITUD \ ALTURA \ } | [ETIQUETA] |

CODIGO_PADRE: CODIGO del concepto padre o concepto descompuesto del presupuesto.

CODIGO_HIJO: CODIGO del concepto hijo o concepto de la línea de descomposición.

Este campo es opcional en el caso de intercambiar mediciones no estructuradas, es decir, que no pertenecen a la estructura general y completa de un presupuesto.

POSICION: Posición del CONCEPTO_HIJO en la descomposición del CONCEPTO_PADRE, este dato permite identificar la medición cuando la descomposición del concepto padre incluye varios conceptos hijo con el mismo CODIGO, la numeración de las posiciones comenzará con el 1.

El campo POSICION deberá especificarse siempre en intercambio de presupuestos cuando éste sea completo y estructurado, e indicará el camino completo de la medición descrita en la estructura del archivo. Por ejemplo 3 \ 5 \ 2, indicará la medición correspondiente al capítulo 3 del archivo; subcapítulo 5 del capítulo 3; y partida 2 del subcapítulo 5. En mediciones no estructuradas este campo es opcional.

MEDICION_TOTAL: Debe coincidir con el rendimiento del registro tipo '~D' correspondiente. Incorpora el sumatorio del producto de unidades, longitud, latitud y altura o el resultado de expresiones de cada línea, al leer este registro se recalculará este valor.

TIPO: Indica el tipo de línea de medición de que se trate. Usualmente este subcampo estará vacío. Los tipos establecidos en esta VERSION son:

'1' Subtotal parcial: En esta línea aparecerá el subtotal de las líneas anteriores desde el último subtotal hasta la línea inmediatamente anterior a ésta.

'2' Subtotal acumulado: En esta línea aparecerá el subtotal de todas las líneas anteriores desde la primera hasta la línea inmediatamente anterior a ésta.

'3' Expresión: Indicará que en el subcampo COMENTARIO aparecerá una expresión algebraica a evaluar. Se podrán utilizar los operadores '(', ')', '+', '-', '*', '/' y '^'; las variables 'a', 'b', 'c' y 'd' (que tendrán por valor las cantidades introducidas en los subcampos UNIDADES, LONGITUD, LATITUD y ALTURA respectivamente); y la constante 'p' para el valor Pi=3.1415926. Esta expresión será válida hasta la siguiente línea de medición en la que se defina otra expresión. Solo se evalúa la expresión y no se multiplica por las unidades. Las expresiones fórmulas utilizan los criterios definidos en el anexo 2.

Cuando la medición sea tipo 1 o tipo 2, los subcampos UNIDADES, LONGITUD, LATITUD y ALTURA tienen que estar vacíos. Si hubiera datos en ellos, no se tendrán en cuenta.

COMENTARIO: Texto en la línea de medición. Podrá ser un comentario o una expresión algebraica.

ID_BIM: Es opcional, surge de la necesidad de transmitir el o los identificadores de los elementos constructivos en el modelo BIM hacia los programas de presupuestos para el intercambio bidireccional de información entre ambas plataformas. Dichos identificadores pueden aparecer en diferentes líneas de medición de otros conceptos del presupuesto ya que un mismo elemento constructivo puede tener asociadas más de un concepto. En caso de incluir dichos identificadores el número de ellos debe coincidir con el número expresado en el campo UNIDADES.

En los modelos BIM siempre existe un identificador único interno que garantiza la identificación de cualquier elemento. En el caso de los modelos en IFC sería de la forma (por ejemplo) "1TKCVeWv1CexJ9vtUdfF1l". En el caso de Revit u otros programas tienen identificadores similares. Estos identificadores internos pueden cambiar si se regenera el modelo (se realiza una nueva exportación a IFC), o de forma general si se modifica internamente la estructura de datos. De modo que en versiones más avanzadas de un mismo proyecto los GUID podrían haber cambiado.

Adicionalmente los elementos del proyecto BIM tienen un identificador o tag. El tag puede ser generado de forma manual o automática. Debe ser único. Es legible y tiene expresiones del tipo (por ejemplo) "V-1-25" (ventana primera planta número 25). El tag es una referencia

alfanumérica crítica, y en el caso del BIM puede ir asociado a un elemento del proyecto. En la descripción del estándar ya aparece la etiqueta al final del registro M.

UNIDADES, LONGITUD, LATITUD, ALTURA: Cuatro números reales con las mediciones. Si alguna magnitud no existe se dejará este campo vacío.

ETIQUETA: Es opcional, surge de la necesidad de transmitir un identificador de los conceptos (capítulos, subcapítulos o partidas). Este identificador lo imprimen, diversos programas, en los listados de mediciones o presupuesto de una Obra (por ejemplo, '2.10', 'A-27b', '001001'…); siendo único para cada concepto (capítulo, subcapítulo o partida) y, en general, diferente de la codificación de la base de datos empleada para confeccionar el presupuesto (El 'CODIGO_HIJO' muchas veces no aparece en los listados mencionados).

Para transmitir la etiqueta de un capítulo, o subcapítulo, del presupuesto, también se utilizará este registro. En ese caso, el campo MEDICION_TOTAL, habitualmente, será 1 y los campos TIPO, COMENTARIO, UNIDADES, LONGITUD, LATITUD y ALTURA no existirán:
~M | [CODIGO_PADRE \ ] CODIGO_HIJO | { POSICION \ } | 1 | | [ETIQUETA] |

~N. REGISTRO TIPO AÑADIR MEDICIONES.

Igual que el registro tipo ~M, pero añade las líneas de medición de este registro a las ya existentes en vez de sustituir toda la medición como hace en aquel.

~N | [CODIGO_PADRE \ ] CODIGO_HIJO | { POSICION \ } | MEDICION | { TIPO \ COMENTARIO { # ID_BIM } \ UNIDADES \ LONGITUD \ LATITUD \ ALTURA \ } | [ETIQUETA] |

~I. REGISTRO TIPO ARCHIVO BIM.

En los registros ~M y ~N existe el subcampo ID_BIM que permite indicar para cada línea de medición a que elemento concreto del modelo BIM se refiere. Lo que se tiene es el nombre del archivo en el que se encuentra el modelo BIM que normalmente será un archivo IFC.

Archivo BIM es el lugar en donde buscar los códigos de los elementos y que permita añadir una lista con varios nombres de archivos, puesto que una obra de construcción puede estar formada por varios modelos 3D vinculados (arquitectura, estructura, instalaciones, etc.).

~I | ARCHIVO_BIM.EXT { \ ARCHIVO_BIM.EXT } |

ARCHIVO_BIM.EXT: Es el nombre del archivo en el que se encuentra el modelo BIM que contiene, entre otras informaciones, los identificadores de los elementos constructivos referenciados en el campo ID_BIM de los registros ~M y/o ~N.

~A. REGISTRO TIPO CLAVES.

Este registro establece la relación entre CODIGOs y descriptores del tesauro, para permitir la búsqueda de conceptos mediante términos clave.

~A | CODIGO_CONCEPTO | < CLAVE_TESAURO \ > |

CODIGO_CONCEPTO: CODIGO del concepto descrito en la base de datos y contenido en ella.

CLAVE_TESAURO: Términos clave relacionados con el concepto. Los términos compuestos (hormigón armado, cartón-yeso, mortero mixto) se identificarán unidos mediante < _ > (ASCII - 95), (hormigón_armado, cartón_yeso, mortero_mixto...). No está permitido el empleo del espacio en blanco.

~B. REGISTRO TIPO CAMBIO DE CÓDIGO.

Con este registro se posibilita el cambio o anulación de los CODIGOs de los conceptos, única unidad de INFORMACION que no se podía modificar con los registros definidos anteriormente.

~B | CODIGO_CONCEPTO | CODIGO_NUEVO |

CODIGO_CONCEPTO: CODIGO del concepto a cambiar o anular. Debe existir y pertenece a un concepto contenido en la BD.

CODIGO_NUEVO: Nuevo CODIGO para CODIGO_CONCEPTO, no debe existir previamente. Si este campo está vacío, se entiende que hay que eliminar CODIGO_CONCEPTO.

~F. REGISTRO TIPO DOCUMENTO ADJUNTO.

Este registro permite asociar a un concepto archivos con diferentes tipos de información.

~F | CODIGO_CONCEPTO | { TIPO \ { ARCHIVO.EXT ; } \ [DESCRIPCION_ARCHIVO] \ } | [URL_EXT] |

donde:

CODIGO_CONCEPTO: Código del concepto descrito en la base de datos y contenido en ella al cual se le asocia una información en forma de archivo.

TIPO: Código del tipo de información que contiene el archivo.
Inicialmente se consideran los siguientes:

0 Otros.
1 Características técnicas y de fabricación.
2 Manual de colocación, uso y mantenimiento.
3 Certificado/s de elementos y sistemas.
4 Normativa y bibliografía.
5 Tarifa de precios.
6 Condiciones de venta.
7 Carta de colores.
8 Ámbito de aplicación y criterios selección.
9 Cálculo de elementos y sistemas.
10 Presentación, datos generales, objetivos, etc. de empresa.
11 Certificado/s de empresa.
12 Obras realizadas.
13 Imagen.

ARCHIVO.EXT: Nombre del archivo con extensión que contiene información de CODIGO_CONCEPTO.
Un archivo (principal) puede contener archivos vinculados, separados con el carácter < ; > (ASCII-59), siendo el principal el primero que se muestre. Todos los archivos (incluso los vinculados) pueden ubicarse en el mismo directorio donde se hallen el/los archivos con extensión BC3 que incluyen su referencia o bien en la url que se indique.
Además de las extensiones permitidas en los registros ~G, ~L y ~J, se añaden los siguientes

usando como referencia programas estandarizados de uso general, para chequear y verificar el contenido del fichero:

Extensión .PDF	Acrobat Reader v.5
Extensión .AVI	Media Player v.7
Extensión .PPT	Power Point v.10

DESCRIPCION_ARCHIVO: Breve descripción de la información contenida en el archivo.

URL_EXT: es un campo opcional. En el caso de no estar vacía, es la url a añadir a la URL_BASE para encontrar el documento. URL_BASE se define en el registro ~V. El comportamiento es: primero se busca el gráfico en el directorio local y si no está se busca en URL_BASE + URL_EXT + ARCHIVO.EXT.

FORMATO FIEBDC-3. RESUMEN.

~V | [PROPIEDAD_ARCHIVO] | VERSION_FORMATO [\ DDMMAAAA] | [PROGRAMA_EMISION] | [CABECERA] \ { ROTULO_IDENTIFICACION \ } | [JUEGO_CARACTERES] | [COMENTARIO] | [TIPO INFORMACIÓN] | [NÚMERO CERTIFICACIÓN] | [FECHA CERTIFICACIÓN] | [URL_BASE] |

~K | { DN \ DD \ DS \ DR \ DI \ DP \ DC \ DM \ DIVISA \ } | [[CI] \ [GG] \ [BI] \ [BAJA] \ [IVA]] | { DRC \ DC \ DFS \ DRS \ DUO \ DI \ DES \ DN \ DD \ DS \ DSP \ DEC \ DIVISA \ } | [n] |

~C | CODIGO { \ CODIGO } | [UNIDAD] | [RESUMEN] | { PRECIO \ } | { FECHA \ } | [TIPO] |

~D | CODIGO_PADRE | < CODIGO_HIJO \ [FACTOR] \ [RENDIMIENTO] \ > | < CODIGO_HIJO \ [FACTOR] \ [RENDIMIENTO] \ {CODIGO_PORCENTAJE ; } \ > |

~Y | CODIGO_PADRE | < CODIGO_HIJO \ [FACTOR] \ [RENDIMIENTO] \ > | < CODIGO_HIJO \ [FACTOR] \ [RENDIMIENTO] \ { CODIGO_PORCENTAJE ; } \ >|

~ R | CODIGO_PADRE | { TIPO_DESCOMPOSICION \ CODIGO_HIJO \ { PROPIEDAD \ VALOR \ [UM] \ } | } |

~T | CODIGO_CONCEPTO | TEXTO_DESCRIPTIVO |

~P | | [DESCRIPCION_PARAMETRICA] | [NOMBRE.DLL] |

~P | CODIGO_FAMILIA | [DESCRIPCION_PARAMETRICA] |

~L | | < CODIGO_SECCION_PLIEGO \ [ROTULO_SECCION_PLIEGO] \ > |

~L | CODIGO_CONCEPTO | { CODIGO_SECCION_PLIEGO \ TEXTO_SECCION_PLIEGO \ } | { CODIGO_SECCION_PLIEGO \ ARCHIVO_TEXTO_RTF \ } | { CODIGO_SECCION_PLIEGO \ ARCHIVO_TEXTO_HTM \ } |

~Q | < CODIGO_CONCEPTO \ > | { CODIGO_SECCION_PLIEGO \ CODIGO_PARRAFO \ { ABREV_AMBITO ; } \ } |

~J | CODIGO_PARRAFO | [TEXTO_PARRAFO] | | [ARCHIVO_PARRAFO_RTF] | [ARCHIVO_PARRAFO_HTM] |

~W | < ABREV_AMBITO \ [AMBITO] \ > |

~G | CODIGO_CONCEPTO | < ARCHIVO_GRAFICO.EXT \ > | [URL_EXT] |

~E | CODIGO_ENTIDAD | [RESUMEN] | [NOMBRE] | { [TIPO] \ [SUBNOMBRE] \ [DIRECCIÓN] \ [CP] \ [LOCALIDAD] \ [PROVINCIA] \ [PAIS] \ { TELEFONO ; } \ { FAX; } \ { PERSONA_CONTACTO ; } \ } | [CIF] \ [WEB] \ [EMAIL] \ |

~O | CODIGO_RAIZ_BD # CODIGO_CONCEPTO | | < CODIGO_ARCHIVO \ CODIGO_ENTIDAD # CODIGO_CONCEPTO \ > |

~X | | < CODIGO_IT \ DESCRIPCION_IT \ UM \ > |

~X | CODIGO_CONCEPTO | < CODIGO_IT \ VALOR_IT \ > |

~M | [CODIGO_PADRE \ ] CODIGO_HIJO | { POSICION \ } | MEDICION_TOTAL | { TIPO \ COMENTARIO { # ID_BIM } \ UNIDADES \ LONGITUD \ LATITUD \ ALTURA \ } | [ETIQUETA] |

~N | [CODIGO_PADRE \ ] CODIGO_HIJO | { POSICION \ } | MEDICION | { TIPO \ COMENTARIO { # ID_BIM } \ UNIDADES \ LONGITUD \ LATITUD \ ALTURA \ } | [ETIQUETA] |

~I | ARCHIVO.EXT { \ ARCHIVO.EXT } |

~A | CODIGO_CONCEPTO | < CLAVE_TESAURO \ > |

~B | CODIGO_CONCEPTO | CODIGO_NUEVO |

~F | CODIGO_CONCEPTO | { TIPO \ { ARCHIVO.EXT ; } \ [DESCRIPCION_ARCHIVO] \ } | [URL_EXT] |

Anexo 1. Cambios respecto a versiones anteriores: 3/2020, 3/2016, 3/2012, 3/2007, 3/2004, 3/2002, 3/98 y 3/95.

Resumen de los cambios de la versión 3/2024 con respecto a 3/2020:

- Especificación: Se añade el carácter < - > (ASCII-45) al juego de caracteres a emplear en los campos CODIGO.
- ~R. Registro tipo DESCOMPOSICIÓN DE RESIDUOS: Se añade un ejemplo en el subapartado Clasificación de residuos.
- ~X. Registro tipo INFORMACIÓN TÉCNICA: En el campo CODIGO_IT:
    - Las variables empiezan y acaban con \.
    - ce también se puede facilitar diferenciando PERT y PENRT.
    - eCO2 pasa a llamarse GWP-total, aunque también se admite eCO2.
    - Se añade la variable predefinida Densidad.
- Anexo 2. Descripción paramétrica Formato ESTANDAR:
    - En SENTENCIA DE RENDIMIENTO se añade el factor de residuo.
    - Se añade SENTENCIA INFORMACIÓN TÉCNICA.
    - En FUNCIONES – VALOR RETORNADO se añaden MAYUSC(a), MINUSC(a) y ORAC(a).
- Nuevo Anexo 10 sobre enriquecimiento de modelos BIM con datos del presupuesto en formato BC3.
- Nuevo Anexo 11 sobre API DE CONEXIÓN REMOTA de bases de datos para el formato FIEBDC.

Resumen de los cambios de la versión 3/2020 con respecto a 3/2016:

- Especificación: La información de la base de datos ya solamente se da en un único archivo. Por otra parte, se especifica como se pueden recoger varias certificaciones a la vez en un único archivo.
- Nuevo registro: ~I. Registro tipo ARCHIVO BIM como complemento a ~M y ~N.
- Anexo 6. Divisas: Se establecen de acuerdo con la ISO 4217.
- Nuevo Anexo 9 sobre Criterios para la asignación de referencias en IFC a bancos de precios en formato FIEBDC.

Cambios de la API estándar para descripciones paramétricas compiladas:

- Introducción: Recomendación tanto para programas de presupuestos como para bases de datos compiladas en DLL para que se creen versiones de 64 bits.
- Ejemplo: Se cambia el ejemplo que ahora pasa a estar preparado para compilarse con Microsoft Visual Estudio 2010 juego de caracteres multibyte como DLL de 32 y 64 bits.

Resumen de los cambios de la versión 3/2016 con respecto a 3/2012:

- Especificación: Se especifica el modo de transferencia de información desde bases de datos en Internet a las aplicaciones en ordenadores locales utilizando FIEBDC.
- ~K. Registro tipo COEFICIENTES: Se aclara que hacer cuando se reciben cantidades con más cifras decimales que las especificadas en el registro ~K de un archivo FIEBDC, y por otra parte que hacer en el caso de recibir un archivo FIEBDC sin el registro ~K.
- ~M. Registro tipo MEDICIONES y ~N Registro tipo AÑADIR MEDICIONES: Se especifica como transmitir el o los identificadores de los elementos constructivos en el modelo BIM hacia los programas de presupuestos para el intercambio bidireccional de información entre ambas plataformas.
- Anexo 2: Modificación de los conceptos paramétricos en el modelo de codificación dependiente de los parámetros, tenga DLL o no, para poder tener más de cuatro parámetros.
- Anexo 4: Se desdoblan los tipos de conceptos obtenidos de los índices y fórmulas

polinómicas de revisión de precios del BOE, entre los del RD 1359/2011 y los anteriores a dicho RD.
- Anexo 7: Actualización de la normativa que regula las unidades de medida.
- Anexo 8: Actualización de la definición de los tipos de presupuesto PEM y PEC, así como añadir el sistema de cálculo de las unidades de obra, de acuerdo con lo indicado en el Real Decreto 1098/2001 Reglamento General de la Ley de Contratos de las Administraciones Públicas.

Resumen de los cambios de la versión 3/2012 con respecto a 3/2007:

- ~V. Registro tipo VERSIÓN: Nuevo campo URL_BASE donde suministrar documentos y gráficos asociados a conceptos en una url.
- ~C. Registro tipo CONCEPTO. Ampliación de los TIPO de conceptos '4' y '5'.
- ~D. Registro tipo DESCOMPOSICIÓN. Nuevo campo 3.
- ~Y. Registro tipo AÑADIR DESCOMPOSICIÓN. Nuevo campo 3.
- Nuevo registro: ~R. Registro tipo DESCOMPOSICIÓN DE RESIDUOS.
- ~G. Registro tipo INFORMACIÓN GRÁFICA y ~F Registro tipo DOCUMENTO ADJUNTO. Nuevo campo URL_EXT para añadir a la URL_BASE donde suministrar documentos y gráficos asociados a conceptos. En el registro ~F se añade el documento TIPO '13'.
- ~O. Registro tipo INFORMACIÓN COMERCIAL. Aclaración del campo CODIGO_ENTIDAD # CODIGO_CONCEPTO.
- ~X. Registro tipo INFORMACIÓN TÉCNICA. Se especifican los tipos 'ce', 'eCO2', 'ler', 'm' y 'v' de CODIGO_IT y se adjunta el sistema de cálculo de coste energético y emisión de CO_2.
- Anexo 4: Se añaden los tipos de concepto 4 y 5.

Cambios de la API estándar para descripciones paramétricas compiladas:
- Nueva función BdcUsos.
- Nueva función BdcDesCodigoPorcentaje.
- Función BdcDocCodigo. Se añade que los archivos pueden contener una URL_EXT.
- Nueva función BdcNumProp.
- Nueva función BdcPropValString.
- Nueva función BdcNumComponentes.
- Nueva función BdcCodigoComponente.
- Nueva funcióm BdcNumPropComponente.
- Nueva función BdcComponenteValString.

Resumen de los cambios de la versión 3/2007 con respecto a 3/2004:

- Especificación: Aclaración sobre la prohibición de usar espacios en blanco en los códigos de los conceptos: "incluyendo A-Z, a-z, 0-9, ñ, Ñ," … "Excluyendo cualquier otro carácter como espacio, tabulador, etc".
- ~V. Registro tipo VERSIÓN: Nuevos campos TIPO INFORMACIÓN, NUMERO CERTIFICACIÓN y FECHA CERTIFICACIÓN.
- ~K. Registro tipo COEFICIENTES: Se suprimen los campos DRO Y DFO. Se añaden los campos DSP y DEC.
- ~C. Registro tipo CONCEPTO: Limitación del código a 20 caracteres. Recomendación de 64 caracteres como máximo para el resumen.
- ~D. Registro tipo DESCOMPOSICIÓN: Eliminar factor y rendimiento de los capítulos.
- ~M. Registro tipo MEDICIÓN: Aclaración sobre el uso de expresiones.
- Anexo 4: Aclaración sobre que el tipo Presupuesto Unitario no debe usar la # del tipo capítulo.

Resumen de los cambios de la versión 3/2004 con respecto a 3/2002:

- Especificación: Se incluye la notación <a> para campos opcionales.

- ~V. Registro tipo VERSIÓN: Nuevo campo COMENTARIO.
- ~K. Registro tipo COEFICIENTES: Se establece los decimales por defecto, modificaciones de algunos campos decimales.
- ~D. Registro tipo DESCOMPOSICION: Se establece el valor por defecto para el rendimiento.
- ~C. Registro tipo CONCEPTO: Modificaciones de los campos CÓDIGO y PRECIO.
- ~M. y ~N. Registro tipo MEDICION y AÑADIR MEDICION: Nuevo campo ETIQUETA.
- Registros actualizados con las notaciones para definir campos obligatorios y opcionales: ~V, ~D, ~Y, ~T, ~L, ~Q, ~W, ~G, ~O, ~X, ~A y ~B.
- Nuevo registro: ~F. Registro tipo DOCUMENTO ADJUNTO.

Cambios de la API estándar para descripciones paramétricas compiladas:
- Nueva función BdcTipoDescripcion().
- Nueva función BdcInvalidos().
- Aclaración de la función BdcPliego().
- Modificación de los parámetros de la función BdcComercCodigo().
- Nuevo mensaje de error BDCERR_PARÁMETRO_INCORRECTO.
- Nuevas funciones BdcDocNumero() y BdcDocCodigo() relacionadas con el registro ~F tipo DOCUMENTO ADJUNTO.

Clasificación en tipos de los conceptos:
- Nuevo tipo Presupuesto Unitario.

Lista de apartados y registros del formato afectados por ampliaciones y/o modificaciones:

Apartado	3/2024	3/2000	3/2016	3/2012	3/2007	3/2004	3/2002	3/98
PRESENTACION.	X	X	X	X	X	X	X	X
FORMATO FIEBDC-3. ESPECIFICACION.	X	X	X		X	X	X	X
~V. Registro tipo PROPIEDAD Y VERSION.				X	X	X	X	X
~K. Registro tipo COEFICIENTES.			X		X	X	X	X
~C. Registro tipo CONCEPTO.				X	X	X	X	X
~D. Registro tipo DESCOMPOSICION.				X	X	X	X	X
~Y. Registro tipo AÑADIR DESCOMPOSICION.				X		X	X	X
~R. Registro tipo DESCOMPOSICIÓN de RESIDUOS.	X			X				
~T. Registro tipo TEXTO.						X	X	X
~P. Registro tipo DESCRIPCIÓN PARAMETRICA.								X
~L. Registro tipo PLIEGOS.						X	X	X
~Q. Registro tipo PLIEGOS.						X	X	X
~W. Registro tipo AMBITO GEOGRAFICO.						X	X	X
~G. Registro tipo INFORMACION GRAFICA.				X		X	X	X
~E. Registro tipo ENTIDAD.							X	X
~O. Registro tipo RELACION COMERCIAL.				X		X	X	X
~X. Registro tipo INFORMACION TECNICA.	X			X		X	X	X
~M. Registro tipo MEDICIONES.			X		X	X	X	X
~N. Registro tipo AÑADIR MEDICIONES.			X			X	X	X
~I. Registro Tipo ARCHIVO BIM		X						
~A. Registro tipo CLAVES.						X	X	X
~B. Registro tipo CAMBIO DE CODIGO.						X	X	X
~F. Registro Tipo DOCUMENTO ADJUNTO.				X		X	X	X
Anexo 2. Descripción paramétrica: Formato ESTANDAR.	X		X					
Anexo 3. Descripción paramétrica: API ESTANDAR para descripciones paramétricas compiladas en DLL.		X		X		X	X	X
Anexo 4. Clasificación en tipos de los Conceptos.			X	X	X	X	X	X
Anexo 6. Divisas		X						

Apartado	3/2024	3/2000	3/2016	3/2012	3/2007	3/2004	3/2002	3/98
Anexo 7. Unidades de medida.			X					
Anexo 8. Definiciones de diferentes tipos de Presupuestos.			X					
Anexo 9. Criterios para la asignación de referencias en IFC a bancos de precios en formato FIEBDC.		X						
Anexo 10. Enriquecimiento de modelos BIM con datos del presupuesto en formato BC3.	X							
Anexo 11. API DE CONEXIÓN REMOTA de bases de datos para el formato FIEBDC.	X							

Relación de las versiones del formato con las Actas de la Asamblea (y las actas de la CTP):

	3/2024	3/2000	3/2016	3/2012	3/2007	3/2004	3/2002	3/98
	2021-06-18 (28)	2016-05-27 (23)	2012-05-10	2009-04-23	2005-04-14 (13)	2002-05-10	1998-06-09	1997-04-08 (1,2,3,4,5,6)
	2022-06-17 (29)	2017-06-01 (24)	2013-05-23 (19)	2010-05-13 (17)	2006-05-11 (14)	2002-11-07	1998-09-21	
	2023-06-15 (30)	2018-06-12 (25)	2014-05-08 (20)	2011-05-20 (18)	2007-05-18 (15)	2003-05-29 (10,11)	1999-04-14 (7)	
		2019-06-28 (26)	2015-05-21 (21,22)		2008-05-23 (16)	2004-05-13 (12)	2000-05-19	
		2020-10-15 (27)					2000-11-10	
							2001-04-23 (7,8)	
							2001-11-08 (9)	

Anexo 2. DESCRIPCIÓN PARAMÉTRICA: Formato ESTÁNDAR.

Un concepto paramétrico es el que define su CODIGO, resumen, texto, pliego, descomposición e INFORMACION comercial de forma paramétrica, esto es, de una forma variable mediante tablas y expresiones aritméticas y lógicas función de parámetros.

En la descripción paramétrica se encuentran las siguientes sentencias:

Se definen las variables:
%A %B %C %D %F %G %H %I %J %K: Parámetros seleccionados del concepto de "a" a "z" ~ 1 a 26
%O %P %Q %R %S %T %U %V %W %X: Parámetros seleccionados de la obra de "a" a "z" ~ 1 a 26

%E:	Variable que define las condiciones de error.
$A $B $C $D $F $G $H $I $J $K:	Textos de los parámetros seleccionados del concepto.
$O $P $Q $R $S $T $U $V $W $X:	Textos de los parámetros seleccionados de la obra.
$E:	Variable que define los textos de error.

De forma equivalente las variables %O a %X y $O a $X tomarían el valor correspondiente a los valores de los parámetros generales de la obra.

Cualquier variable de la 'A' a la 'Z' tanto numérica (%) como alfanumérica ($) se puede definir o redefinir con cualquier número de dimensiones para ser utilizada posteriormente en expresiones.

Se definen las constantes de la 'a' a la 'z' con los valores numéricos del 1 al 26 respectivamente, para permitir referenciar los parámetros de forma nemotécnica. Para la utilización de otro tipo de caracteres, se determinará en el texto de la opción del parámetro seleccionado el carácter de sustitución que se desea utilizar, anteponiéndole un carácter especial ' ! ' .Si dicho carácter no existe la sustitución se realiza relacionando el carácter con la posición que ocupa.

Ejemplo: PBPO.2$ M3 Hormigón $B $A
\ CONSISTENCIA \ plástica \ fluida \ blanda \
\ RESISTENCIA \ H-125 \ H-150 \ H-175 \ H-200 \
El derivado PBPO.2aa sería: M3 Hormigón H-125 plástica

Con el carácter especial:
\ CONSISTENCIA \ !p plástica \ !f fluida \ !b blanda \
\ RESISTENCIA \ !2 H-125 \ !5 H-150 \ !7 H-175 \ !0 H-200 \
El mismo derivado sería: PBPO.2p2 M3 Hormigón H-125 plástica.

Las variables numéricas deben permitir valores reales en coma flotante de doble precisión (64bits) y las variables alfanuméricas deben poder almacenar textos de cualquier tamaño.

Cualquier variable puede definirse, en la misma asignación, con cualquier número y tamaño de dimensiones (hasta 4), en la DEFINICION de dimensiones tendrán que hacerse explícitas todas las dimensiones.

```
%U =.....      # define una variable con un dato numérico
$X(8) =.....   # define una lista de 8 datos alfanuméricos
%V(3,4) =..... # define una tabla con 3 filas y 4 columnas de datos n.
```

Las variables %E y $E son especiales para devolver errores producidos por selecciones de parámetros no coherentes. En una evaluación secuencial de expresiones, si en una expresión la variable %E adopta un valor distinto de 0, ha habido algún error, se interrumpe la evaluación de expresiones y se presenta el contenido de la variable $E donde se almacena el texto del error producido.
Puede haber múltiples asignaciones de %E, cada una de ellas precedida de su correspondiente texto de error, asignación de $E.

Las constantes alfanuméricas se definirán entre comillas ($I="incluida parte proporcional").

En la descripción paramétrica podemos encontrar los siguientes tipos de sentencias:

SENTENCIA DE ROTULOS DE PARAMETRO:
\ <ROTULO del parámetro> \ { <ROTULO de la opción> \ }
Los parámetros definidos, hasta 10, se irán asignando a las variables ABCDFGHIJK en el orden que se encuentren.

SENTENCIA DE ASIGNACION NUMERICA:
<variable numerica> = <expresión numérica>

SENTENCIA DE ASIGNACION ALFANUMERICA:
<variable alfanumerica> = <expresión alfanumérica>

SENTENCIA DE RENDIMIENTO (CONCEPTOS DESCOMPUESTOS):
<texto de sustitución de CODIGO> : <expresión numérica> [: <exp.num.>] [; <factor de residuo>] Se pueden definir uno u opcionalmente dos rendimientos, el defecto del rendimiento opcional es 1.
También es opcional el factor de residuo, con separador previo punto y coma (;). Su valor por defecto es también 1.

SENTENCIA DE MEDIOS AUXILIARES:
%: <expresión numérica> (en tanto por cien)
%%: <expresión numérica> (en tanto por uno)

SENTENCIA DE PRECIO (CONCEPTOS SIMPLES): <expresión numérica>
En caso de figurar conjuntamente un juego de sentencias de rendimiento a modo de descomposición y una sentencia de precio, tendrá prioridad la sentencia de precio, ignorando las sentencias de rendimiento.

SENTENCIA DE COMENTARIO:
\ COMENTARIO \ ó \ C \ <texto del comentario> \
Si existe texto de comentario, se presentará como ayuda a la selección de parámetros junto a las opciones de éstos.

SENTENCIA DE SUSTITUCION:
\ RESUMEN \ ó \ R \ <texto de sustitución del texto resumido> \
\ TEXTO \ ó \ T \ <texto de sustitución del texto descriptivo> \
\ PLIEGO \ ó \ P \ { <texto de sustitución de pliego> \ }
\ CLAVES \ ó \ K \ { <texto de sustitución de clave> \ }
\ COMERCIAL \ ó \ F \ { <texto de sustitución de CODIGO>\ <expresión_numérica>\}

Se considera que una sentencia contínua en la línea siguiente si:
- Acaba en un operador
- Acaba sin cerrar comillas ""
- Comienza con '\' y no acaba con '\'

<constantes> PI, números, "texto" ...

<funciones> ABS( ), INT( ), SQRT( ) ...

<variables> [$%] [A-Z] [(dimensión{,dimensión})]

<expresión numérica>:
Son aquellas que dan como resultado un número en función de constantes y variables numéricas, expresiones lógicas, funciones y operadores.
por ejemplo: %I= %A + 3.17*(1+%B) + ABS(%P+3.15*%Q)/12000

<expresión alfanumérica>:
Son aquellas que dan como resultada un texto en función de constantes y variables alfanuméricas, operadores y funciones numéricas.
Una expresión alfanumérica puede incluir expresiones lógicas.
por ejemplo: $I="parte proporcional"+" de pérdidas"*(%A>a)
añadir " de pérdidas" a $I si el valor actual de %A es mayor que <a> ó 1.

<expresiones lógicas>:
Son aquellas que dan como resultado VERDADERO o FALSO. En expresiones numéricas el verdadero se considera como 1 y el falso como 0, en alfanuméricas el falso se considera suprimir texto.
%I = 323*(%A=a) + 345*(%A=b) + 1523*($I=$A & $J=$B)
$I = "blanco"*(%C=c) + "negro"*(%C=d)

<texto de sustitución>:
En los textos de sustitución la INFORMACION es un texto constante (sin comillas) con variables embebidas en él. Se consideran variables los caracteres $ y % inmediatamente seguidos por una letra de la A a la Z.
En los textos de sustitución, las variables alfanuméricas se sustituyen por sus contenidos de texto correspondiente, las numéricas se sustituyen por las constantes de la "a" a la "z" correspondientes al valor numérico de su contenido.
En la expresión del rendimiento, la primera parte de la sentencia es un texto de sustitución que una vez sustituidas las variables será el CODIGO del concepto al que le corresponde la expresión numérica de la segunda parte de la expresión como rendimiento. Si el resultado es 0, se ignora la sentencia y no se considera ese componente o línea de descomposición.

SENTENCIA INFORMACIÓN TÉCNICA

Se introducen las variables predefinidas necesarias para la definición de la información técnica, que posteriormente se trasladará a la definición del concepto generado a través de los registros ~R y ~X.

La sentencia comenzará por \INFOTEC y continuará, en la misma línea, con las variables predefinidas que se necesiten, de entre éstas:

\ce\	Coste energético
\PERT\	Información uso total de energía primaria renovable
\PENRT\	Información uso total de energía primaria no renovable
\GWP-total\	Información GWP-total (también puede usarse \eCO2\)
\ler\	Información de residuos generados
\m\	Masa
\v\	Volumen
\d\	Densidad

Por ejemplo, del siguiente modo:

\INFOTEC\v\70\m\%L(%C)\GWP-total\%M(%D)\ler\$F(%A)\

Ejemplificación del proceso:

En los conceptos paramétricos simples (sin descomposición)

La información técnica debe quedar incluida dentro de los conceptos paramétricos simples, y su operatividad debe ser similar a la de los Precios, generando sumatorios totales por partidas utilizando los rendimientos asignados a cada unidad de obra.

Por ejemplo:
Tenemos que realizar un concepto paramétrico simple para ladrillos macizos de 20 cm de largo y 10 cm de anchura y diferentes espesores:

~C | PBPLMC$ | u | Ladrillo cerámico macizo de tejar 20x10 cm | 0 | 02062021 | 3 |
~P | PBPLMC$ |
\ESPESOR (cm)\3\4\5\6\
%M= ATOF($A)
#Matriz definición de precios
%N(4)= 1,2,3,4
#Definición de volumen en m3
%L=0.20*0.10*%M
#Definición de masa en kg (densidad de cerámica de tejar =1800 kg/m3)
%K=%L*1800
#Definición de emisiones de CO2 en kg (eCO2 equivalente A1-A3 =2.63 kgCO2eq/kg)
%J=%K*2.63
#Definición de coste energético en MJ (ce Equivalente = 12 MJ/m3)
%I=%L*12
#Definición de la cantidad de residuos generados (1.23 kg de plástico/ m3)
%H=%L*1.23
#SENTENCIA DE PRECIO
::%N(%A)
#SENTENCIAS DE SUSTITUCIÓN
\RESUMEN\Ladrillo cerámico macizo de tejar 20x10x$A cm\
#SENTENCIA INFORMACIÓN TÉCNICA
\INFOTEC\ce\%I\GWP-total\%J\ler\%H\m\%K\v\%L\

Esta sentencia de información técnica incluye el coste energético (ce), la información eCO2 (GWP-total), la información de residuos generados (ler), la masa (m) y el volumen (v).

De esta forma, la información técnica de cada concepto generado se trasladaría en el registro ~X de la derivada paramétrica seleccionada.

La información relativa a los residuos generados debería trasladarse en los registros ~R y ~X de la derivada paramétrica seleccionada.

En este caso concreto una de las derivadas tendría la siguiente expresión en el estándar BC3:
~X | | ce \ coste energético \ MJ \ GWP-total \ emisión de CO2 \ kg \ ler \ Código LER \ \ m \ masa \ kg \ v \ volumen \ m3 \ |
[…]
~C | PBPLMCa | u | Ladrillo cerámico macizo de tejar 20x10x3 cm | 1.00 | 02062021 | 3 |
~X | PBPLMCa | ce \ 0.072 \ GWP-total \ 2.8404 \ ler \ 1.3284 \ m \ 1.08 \ v \ 0.000738 \ |

Los datos numéricos de precio e información técnica resultarían de las operaciones lógicas o matrices empleadas en la descripción paramétrica de cada concepto, quedando de este modo incorporadas en el concepto simple.

Se debe poner especial atención a la formulación del registro ~R en las definiciones paramétricas, ya que un único material puede generar diferentes tipos y cantidades de residuos.

<u>En los conceptos paramétricos con descomposición</u>

A la hora de generar conceptos paramétricos con descomposición (sin sentencia directa de precio), la información técnica de los registros ~X y ~R debe ser el sumatorio de todos los datos técnicos unitarios multiplicados por el rendimiento asignado a cada elemento de la descomposición (al igual que se hace con los precios).

También debería existir la posibilidad de introducir directamente una “Sentencia de Información Técnica” cuyo dato prevalezca sobre el obtenido del sumatorio de la descomposición (al igual que se hace con los precios).

CONVENIOS DE NOTACION (EBNF):

[a]	Indica nada o "a"
{a}	Indica cero o más ocurrencias de "a"
[a-b]	Indica cualquier valor desde "a" a "b" inclusivas
[abc]	Indica cualquiera de los valores a, b ó c
<abc>	Indica descripción informal
abc	Indica símbolo terminal
%[A-Z]	Variable numérica
$[A-Z]	Variable alfanumérica

Variables predefinidas:

[%$][ABCDFGHIJKLM]	Parámetros del concepto
[%$][OPQRSTUVWX]	Parámetros de la obra
[%$]E	Variable especial para reportar errores
[%$][A-Z][(dim{,dim})]	Variables definibles
#	Comentarios (el texto comprendido entre este carácter y el siguiente final de línea exclusive, no se tiene en cuenta)
,	Separador de datos
:	DEFINICION de rendimiento
::	DEFINICION de precio
%:	DEFINICION de medios auxiliares en tanto por cien
%%:	DEFINICION de medios auxiliares en tanto por uno

OPERADORES NUMERICOS (De menor a mayor precedencia):

+	Suma
-	Resta
*	Multiplicación
/	División
^	Operador elevado a

OPERADORES LOGICOS (De menor a mayor precedencia):

@	Operador lógico O
&	Operador lógico Y
<	Menor
>	Mayor
<=	Menor o igual
>=	Mayor o igual
=	Igual
<>	Diferente
!	Operador lógico NO

FUNCIONES -- VALOR RETORNADO:

ABS(n)	Valor absoluto de "n"
INT(n)	Parte entera de "n"
ROUND(n,d)	Redondeo de "n" a "d" decimales
SIN(n)	Seno (grados sexagesimales)
COS(n)	Coseno (grados sexagesimales)
TAN(n)	Tangente (grados sexagesimales)
ASIN(n)	Arco seno (gs)
ACOS(n)	Arco coseno (gs)
ATAN(n)	Arco tangente (gs)
ATAN2(x,y)	Arco tangente con dos parámetros "x" e "y"
SQRT(n)	Raíz cuadrada de "n"

ATOF(a)	Conversión de alfanumérico "a" a numérico
FTOA(n)	Conversión de numérico "n" a alfanumérico
MAYUSC(a)	Conversión a letras mayúsculas
MINUSC(a)	Conversión a letras minúsculas
ORAC(a)	Conversión de texto a tipo oración (primera letra de la primera palabra en mayúsculas y el resto en minúsculas)

Las funciones MAYUSC(a), MINUSC(a) y ORAC(a) no tienen compatibilidad para versiones anteriores del formato FIEBDC-3/2024.

Cada instrucción irá en distinta línea, a menos que la instrucción acabe en un operador en cuyo caso se considera que sigue en la siguiente línea.
Si una línea acaba sin haber cerrado las comillas "" o delimitador '\', se considerará que sigue en la línea siguiente. Los caracteres fin de línea (ASCII-13 + ASCII-10) contenidos en las descripciones paramétricas se mantendrán al reformatear.

CONTROL DE ERRORES DE SELECCION.

Es frecuente encontrar un gran número de combinaciones de parámetros posibles, pero tener pocas de ellas resueltas. Para evitar que el operador del sistema pruebe distintas combinaciones de parámetros consiguiendo en todas ellas un mensaje de error, el sistema debe ser capaz de guiarle en la selección de combinaciones correctas.
Cada vez que el operador define un parámetro, el sistema evaluará todas las sentencias posibles y en las sentencias del tipo: %E= ..., función de parámetros.

Si todos los parámetros intervinientes en la expresión son conocidos, se evaluará ésta y si el resultado fuera de ERROR se presentará la previa DEFINICION de $E con el mensaje del error.

Si todos los parámetros menos uno, son conocidos, se irá dando valores al parámetro desconocido y evaluando la expresión hasta recorrer todos los valores válidos del parámetro. De alguna forma, el sistema "marcará" los valores que producen ERROR del parámetro estudiado en la pantalla de selección, para ayudar al operador a seleccionar las combinaciones correctas.

Cada vez que se defina o redefina un parámetro el sistema actualizará todos los valores marcados en pantalla, por ejemplo, pondrá en "medio brillo" los ROTULOs de las opciones cuya selección no sería compatible con los parámetros seleccionados previamente.

Este sistema de control de errores de selección es sencillo de implementar en cualquier soporte, pero obliga a los redactores de los descompuestos paramétricos a definir explícitamente las combinaciones de parámetros incorrectas, ya que con este método no se podrían encontrar combinaciones no permitidas cuando en la descomposición paramétrica se llama a otros descompuestos o precios paramétricos.

PROCEDIMIENTO DE LECTURA DE DESCRIPCIONES PARAMETRICAS.

Recorrer la descripción paramétrica ejecutando los siguientes pasos:

1. Eliminar desde el carácter '#' inclusive hasta el siguiente cambio de línea exclusive.

2. Cambiar tabuladores (9) por caracteres ' ' (32)

3. Eliminar caracteres ' ' (32) delante y detrás de los caracteres '\'

4. Unir líneas, eliminando el fin de línea, en líneas que comienzan con '\' y no acaban con '\', que terminan con un operador y en la separación de datos de una variable matricial.

5. Eliminar todos los caracteres ' ' (32) en zonas no entrecomilladas ("...") o delimitadas (\...\)

6. Eliminar líneas vacías.

7. Leer y evaluar secuencialmente las sentencias de la forma:

   Si la sentencia comienza con '\' leer el ROTULO hasta el siguiente '\', si el ROTULO es:
   COMENTARIO ó C- Palabra o carácter reservado que identifica el siguiente ROTULO entre '\' como comentario a la selección de parámetros.
   RESUMEN ó R- Palabra o carácter reservado que identifica el siguiente ROTULO entre '\' como el texto de sustitución del resumen del concepto.
   TEXTO ó T- Palabra o carácter reservado que identifica el siguiente ROTULO entre '\' como el texto de sustitución del texto descriptivo del concepto.
   PLIEGO ó P- Palabra o carácter reservado que identifica los siguientes ROTULOs entre '\' como los textos de sustitución de las distintas secciones del pliego.
   CLAVES ó K- Palabra o carácter reservado que identifica los siguientes ROTULOs entre '\' como los textos de sustitución de los términos claves asociados al concepto.
   COMERCIAL ó F - Palabra o carácter reservado que identifica los siguientes ROTULOs entre '\' como los textos de sustitución y tarifas de la INFORMACION comercial del concepto.
   Cualquier otro ROTULO identificará el nombre del siguiente parámetro y los siguientes ROTULOs entre '\' como los ROTULOs de las opciones de dicho parámetro.
   Si la sentencia comienza con '::' el resto de esta debe ser una expresión numérica indicadora del Precio, sólo en familias de conceptos simples (sin descomposición) y sólo puede haber una sentencia de este tipo.

   Si la sentencia comienza con '%:' el resto de esta debe ser una expresión numérica indicadora del Porcentaje de Medios Auxiliares, sólo puede haber una sentencia de este tipo.

   En otro caso si la sentencia contiene el carácter ':' la parte anterior a él es un texto de sustitución del CODIGO de una línea de descomposición y la posterior una expresión numérica, o dos separadas por ':', indicadoras de el o los rendimientos de dicha línea de descomposición.

   En aquellos casos donde pueda aparecer un carácter '%' seguido de un carácter alfabético que se considere como tal y no como una variable de sustitución, deberá emplearse '%%', para evitar la ambigüedad que se puede producir entre una variable numérica que deba ser sustituida, una sentencia de medio auxiliar o un texto.

   El resto de las sentencias deberán ser de asignación de la forma variable/s = expresión/es

RESUMEN DE TIPOS DE SENTENCIAS.

Después de realizado el filtro descrito arriba, cada línea, tira de caracteres acabada en (ASCII-13) (ASCII-10), será una sentencia de alguno de los siguientes tipos:

```
{ \ ROTULO_parámetro \ { opción_parámetro \ } (13)(10) }
{ variable = expresión (13)(10) }
{ CODIGO : rendimiento [ : rendimiento ] (13)(10) }
[ %: ó %%: medios_auxiliares (13)(10) ]  % (tanto por cien) %% (tanto por uno)
[ :: expresión_precio (13)(10) ]
[ \ COMENTARIO \ ó \ C \ texto_comentario \ (13)(10) ]
[ \ RESUMEN \ ó \ R \ texto_resumen \ (13)(10) ]
[ \ TEXTO \ ó \ T \ texto_descriptivo \ (13)(10) ]
[ \ PLIEGO \ ó \ P \ { texto_faceta_pliego \ }(13)(10)  ]
[ \ CLAVES \ ó \ K \ { término_clave \ }(13)(10)  ]
[ \ COMERCIAL \ ó \ F \ { CODIGO_producto_comercial \ tarifa \ } (13)(10) ]
```

Anexo 3. DESCRIPCIÓN PARAMÉTRICA: API ESTÁNDAR para descripciones paramétricas compiladas en DLL

INTRODUCCION.

Debido a la necesidad expuesta por los desarrolladores de bases de datos paramétricas de ampliar las posibilidades del lenguaje de descripción paramétrica, poder compilar éste por eficiencia y protección de datos y posibilitar la protección contra copia de bases de datos paramétricas, se establece la siguiente ESPECIFICACION.

En este documento se definen los componentes necesarios para el desarrollo de descripciones paramétricas en cualquier lenguaje de aplicaciones para Windows (C, C++, Pascal, Fortran, etc.) y sin ninguna limitación. Se incluye la DEFINICION de un API ESTANDAR en C, un ejemplo de base de datos en formato DLL de 32 bits desarrollado en C++ y un ejemplo de aplicación con la implementación del interfaz con el API en C, definidos ambos en Microsoft Visual C++. Se podría implementar el interfaz con el API para otros compiladores y lenguajes para acceder a las mismas DLL.

Es decir; es posible construir una base de datos que cumpla este API utilizando para ello cualquier lenguaje de programación que permita desarrollar librerías de enlace dinámico Windows (DLL). Asimismo, es posible construir un programa que lea cualquier base de datos de estas características utilizando lenguajes de aplicaciones para Windows.

El juego de caracteres utilizado en los textos devueltos por las funciones del API será el especificado en el registro ~V.

Un archivo de un presupuesto con la extensión BC3 no debe llevar información de la DLL para los paramétricos.

Se recomienda que tanto los desarrolladores de programas de presupuestos como los redactores de bases de datos con descripciones paramétricas compiladas en DLL añadan versiones de 64 bits y que mantengan las versiones de 32 bits hasta que ya sean de uso consumado las de 64 bits en cuyo momento deberían dejar de usarse las de 32 bits.

El número de decimales de cada campo numérico devuelto por las funciones del API será el especificado en el registro ~K.

ARCHIVOS QUE DEBE CONTENER UNA BASE DE DATOS.

Una base de datos que se desee distribuir con las definiciones paramétricas compiladas en DLL, debe contener los siguientes archivos:

base.dll — En este archivo, único para cada base de datos y de nombre cualquiera, pero extensión '.DLL', se encuentran las funciones del API que la base de datos ofrece a las aplicaciones para que éstas obtengan la INFORMACION que contiene la base.

base.bc3 — Archivo o archivos ASCII de la base de datos en formato FIEBDC-3. Los registros ~P de los conceptos cuya descripción paramétrica se acceda a través del archivo 'base.dll', tendrán el campo DESCRIPCIÓN_PARAMÉTRICA vacío. El CODIGO del concepto de este registro deberá coincidir con el CODIGO del registro ~C correspondiente y con el CODIGO utilizado en las llamadas a las funciones del API, incluida(s) la(s) posible(s) almohadillas ('#'). Ejemplo: ~P|ABCD12$| |
El registro ~P correspondiente al paramétrico global, tendrá el campo DESCRIPCIÓN_PARAMÉTRICA vacío, y tendrá un tercer campo con el nombre del archivo DLL en el que se encuentren las funciones del API de la base. Ejemplo: ~P| | | BASE.DLL |

La DEFINICION paramétrica de los conceptos implementados de esta forma podrá estar en el mismo archivo que las funciones del API (el archivo ‘base.dll’) o situado en otro u otros archivos cualesquiera, conforme desee el desarrollador de la base de datos. Las aplicaciones sólo accederán a las funciones del API incluidas en el archivo ‘base.dll’, y éstas serán las encargadas de acceder a la INFORMACION en la forma que el desarrollador de la base implemente.

DEFINICION DEL API: FIEBDC.H

Único archivo que define el ESTANDAR. En este archivo se define el API en C, que las descripciones paramétricas en DLL ofrecen a las aplicaciones. Se recomienda incluir en las Bases de Datos todas las funciones del API, aunque no se utilicen todas por la misma. Este interfaz permite definiciones paramétricas de ilimitado número de parámetros e ilimitadas opciones por parámetro. Se soportan dos modelos de codificaciones:

1. Un modelo de codificación independiente de parámetros, en el que el CODIGO de un concepto paramétrico es completamente libre y el número de caracteres del CODIGO es independiente del número de parámetros. En este modelo, los códigos de las familias paramétricas no tienen por qué tener el carácter $.

2. Un modelo dependiente de los mismos. Es el modelo que definía FIEBDC-3/95 y en el que el CODIGO de un concepto paramétrico debe tener un símbolo ‘$’ en su séptima posición y en el que se asigna de la 'a' a la 'z' las opciones 0 a 25 de cada parámetro, ampliándose en esta VERSION con los rangos 'A' a 'Z' y '0' a '9' para que el número de opciones por parámetro en este modelo de codificación pase a 62 (de 0 a 61).

Para que los programas puedan determinar si una base de datos responde a uno u otro modelo, se ha definido la función BdcCodificacion(), que se especifica más adelante y que indica si el sistema de codificación usado en la base de datos es dependiente o independiente.

Si se adopta el primer modelo, no es posible averiguar ‘a priori’, a partir de un CODIGO ‘ABCDEFGHIJ’ de concepto, si éste es un derivado paramétrico ni de que concepto paramétrico procede o con qué valores de sus parámetros. Por ello, es establece el siguiente criterio de búsqueda:

1. Si el concepto existe con este CODIGO en la base, se escogerá dicho concepto.

2. En caso de no existir, se intentará localizarlo en la base de datos como perteneciente a un concepto paramétrico ‘al estilo’ FIEBDC-3/95. En el ejemplo, se intentará buscar el concepto paramétrico ‘ABCDEF$’ y pasarle los parámetros ‘GHIJ’ (que implica pasarles a sus cuatro parámetros valores ‘31’, ‘32’, ‘33’ y ‘34’ respectivamente).

3. En caso de no existir, se intentará localizarlo en la DLL. Si ésta posee un modelo de codificación dependiente, se utilizará el mismo criterio que en el punto anterior: en el ejemplo, buscar el concepto paramétrico ‘ABCDEF$’ y pasarle los parámetros ‘GHIJ’. Si la base posee un modelo independiente, se utilizará la función ‘BdcDecodifica()’, tal como se especifica más adelante.

4. Si no se cumplen ninguna de las condiciones anteriores, se supone que el concepto no existe en la base.

En la carpeta Anexo3 que acompaña a este documento vea el archivo ‘fiebdc.h’.

ESPECIFICACION DE LAS FUNCIONES DEL API.

1. FUNCIONES GENERALES.

```
LONG EXPORTA BdcCodificacion (
        VOID
);
```

Propósito
Indica si la base de datos utiliza un modelo de codificación dependiente o independiente del número y valor de los parámetros.

Valor devuelto
Devolverá '0' si la codificación sigue un modelo dependiente (al 'estilo' FIEBDC-3/95), y '1' si sigue un modelo independiente.

```
LONG EXPORTA BdcTipoPliego (
        VOID
);
```

Propósito
Indica qué tipo o tipos de Pliegos de Condiciones están implementados en la base. Dichos modelos se especifican en el apartado 'REGISTROS TIPO PLIEGOS' de las especificaciones del formato.

Valor devuelto
Devolverá '0' si no está implementado ningún tipo de pliego.
Devolverá '1' si está implementado el modelo uno de textos de pliegos. Es este caso, se utilizará la función BdcPliego() para obtener los textos de los pliegos.
Devolverá '2' si está implementado el modelo dos de textos de pliegos. En este caso, se utilizarán las funciones BdcCodigoParrafo() y BdcTextoParrafo() para obtener los textos de los pliegos.
Devolverá '3' si están implementados tanto el modelo uno como el modelo dos.

```
LPCSTR EXPORTA BdcFecha (
        VOID
);
```

Propósito
Obtener la fecha de la Base de Datos.

Valor devuelto
Devuelve la fecha de la Base de Datos en formato DDMMAAAA, donde DD representa al día con dos dígitos, MM el mes y AAAA el año.
La propia función es responsable de asignar memoria al puntero. En caso de error, la función devuelve NULL. Para obtener más INFORMACIÓN sobre el error producido, llame a la función BdcError().

2. FUNCIONES REFERENTES AL PARAMÉTRICO GLOBAL.

2.1. Accesibles en cualquier momento.

2.1.1. Obtención de sus parámetros.

```
LONG EXPORTA BdcGloParNumero (
        VOID
);
```

Propósito
Obtener el número de parámetros de concepto paramétrico global.

Valor devuelto
Devuelve el número de parámetros. En caso de error, la función devuelve -1. Para obtener más INFORMACION sobre el error producido, llame a la función BdcGloError().

```
LONG EXPORTA BdcGloOpcNumero (
        LONG par        // número de parámetro del concepto
);
```

Propósito
Obtiene el número de opciones de que consta el parámetro ‘par’.

Parámetros
par: Número del parámetro. Debe ser un valor entre ‘0’ y ‘n-1’, siendo ‘n’ el número de parámetros del concepto paramétrico global.

Valor devuelto
Devuelve el número de opciones del parámetro ‘par’. En caso de producirse un error, devuelve –1. Para obtener más INFORMACION sobre el error producido, llame a la función BdcGloError().

```
LPCSTR EXPORTA BdcGloParRotulo (
        LONG par        // número de parámetro del concepto
);
```

Propósito
Obtiene el rótulo que identifica el parámetro ‘par’ del concepto.

Parámetros
par: Número del parámetro. Debe ser un valor entre ‘0’ y ‘n-1’, siendo ‘n’ el número de parámetros del concepto paramétrico global.

Valor devuelto
Devuelve el rótulo que identifica el parámetro ‘par’ del concepto, como puntero constante ‘far’ a una cadena de caracteres. La propia función es responsable de asignar memoria al puntero. En caso de error, la función devuelve NULL. Para obtener más INFORMACION sobre el error producido, llame a la función BdcGloError().

```
LPCSTR EXPORTA BdcGloOpcRotulo (
        LONG par        // número de parámetro del concepto
```

```
        LONG opc        // número de la opción del parámetro
);
```

Propósito
Obtiene el rótulo que identifica la opción ‘opc’ del parámetro ‘par’ del concepto.

Parámetros
par: Número del parámetro. Debe ser un valor entre ‘0’ y ‘n-1’, siendo ‘n’ el número de parámetros del concepto paramétrico global.
opc: Número del parámetro. Debe ser un valor entre ‘0’ y ‘n-1’, siendo ‘n’ el número de opciones que posee el parámetro ‘par’ del concepto paramétrico global.

Valor devuelto
Devuelve el rótulo que identifica la opción ‘opc’ del parámetro ‘par’ del concepto, como puntero constante ‘far’ a una cadena de caracteres. La propia función es responsable de asignar memoria al puntero. En caso de error, la función devuelve NULL. Para obtener más INFORMACION sobre el error producido, llame a la función BdcGloError().

2.1.2. Mensajes / CODIGOs de error.

```
LONG EXPORTA BdcGloError (
        LPCSTR *err     // mensaje de error devuelto
);
```

Propósito
Obtiene el tipo de error producido. Una vez leído, se inicializa el CODIGO de error.

Parámetros
err: Puntero a un puntero constante ‘far’ a una cadena de caracteres. En él se almacena el mensaje de error referente al error producido. La función es responsable de asignar memoria al puntero. Si no existe un mensaje definido para el error existente, ‘*err’ apuntará a la cadena vacía “”.

Valor devuelto
Devuelve el CODIGO de error producido. Vea al final el apartado ‘CODIGOs de los mensajes de error’ para más INFORMACION.

2.1.3. Asignación de opciones a los parámetros.

```
BOOL EXPORTA BdcGloCalcula (
        LPLONG opcl,  // lista de opciones de los parámetros
);
```

Propósito
Asigna los valores de los parámetros del concepto paramétrico global.

Parámetros
opcl: Puntero a un vector (array) de LONGs con las opciones que se desea fijar a cada parámetro. Las opciones se numeran empezando por cero.

Valor devuelto
Devuelve ‘0’ si se ejecuta correctamente. En caso de error, devuelve ‘-1’. Para obtener más INFORMACION sobre el error producido, llame a la función BdcGloError().

3. FUNCIONES REFERENTES AL RESTO DE PARAMÉTRICOS.

3.1. Accesibles en cualquier momento.

3.1.1. Lectura de un concepto paramétrico.

```
HANDLE EXPORTA BdcLee (
        LPCSTR cod     // CODIGO del concepto
);
```

Propósito
Lee el concepto paramétrico identificado por 'cod'.

Parámetros
cod: Puntero constante 'far' a una cadena de caracteres con el CODIGO del concepto paramétrico a leer. Si se utiliza un modelo de codificación dependiente, se asume que dicho CODIGO tenga 7 caracteres y que el séptimo sea '$'. Dentro del CODIGO, los caracteres pueden ser cualesquiera salvo el 0x00 (que indica el final del CODIGO).

Valor devuelto
Si la función encuentra el paramétrico, retorna un HANDLE distinto de cero. En caso de error, o si no existe el paramétrico, la función devuelve cero.

3.1.2. Lectura de un concepto paramétrico a partir del CODIGO completo del derivado.

```
HANDLE EXPORTA BdcDecodifica (
        LPCSTR cod,    // CODIGO completo del derivado paramétrico
        LPLONG opcl    // puntero al espacio de memoria a rellenar con las opciones
);
```

Propósito
Lee el concepto paramétrico al que pertenece el concepto de CODIGO 'cod'. El HANDLE y las opciones 'opcl' devueltas se pueden utilizar directamente en una llamada a BdcCalcula().

Parámetros
cod: Puntero constante 'far' a una cadena de caracteres con el CODIGO del concepto del que se desea obtener el concepto paramétrico a la que pertenece. Dentro del CODIGO, los caracteres pueden ser cualesquiera salvo el 0x00 (que indica el final del CODIGO).
opcl: Puntero a un vector (array) de LONGs en el que la función devolverá las opciones a las que corresponda el derivado paramétrico. El array debe estar previamente dimensionado con al menos el número de parámetros del concepto. Las opciones se numeran empezando por cero.

Valor devuelto
Si la función encuentra el paramétrico, retorna un HANDLE distinto de cero. En caso de error, o si no existe ningún concepto paramétrico del que el concepto 'cod' es derivado, la función devuelve cero.

```
LPCSTR EXPORTA BdcFamilia (
        LPCSTR cod)    // CODIGO completo del derivado paramétrico
);
```

Propósito
Lee el código del concepto paramétrico al que pertenece el concepto de CODIGO 'cod'.

Parámetros

cod: Puntero constante ‘far’ a una cadena de caracteres con el CODIGO del concepto del que se desea obtener el concepto paramétrico al que pertenece. Dentro del CODIGO, los caracteres pueden ser cualesquiera salvo el 0x00 (que indica el final del CODIGO).

Valor devuelto
Si la función encuentra el paramétrico, retorna su código como puntero constante ‘far’ a una cadena de caracteres. La propia función es responsable de asignar memoria al puntero. En caso de error, o si no existe ningún concepto paramétrico del que el concepto ‘cod’ es derivado, la función devuelve NULL. Para obtener más INFORMACION sobre el error producido, llame a la función BdcError().

3.1.3. Búsqueda de conceptos a partir de un texto o un código.

```
BOOL EXPORTA BdcUsos (
        LPCSTR texto, // código o texto a buscar
        LONG donde // lugar donde tengo que buscar el texto o código
        LPCSTR *lcodigo // lista de códigos donde se está utilizando el texto/código
)
```

Propósito
Obtiene una lista de códigos dónde se utiliza un texto o un código de un concepto.

Parámetros
Texto: Indica el texto o código que se quiere buscar.
Donde: Parámetro que le indica a la función si el texto es un código o bien un texto (que puede ser una cadena). Los valores posibles son 1: cuando el parámetro es un código, 2: cuando el parámetro es un texto.
lcodigo: Texto en formato texto plano que contiene la lista de códigos donde se utiliza el texto o código separados por el separador de campos ASCII 124.

Valor devuelto
Devuelve '0' si se ejecuta correctamente. En caso de error, la función devuelve '-1'. Para obtener más INFORMACIÓN sobre el error producido, llame a la función BdcError().

Si el parámetro ‘donde’ es 1 el texto corresponde a un código del banco y la función devolverá la lista de conceptos donde aparece el código en la justificación directamente o bien en alguno de sus componentes. Si el parámetro ‘donde’ es 2 el primer parámetro corresponde a un texto y la función devolverá la lista de los conceptos que contienen el texto en la definición (resumen y completa).

3.1.4. Mensajes / CODIGOs de error.

```
LONG EXPORTA BdcError (
        HANDLE h,      // identificador del concepto
        LPCSTR *err    // mensaje de error devuelto
);
```

Propósito
Obtiene tipo de error producido.

Parámetros
h: Identificador (HANDLE) del concepto, que debe ser obtenido en una llamada anterior a la función BdcLee().
err: Puntero a un puntero constante ‘far’ a una cadena de caracteres. En él se almacena el mensaje de error referente al error producido. La función es responsable de asignar memoria al puntero. Si no existe un mensaje definido para el error existente, ‘*err’ apuntará a la cadena vacía “”.

Valor devuelto
Devuelve el CODIGO de error producido. Vea al final el apartado ‘CODIGOs de los mensajes de error’ para más INFORMACION.

3.2. Accesibles después de BdcLee.

3.2.1. Obtención de sus parámetros.

```
LONG EXPORTA BdcParNumero (
        HANDLE h        // identificador del concepto
);
```

Propósito
Obtiene el número de parámetros de concepto paramétrico.

Parámetros
h: Identificador (HANDLE) del concepto paramétrico, que debe ser obtenido en una llamada anterior a la función BdcLee().

Valor devuelto
Devuelve el número de parámetros. En caso de error, la función devuelve -1. Para obtener más INFORMACION sobre el error producido, llame a la función BdcError().

```
LONG EXPORTA BdcOpcNumero (
        HANDLE h,       // identificador del concepto
        LONG par        // número de parámetro del concepto
);
```

Propósito
Obtiene el número de opciones de que consta el parámetro ‘par’.

Parámetros
h: Identificador (HANDLE) del concepto paramétrico, que debe ser obtenido en una llamada anterior a la función BdcLee().
par: Número del parámetro. Debe ser un valor entre ‘0’ y ‘n-1’, siendo ‘n’ el número de parámetros del concepto.

Valor devuelto
Devuelve el número de opciones del parámetro ‘par’. En caso de producirse un error, la función devuelve -1. Para obtener más INFORMACION sobre el error producido, llame a la función BdcError().

```
LPCSTR EXPORTA BdcParRotulo (
        HANDLE h,       // identificador del concepto
        LONG par        // número de parámetro del concepto
);
```

Propósito
Obtiene el rótulo que identifica el parámetro ‘par’ del concepto.

Parámetros

h: Identificador (HANDLE) del concepto paramétrico, que debe ser obtenido en una llamada anterior a la función BdcLee().
par: Número del parámetro. Debe ser un valor entre ‘0’ y ‘n-1’, siendo ‘n’ el número de parámetros del concepto.

Valor devuelto
Devuelve el rótulo que identifica el parámetro ‘par’ del concepto, como puntero constante ‘far’ a una cadena de caracteres. La propia función es responsable de asignar memoria al puntero. En caso de error, la función devuelve NULL. Para obtener más INFORMACION sobre el error producido, llame a la función BdcError().

```
LPCSTR EXPORTA BdcOpcRotulo (
        HANDLE h,     // identificador del concepto
        LONG par,     // número de parámetro del concepto
        LONG opc      // número de la opción del parámetro
);
```

Propósito
Obtiene rótulo que identifica la opción ‘opc’ del parámetro ‘par’ del concepto.

Parámetros
h: Identificador (HANDLE) del concepto paramétrico, que debe ser obtenido en una llamada anterior a la función BdcLee().
par: Número del parámetro. Debe ser un valor entre ‘0’ y ‘n-1’, siendo ‘n’ el número de parámetros del concepto.
opc: Número del parámetro. Debe ser un valor entre ‘0’ y ‘n-1’, siendo ‘n’ el número de opciones que posee el parámetro ‘par’ del concepto.

Valor devuelto
Devuelve el rótulo que identifica la opción ‘opc’ del parámetro ‘par’ del concepto, como puntero constante ‘far’ a una cadena de caracteres. La propia función es responsable de asignar memoria al puntero. En caso de error, la función devuelve NULL. Para obtener más INFORMACION sobre el error producido, llame a la función BdcError(). En caso de referirse a ámbitos territoriales y a divisas, véanse los anexos 5 y 6.

3.2.2. Obtención de un comentario.

```
LPCSTR EXPORTA BdcComentario (
        HANDLE h      // identificador del concepto
);
```

Propósito
Obtiene un texto de comentario del concepto paramétrico.

Parámetros
h: Identificador (HANDLE) del concepto paramétrico, que debe ser obtenido en una llamada anterior a la función BdcLee().

Valor devuelto
Devuelve el comentario del concepto, como puntero constante ‘far’ a una cadena de caracteres. La propia función es responsable de asignar memoria al puntero. En caso de error, la función devuelve NULL. Para obtener más INFORMACION sobre el error producido, llame a la función BdcError().

3.2.3. Asignación de opciones de los parámetros y validación o cálculo del derivado.

```
BOOL EXPORTA BdcValida (
        HANDLE h,      // identificador del concepto
        LPLONG opcl,   // lista de opciones de los parámetros
);
```

Propósito
Averigua si una determinada combinación paramétrica es correcta o no.

Parámetros
h: Identificador (HANDLE) del concepto paramétrico, que debe ser obtenido en una llamada anterior a la función BdcLee().
opcl: Puntero a un vector (array) de LONGs con las opciones que se desea fijar a cada parámetro. Las opciones se numeran empezando por cero.

Valor devuelto
Devuelve '0' si la combinación es correcta. En caso contrario, devuelve '-1'. Para obtener más INFORMACION sobre el error producido, llame a la función BdcError().

```
BOOL EXPORTA BdcCalcula (
        HANDLE h,      // identificador del concepto
        LPLONG opcl,   // lista de opciones de los parámetros
);
```

Propósito
Calcula los datos correspondientes a un derivado paramétrico.

Parámetros
h: Identificador (HANDLE) del concepto paramétrico, que debe ser obtenido en una llamada anterior a la función BdcLee().
opcl: Puntero a un vector (array) de LONGs con las opciones que se desea fijar a cada parámetro. Las opciones se numeran empezando por cero.

Valor devuelto
Devuelve '0' si se ejecuta correctamente. En caso de error, o de que la combinación no sea correcta, devuelve '-1'. Para obtener más INFORMACION sobre el error producido, llame a la función BdcError().

```
LONG EXPORTA BdcValidos (
        HANDLE h,      // identificador del concepto
        LPBYTE *opcl,  // lista de opciones de los parámetros de todos los derivados válidos
);
```

Propósito
Obtiene las opciones de cada parámetro de todos los derivados paramétricos válidos de la familia paramétrica. Si no se desea implementar esta función en una determinada familia, la función deberá devolver '0'. Está pensada para aquellas familias que posean un número elevado de combinaciones posibles, en las que el averiguar cuáles de ellas son válidas mediante sucesivas llamadas a las funciones BdcCalcula() o BdcValida() es muy costoso en tiempo. En una sola llamada a esta función es posible, en estos casos, obtener todas las combinaciones válidas.

Parámetros

h: Identificador (HANDLE) del concepto paramétrico, que debe ser obtenido en una llamada anterior a la función BdcLee().

opcl: Puntero a una matriz (array) bidimensional de BYTEs a rellenar por la función, en el que cada columna corresponde a un derivado paramétrico válido, y cada fila corresponde a los valores de cada uno de los parámetros. La matriz se devuelve al estilo de 'C'; es decir, por columnas. Las opciones se numeran empezando por cero. La propia función es responsable de asignar memoria al puntero. La memoria asignada en bytes será el número de combinaciones válidas multiplicado por el número de parámetros.

Valor devuelto

Devuelve el número de derivados paramétricos válidos. En caso de que dicha información no esté disponible, devuelve '0'. En caso de error, devuelve '-1'. Para obtener más INFORMACION sobre el error producido, llame a la función BdcError().

Ordenación

Las combinaciones válidas para devolver se ordenarán teniendo en cuenta que el primer parámetro tiene más precedencia que el segundo, éste que el tercero, y así sucesivamente.

Ejemplo

Si una familia paramétrica tiene 3 parámetros, con 9, 10 y 11 opciones por parámetro respectivamente, el número total de combinaciones posibles es 9x10x11 = 990. Si son válidas sólo las siguientes cuatro (escritos en el orden especificado en el párrafo anterior):

1. Valores '0', '5', y '9' de los parámetros uno al tres, respectivamente.
2. Valores '7', '6' y '5' de los parámetros uno al tres, respectivamente.
3. Valores '8', '2' y '4' de los parámetros uno al tres, respectivamente.
4. Valores '8', '2' y '10' de los parámetros uno al tres, respectivamente.

Entonces la función devolverá los 4x3=12 bytes siguientes: 0, 5, 9, 7, 6, 5, 8, 2, 4, 8, 2 y 10.

3.2.4. Liberación de memoria.

```
LONG EXPORTA BdcInValidos (
        HANDLE h,      // identificador del concepto
        LPBYTE *opcl,  // lista de opciones de los parámetros de todos los derivados inválidos
);
```

Con el mismo propósito y parámetros que BdcValidos.

Esta función se implementará en lugar de aquella cuando no exista BdcValidos para un concepto paramétrico y viceversa.

```
BOOL EXPORTA BdcCierra (
        HANDLE h       // identificador del concepto
);
```

Propósito

Cierra el concepto paramétrico y libera la memoria asignada.

Parámetros

h: Identificador (HANDLE) del concepto paramétrico, que debe ser obtenido en una llamada anterior a la función BdcLee().

Valor devuelto

Devuelve '0' si realiza la operación correctamente. En caso de error, la función devuelve '-1'. Para obtener más INFORMACION sobre el error producido, llame a la función BdcError().

3.3. Accesibles después de BdcCalcula.

3.3.1. Obtención del derivado paramétrico.

```
LONG EXPORTA BdcDesNumero (
        HANDLE h        // identificador del concepto
);
```

Propósito
Obtiene el número de conceptos en los que se descompone el derivado paramétrico.

Parámetros
h: Identificador (HANDLE) del concepto paramétrico, que debe ser obtenido en una llamada anterior a la función BdcLee().

Valor devuelto
Devuelve el número de elementos de su descomposición. Un valor de cero indicará que el concepto no tiene descomposición. Es posible que un mismo concepto paramétrico posea derivados simples y compuestos. En caso de error, la función devuelve '-1'. Para obtener más INFORMACION sobre el error producido, llame a la función BdcError().

```
LPCSTR EXPORTA BdcDesCodigo (
        HANDLE h,       // identificador del concepto
        LONG des        // número del elemento de la descomposición
);
```

Propósito
Obtiene el CODIGO del elemento número 'des' en el que se descompone el derivado paramétrico.

Parámetros
h: Identificador (HANDLE) del concepto paramétrico, que debe ser obtenido en una llamada anterior a la función BdcLee().
des: Número del elemento de la descomposición del concepto. Los elementos se numeran empezando por cero.

Valor devuelto
Devuelve el CODIGO del elemento número 'des' en el que se descompone el derivado paramétrico, como puntero constante 'far' a una cadena de caracteres. La propia función es responsable de asignar memoria al puntero. En caso de error, la función devuelve NULL. Para obtener más INFORMACION sobre el error producido, llame a la función BdcError().

```
BOOL EXPORTA BdcFactor (
        HANDLE h,               // identificador del concepto
        LONG des,               // número del elemento de la descomposición
        double FAR *factor      // factor a obtener
);
```

Propósito
Obtiene el factor de rendimiento del elemento número 'des' en el que se descompone el derivado paramétrico.

Parámetros

h: Identificador (HANDLE) del concepto paramétrico, que debe ser obtenido en una llamada anterior a la función BdcLee().
des: Número del elemento de la descomposición del concepto. Los elementos se numeran empezando por cero.
*factor: Puntero en el que devolver el factor deseado, por defecto 1. El factor puede ser positivo, cero o negativo.

Valor devuelto
Devuelve '0' si se ejecuta correctamente. En caso de error, el factor se asigna a cero y la función devuelve '-1'. Para obtener más INFORMACION sobre el error producido, llame a la función BdcError().

```
BOOL EXPORTA BdcRendimiento (
        HANDLE h,           // identificador del concepto
        LONG des,           // número del elemento de la descomposición
        double FAR *ren     // rendimiento a obtener
);
```

Propósito
Obtiene el rendimiento del elemento número 'des' en el que se descompone el derivado paramétrico.

Parámetros
h: Identificador (HANDLE) del concepto paramétrico, que debe ser obtenido en una llamada anterior a la función BdcLee().
des: Número del elemento de la descomposición del concepto. Los elementos se numeran empezando por cero.
*ren: Puntero en el que devolver el rendimiento deseado. El rendimiento puede ser positivo, cero o negativo.

Valor devuelto
Devuelve '0' si se ejecuta correctamente. En caso de error, el rendimiento se asigna a cero y la función devuelve '-1'. Para obtener más INFORMACION sobre el error producido, llame a la función BdcError().

```
LPCSTR EXPORTA BdcDesCodigoPorcentaje (
        HANDLE h,       // identificador del concepto
        LONG des        // número del elemento de la descomposición
);
```

Propósito
Obtiene los CODIGO_PORCENTAJE aplicable al elemento número 'des' en el que se descompone el derivado paramétrico.

Parámetros
h: Identificador (HANDLE) del concepto paramétrico, que debe ser obtenido en una llamada anterior a la función BdcLee().
des: Número del elemento de la descomposición del concepto. Los elementos se numeran empezando por cero.

Valor devuelto
Devuelve los CODIGO_PORCENTAJE aplicables al elemento número 'des' en el que se descompone el derivado paramétrico, como puntero constante 'far' a una cadena de caracteres. La propia función es responsable de asignar memoria al puntero. Los CODIGO_PORCENTAJE van separados el carácter < ; > (ASCII-59) y puede ser que se devuelva la cadena vacía si no se aplica ningún

porcentaje sobre la línea. En caso de error, la función devuelve NULL. Para obtener más INFORMACION sobre el error producido, llame a la función BdcError().

```
BOOL EXPORTA BdcPrecio (
        HANDLE h,               // identificador del concepto
        double FAR *pre         // precio unitario a devolver
);
```

Propósito
Obtiene el precio unitario en el caso de que el derivado paramétrico sea un simple. Es posible que un mismo concepto paramétrico tenga como derivados tanto simples como compuestos.

Parámetros
h: Identificador (HANDLE) del concepto paramétrico, que debe ser obtenido en una llamada anterior a la función BdcLee().
*pre: Puntero en el que devolver el precio unitario. Dicho precio puede ser positivo, cero o negativo.

Valor devuelto
Devuelve '0' si se ejecuta correctamente. En caso de error, el precio se asigna a cero y la función devuelve '-1'. Para obtener más INFORMACION sobre el error producido, llame a la función BdcError().

```
LPCSTR EXPORTA BdcCodigo (
        HANDLE h        // identificador del concepto
);
```

Propósito
Obtiene el CODIGO del concepto.

Parámetros
h: Identificador (HANDLE) del concepto paramétrico, que debe ser obtenido en una llamada anterior a la función BdcLee().

Valor devuelto
Devuelve el CODIGO del concepto, como puntero constante 'far' a una cadena de caracteres. Si se ha calculado un derivado paramétrico (se ha llamado a BdcCalcula), este CODIGO será el del derivado paramétrico. En caso contrario, será el CODIGO del concepto paramétrico. La propia función es responsable de asignar memoria al puntero. En caso de error, la función devuelve NULL. Para obtener más INFORMACION sobre el error producido, llame a la función BdcError().

```
LPCSTR EXPORTA BdcUnidad (
        HANDLE h        // identificador del concepto
);
```

Propósito
Obtiene la unidad de medida del derivado paramétrico. Esta función permite que un concepto paramétrico pueda generar elementos derivados con distintas unidades de medición. Para que dicha función actúe, el registro ~C debe contener el carácter especial '*' dentro del campo unidad de medida. Dicho carácter indica que la unidad de medida de los conceptos derivados los debe proporcionar el API. Si el redactor de la BD desea que la unidad de medida sea un valor para escoger por parte del usuario, deberá además añadirse la unidad de medida como una propiedad más del concepto paramétrico.

Parámetros
h: Identificador (HANDLE) del concepto paramétrico, que debe ser obtenido en una llamada anterior a la función BdcLee().

Valor devuelto
Devuelve el texto correspondiente a la unidad de medida del derivado paramétrico, como puntero constante 'far' a una cadena de caracteres. La propia función es responsable de asignar memoria al puntero. Si no existe definido un texto resumido, la función devuelve la cadena vacía "". En caso de error, la función devuelve NULL. Para obtener más INFORMACION sobre el error producido, llame a la función BdcError().

```
LPCSTR EXPORTA BdcResumen (
        HANDLE h        // identificador del concepto
);
```

Propósito
Obtiene el texto resumido del derivado paramétrico.

Parámetros
h: Identificador (HANDLE) del concepto paramétrico, que debe ser obtenido en una llamada anterior a la función BdcLee().

Valor devuelto
Devuelve el texto resumido del derivado paramétrico, como puntero constante 'far' a una cadena de caracteres. La propia función es responsable de asignar memoria al puntero. Si no existe definido un texto resumido, la función devuelve la cadena vacía "". En caso de error, la función devuelve NULL. Para obtener más INFORMACION sobre el error producido, llame a la función BdcError().

```
LPCSTR EXPORTA BdcTexto (
        HANDLE h        // identificador del concepto
);
```

Propósito
Obtiene el texto completo de descripción del derivado paramétrico.

Parámetros
h: Identificador (HANDLE) del concepto paramétrico, que debe ser obtenido en una llamada anterior a la función BdcLee().

Valor devuelto
Devuelve el texto completo de descripción del derivado paramétrico, como puntero constante 'far' a una cadena de caracteres. La propia función es responsable de asignar memoria al puntero. Si no existe definido un texto completo de descripción, la función devuelve la cadena vacía "". En caso de error, la función devuelve NULL. Para obtener más INFORMACION sobre el error producido, llame a la función BdcError().

```
VOID EXPORTA BdcTipoDescripcion (
        LONG tipo       // tipo de descripción
);
```

Propósito

Indica qué tipo de descripción se va a obtener de un concepto derivado.

Parámetros
Tipo: Se refiere a las diferentes descripciones (obtenidas con BdcResumen y BdcTexto) que puede tener un concepto derivado en función de los diferentes contenidos de que trate la base de datos.
Valor '0': Descripción por defecto de un derivado paramétrico.
Valor '1': Descripción de un derivado paramétrico que contiene insertados uno o varios códigos de conceptos que intervienen en su descomposición. Dichos códigos aparecen indicados entre dólares ($). Esta opción sirve para que los programas puedan particularizar la descripción de un concepto derivado cuando se han proporcionado productos comerciales asociados a sus componentes (con BdcComercNumero y BdcComercCodigo, o con el registro ~O).

Ejemplo: BdcTexto del concepto derivado E612235K.
Devolución de BdcTexto con valor 0 para BdcTipoDescripcion: Inodoro con tapa.
Devolución de BdcTexto con valor 1 para BdcTipoDescripcion: Inodoro $B1234567$ con tapa $B7654321$.

```
LPCSTR EXPORTA BdcPliego (
        HANDLE h,           // identificador del concepto
        LONG formato,       // identificador del formato
        LONG tipo,          // especifica si el pliego a obtener es de la familia o del derivado
        LPCSTR seccion,     // código de la sección del pliego
        LPCSTR ambito       // abreviatura del ámbito
);
```

Propósito
Obtiene el texto del pliego, según el modelo uno.

Parámetros
h: Identificador (HANDLE) del concepto paramétrico, que debe ser obtenido en una llamada anterior a la función BdcLee().

formato: Identificador del formato de texto. Se definen, por el momento, los siguientes formatos: 'BDCFMT_ASCII' (si se desea obtener el pliego en formato ASCII), 'BDCFMT_ANSI' (si se desea obtener el pliego en formato ANSI), 'BDCFMT_RTF' (si se desea obtener el pliego en formato RTF) y 'BDCFMT_HTM' (si se desea obtener el pliego en formato HTM).

tipo: Especifica si se desea obtener el pliego común de la familia paramétrica (valor 'BDCPLI_FAMILIA') o el del derivado (valor 'BDCPLI_DERIVADO'). En el primer caso, no es necesario haber realizado previamente una llamada a BdcCalcula().

seccion: Puede tomar como valor el código de la sección del pliego cuyo texto se desea obtener, en cuyo caso deberá ser una de las secciones especificadas en el registro ~L, según se indica en el apartado 'REGISTROS TIPO PLIEGOS' de la especificación del formato. Si no existen secciones definidas, el valor de este parámetro no es utilizado. Si por el contrario toma el valor NULL, la función devolverá los códigos de las secciones del pliego para los que el concepto posea texto del pliego, separados por el separador de de subcampos ('\').

ambito: Ámbito del cual se desea obtener el texto del pliego. Corresponde a uno de los campos 'ABREV_AMBITO' especificados en el registro ~W, según se indica en el apartado 'REGISTRO TIPO ÁMBITO GEOGRÁFICO' de las especificaciones del formato. Si no existen ámbitos definidos (no existe un registro ~W), el valor de este parámetro es ignorado. En ese caso, es posible que el ámbito sea un parámetro global de la base.

Valor devuelto
Si el parámetro sección contiene el código de la sección del pliego cuyo texto se desea obtener, la función devuelve el texto del pliego del derivado paramétrico en el formato solicitado, como puntero

constante ‘far’ a una cadena de caracteres. Si no existe definido un texto de pliego, la función devuelve la cadena vacía “”.
Cuando el parámetro sección es NULL, la función devuelve los rótulos de los pliegos existentes asociados al concepto paramétrico en el formato {rotulo\}, si no hubiera rótulos asociados al concepto la función devuelve "".
La propia función es responsable de asignar memoria al puntero. En caso de error (por ejemplo, formato no soportado por la BDC), la función devuelve NULL. Para obtener más INFORMACION sobre el error producido, llame a la función BdcError().

```
LPCSTR EXPORTA BdcCodParrafo (
        HANDLE h,           // identificador del concepto
        LONG tipo,          // especifica si el pliego a obtener es de la familia o del derivado
        LPCSTR seccion,     // código de la sección del pliego
        LPCSTR ambito       // abreviatura del ámbito
);
```

Propósito
Obtiene los códigos de los párrafos de pliego, según el modelo dos.

Parámetros
h: Identificador (HANDLE) del concepto paramétrico, que debe ser obtenido en una llamada anterior a la función BdcLee().
tipo: Especifica si se desea obtener el pliego común de la familia paramétrica (valor ‘BDCPLI_FAMILIA’) o el del derivado (valor ‘BDCPLI_DERIVADO’). En el primer caso, no es necesario haber realizado previamente una llamada a BdcCalcula().
seccion: Código de la sección del pliego cuyos códigos de párrafo se desea obtener. Deberá ser una de las secciones especificadas en el registro ~L, según se indica en el apartado ‘REGISTROS TIPO PLIEGOS’ de la especificación del formato. Si no existen secciones definidas, el valor de este parámetro no es utilizado.
ambito: Ámbito del cual se desea obtener los códigos de párrafo. Corresponde a uno de los campos ‘ABREV_AMBITO’ especificados en el registro ~W, según se indica en el apartado ‘REGISTRO TIPO ÁMBITO GEOGRÁFICO’ de las especificaciones del formato. Si no existen ámbitos definidos (no existe un registro ~W), el valor de este parámetro es ignorado. En ese caso, es posible que el ámbito sea un parámetro global de la base.

Valor devuelto
Devuelve un texto con los códigos de párrafo del pliego del derivado paramétrico, como puntero constante ‘far’ a una cadena de caracteres. La propia función es responsable de asignar memoria al puntero. Si no existe definido ningún código de párrafo, la función devuelve la cadena vacía “”. En caso de error (por ejemplo, ámbito no soportado por la BDC), la función devuelve NULL. Para obtener más INFORMACION sobre el error producido, llame a la función BdcError(). Los códigos irán separados con el separador de subcampos habitual, es decir, el texto devuelto tendrá la sintaxis
{ CODIGO_PARRAFO \ }

```
LPCSTR EXPORTA BdcTexParrafo (
        LONG formato        // identificador del formato
        LPCSTR cod_parrafo  // código del párrafo del pliego
);
```

Propósito
Obtiene el texto del pliego del derivado paramétrico correspondiente al código de párrafo ‘cod_parrafo’, según el modelo dos. El código de párrafo se obtiene mediante una llamada a la función BdcCodParrafo.

Parámetros

formato: Identificador del formato de texto. Se definen, por el momento, los siguientes formatos: 'BDCFMT_ASCII' (si se desea obtener el pliego en formato ASCII), 'BDCFMT_RTF' (si se desea obtener el pliego en formato RTF) y 'BDCFMT_HTM' (si se desea obtener el pliego en formato HTM).

cod_parrafo: Código del párrafo del pliego cuyo texto se desea obtener.

Valor devuelto

Devuelve el texto del párrafo en el formato solicitado, como puntero constante 'far' a una cadena de caracteres. La propia función es responsable de asignar memoria al puntero. Esta memoria se liberará en la siguiente llamada a esta función dentro del mismo proceso. Si no existe definido un texto de pliego con este código de párrafo, la función devuelve la cadena vacía "". En caso de error (por ejemplo, formato no soportado por la BDC), la función devuelve NULL. Para obtener más INFORMACION sobre el error producido, llame a la función BdcError().

```
LPCSTR EXPORTA BdcClaves (
        HANDLE h        // identificador del concepto
);
```

Propósito

Obtiene las claves de tesauro del derivado paramétrico.

Parámetros

h: Identificador (HANDLE) del concepto paramétrico, que debe ser obtenido en una llamada anterior a la función BdcLee().

Valor devuelto

Devuelve las claves de tesauro del derivado paramétrico, como puntero constante 'far' a una cadena de caracteres, con el mismo formato que el registro '~A', es decir, '{CLAVE_TESAURO\}'. La propia función es responsable de asignar memoria al puntero. Si no existen definidas claves del tesauro, la función devuelve la cadena vacía "". En caso de error, la función devuelve NULL. Para obtener más INFORMACION sobre el error producido, llame a la función BdcError().

```
LPCSTR EXPORTA BdcTipo (
        HANDLE h,               // identificador del concepto
);
```

Propósito

Obtener el tipo del concepto.

Parámetros

h: Identificador (HANDLE) del concepto paramétrico, que debe ser obtenido en una llamada anterior a la función BdcLee().

Valor devuelto

Devuelve el tipo del concepto, como puntero constante 'far' a una cadena de caracteres. Si se ha calculado un derivado paramétrico (se ha llamado a la función BdcCalcula) este tipo será el del derivado paramétrico. En caso contrario, será el tipo del concepto paramétrico. La propia función es responsable de asignar memoria al puntero. En caso de error, la función devuelve NULL. Para obtener más INFORMACION sobre el error producido, llame a la función BdcError(). Corresponde al campo TIPO del registro ~C.

```
LONG EXPORTA BdcComercNumero(
        HANDLE h                    // identificador del concepto
        LPCSTR   codigo_raiz_BD     // identificador de la entidad que elabora la BD
);
```

Propósito
Obtener el número de conceptos vinculados a un concepto que pertenece a codigo_raiz_BD y empleado por ésta en su sistema de clasificación por codificación.

Parámetros
h: Identificador (HANDLE) del concepto paramétrico.
codigo_raiz_BD: Se refiere a la identificación del CODIGO de la entidad que elabora la BD. Este CODIGO debe ser facilitado por la entidad que elabora la BD, para evitar ambigüedades. Se recomienda que éste sea el propio CIF de la entidad.

Valor devuelto
Devuelve el número de conceptos vinculados. Si se ha calculado un derivado paramétrico (se ha llamado a la función BdcCalcula) este número será el del derivado paramétrico. En caso contrario, será el número del concepto paramétrico. Un valor de cero significa que no tiene conceptos vinculados. En caso de error la función devuelve '-1' . Para obtener más INFORMACIÓN sobre el error producido, llame a la función BdcError().

```
BOOL EXPORTA BdcComercCodigo(
        HANDLE h                    // identificador del concepto
        LONG comerc                 // número del concepto vinculado
        LPLCSTR *Codigo_archivo     // nombre de archivo a devolver
        LPLCSTR *Codigo_Entidad     // identificación del código de la entidad a la que se
                                    // asocia la información a devolver
        LPLCSTR *Codigo_Concepto    // concepto que pertenece a Codigo_Entidad a devolver
);
```

Propósito
Obtener el código de archivo que indica el lugar donde se encuentra la información referente a codigo_entidad#codigo_concepto, o bien el codigo_entidad y el codigo_concepto que se encuentra en la misma Base de Datos.

Parámetros
h: Identificador (HANDLE) del concepto paramétrico.
comerc: Número del concepto vinculado. Los conceptos vinculados se numeran empezando por cero.
codigo_archivo: Se refiere al nombre del archivo que, de existir, indica el lugar donde se encuentra la información referente a codigo_entidad#codigo_concepto. Si el valor es nulo los campos código entidad y código concepto deberán existir. La propia función es responsable de asignar memoria al puntero.
codigo_entidad: Se refiere a la identificación del CODIGO de la entidad a la que se le asocia INFORMACION. Este CODIGO debe ser facilitado por la entidad que elabora la BD, de acuerdo con su sistema de clasificación, para evitar ambigüedades. Se recomienda que éste sea el propio CIF de la entidad. La propia función es responsable de asignar memoria al puntero.
codigo_concepto: Se refiere a un concepto que pertenece a CODIGO_ENTIDAD, y empleado por la entidad que elabora la BD. Cuando CODIGO_CONCEPTO se refiera a un producto comercial, dicho CODIGO deberá ser facilitado por el fabricante, y no podrá coincidir nunca con la designación de CODIGO_RAIZ_BD, CODIGO_ENTIDAD o CODIGO_CONCEPTO, cuando éste se refiere a un concepto genérico. La propia función es responsable de asignar memoria al puntero.
Al tener dicho producto comercial un tratamiento de CONCEPTO, éste puede utilizar todos los registros existentes en el formato para especificar su INFORMACION asociada (precio,

INFORMACION gráfica, etc.). Para poder utilizar los registros mencionados, el código identificador del concepto será CODIGO_ENTIDAD # CODIGO_CONCEPTO.

Valor devuelto
Devuelve '0' si se ejecuta correctamente. En caso de error, la función devuelve '-1'. Para obtener más INFORMACIÓN sobre el error producido, llame a la función BdcError().

```
LONG EXPORTA BdcDocNumero (
        HANDLE h        // identificador del concepto
);
```

Propósito
Obtener el número de archivos asociados a un concepto.

Parámetros
h: Identificador (HANDLE) del concepto.

Valor devuelto
Devuelve el número de archivos asociados. Un valor de cero significa que no tiene archivos asociados. En caso de error la función devuelve '-1'. Para obtener más INFORMACIÓN sobre el error producido, llame a la función BdcError()

```
BOOL EXPORTA BdcDocCodigo (
        HANDLE  h,              // identificador del concepto.
        LONG num,               // número del archivo asociado.
        LONG * tipo             // clasificación del tipo de información del archivo.
        LPCSTR * archivo,       // nombre del archivo que contiene información del concepto.
        LPCSTR * descripción    // descripción de la información que contiene el archivo.
);
```

Propósito
Obtener el archivo asociado a un concepto, así como su clasificación temática y su descripción asociada. El comportamiento sería igual al discreto, primero se buscaría el archivo en el directorio local, y si no estuviera, en URL_BASE+archivo. URL_BASE se define en el registro ~V.

Parámetros
h: Identificador (HANDLE) del concepto.
num: Número del archivo asociado al concepto. Se numeran empezando por cero.
tipo: Devuelve el código de uno de los tipos enumerados en el registro ~F.
archivo: Devuelve el nombre del archivo con extensión que contiene información del concepto. El archivo puede contener una URL_EXT. Las extensiones permitidas son las que se especifican en el registro ~F. En el caso que se devuelvan varios archivos, el primero de los archivos es el principal y el resto son vinculados. En este caso el formato del string sería el siguiente: “archivo 1 | ... | archivo n | ”
descripción: Devuelve una breve descripción de la información contenida en el archivo. devuelve NULL si no hay ningún texto descriptivo asociado.

Valor devuelto
Devuelve '0' si se ejecuta correctamente. En caso de error, la función devuelve '-1'. Para obtener más INFORMACIÓN sobre el error producido, llame a la función BdcError().

```
LONG EXPORTA BdcNumProp (
        HANDLE h,       // identificador del concepto
```

```
        LONG uso        // aplicación para la que se requiere el número de propiedades
                        // técnicas
);
```

Propósito
Obtiene el número de propiedades técnicas que hay definidas para un concepto para el uso que se solicita. Inicialmente se define el uso 0 para el cálculo del coste energético, emisión de CO2 y/o residuos.

Parámetros
h: Identificador (HANDLE) del concepto paramétrico, que debe ser obtenido en una llamada anterior a la función BdcLee().
Uso: Número que indica qué cálculo se está definiendo.

Valor devuelto
Número de propiedades técnicas definidas para el concepto paramétrico para ese uso.

```
BOOL EXPORTA BdcPropValString (
        HANDLE h,               // identificador del concepto
        LONG uso,               // aplicación para la que se requiere el número de propiedades
                                // técnicas
        LONG nprop,             // número de propiedad
        LPCSTR *propiedad,      // puntero al nombre de la propiedad
        LPCSTR *valor,          // puntero al valor de la propiedad
        LPCSTR *um              // puntero a la unidad de medida de la propiedad
);
```

Propósito
Obtiene el nombre de la propiedad técnica, su valor y la unidad de medida.

Parámetros
h: Identificador (HANDLE) del concepto paramétrico.
uso: Número que indica qué cálculo se está definiendo.
nprop: Número de propiedad. Se numeran empezando por cero.
propiedad: Puntero al CODIGO_IT de una de las propiedades enumeradas en el registro ~X.
valor: Puntero al valor de la propiedad que puede ser alfabético o numérico.
um: Puntero a la unidad de medida de la propiedad. En el caso que los valores de la propiedad sean valores numéricos se indicará de acuerdo con el Sistema Internacional de Unidades de Medida (ver anexo 7).

Valor devuelto
Devuelve ‘0’ si se ejecuta correctamente. Para obtener más INFORMACION sobre el error producido, llame a la función BdcError().

```
LONG EXPORTA BdcNumComponentes (
        HANDLE h,               // identificador del concepto
        LONG tipo               // tipo de descomposición
);
```

Propósito
Obtiene el número de componentes de un concepto para cada tipo de descomposición.

Parámetros
h: Identificador (HANDLE) del concepto paramétrico.

tipo: Número del tipo de descomposición descrito en el campo TIPO_DESCOMPOSICION del registro ~R.

Valor devuelto
Devuelve el número de componentes de la relación tipo.
Si el valor devuelto = 0, el concepto no tiene componentes de este tipo.
Si el valor devuelto = -1, el concepto tiene la información incompleta.

```
BOOL EXPORTA BdcCodigoComponente (
        HANDLE h,           // identificador del concepto
        LONG tipo,          // tipo de descomposición
        LONG ncomp,         // número del componente de la descomposición tipo del
                            // concepto
        LPCSTR *codigo      // puntero al código del componente
);
```

Propósito
Obtiene el código de los componentes de un elemento (número 'ncomp') para cada tipo de descomposición.

Parámetros
h: Identificador (HANDLE) del concepto paramétrico.
tipo: Número del tipo de descomposición descrito en el campo TIPO_DESCOMPOSICION del registro ~R.
ncomp: Número del componente de la relación tipo del concepto. Se numeran empezando por cero.
Codigo: Puntero al código del componente.

Valor devuelto
Devuelve '0' si se ejecuta correctamente. Para obtener más INFORMACION sobre el error producido, llame a la función BdcError().

```
LONG EXPORTA BdcNumPropComponente (
        HANDLE h,           // identificador del concepto
        LONG tipo,          // tipo de descomposición
        LONG ncomp          // número del componente
);
```

Propósito
Obtiene el número de propiedades del componente para cada tipo de descomposición.

Parámetros
h: Identificador (HANDLE) del concepto paramétrico.
tipo: Número del tipo de descomposición descrito en el campo TIPO_DESCOMPOSICION del registro ~R.
ncomp: Número del componente de la relación tipo del concepto. Se numeran empezando por cero.

Valor devuelto
Número de propiedades de la relación.

```
BOOL EXPORTA BdcComponentePropValString (
        HANDLE h,           // identificador del concepto
        LONG tipo,          // tipo de descomposición
```

```
        LONG ncomp,          // número de componente
        LONG nprop,          // número de la propiedad
        LPCSTR *propiedad,   // puntero al nombre de la propiedad
        LPCSTR *valor,       // puntero al valor de la propiedad
        LPCSTR *um           // puntero a la unidad de medida de la propiedad
);
```

Propósito
Obtiene el nombre de la propiedad ‘nprop’, el valor y la unidad de medida.

Parámetros
h: Identificador (HANDLE) del concepto paramétrico.
tipo: Número del tipo de descomposición descrito en el campo TIPO_DESCOMPOSICION del registro ~R.
ncomp: Número del componente de la relación tipo del concepto. Se numeran empezando por cero.
nprop: Número de propiedad. Se numeran empezando por cero.
propiedad: Puntero al nombre de la propiedad de la relación.
valor: Puntero al valor de la propiedad que puede ser alfabético o numérico.
um: Puntero a la unidad de medida de la propiedad. En el caso que los valores de la propiedad sean valores numéricos se indicará de acuerdo con el Sistema Internacional de Unidades de Medida (ver anexo 7).

Valor devuelto
Devuelve ‘0’ si se ejecuta correctamente. Para obtener más INFORMACION sobre el error producido, llame a la función BdcError().

4. MENSAJES DE ERROR.

Los CODIGOs de error se almacenan en un LONG (entero de 32 bits) de manera que cada error corresponde a un bit. De esta forma, es posible definir hasta 32 CODIGOs de error que pueden producirse de forma aislada o conjunta. Las llamadas a las funciones BdcGloError() y BdcError() eliminan los CODIGOs de error producidos anteriormente.

Por ejemplo, para saber si un determinado error se ha producido con el concepto 'Concepto', se debe utilizar la sintaxis:

```
const char *Mensaje;
LONG cod_err = BdcError ((HANDLE)Concepto, &Mensaje);
if (cod_err & BDCERR_BASE_DATOS) {
        // Se ha producido el error 'BDCERR_BASE_DATOS'
        ...
}
```

4.1. CODIGOs de los mensajes de error.

BDCERR_CORRECTO	No hay error.
BDCERR_BASE_DATOS	Existe un mensaje de error. Es el caso en el que la DEFINICION paramétrica se indica una combinación inválida y se devuelve un mensaje de error explicativo.
BDCERR_PARAMETRO	Se pasó a BdcCalcula o BdcGloCalcula un parámetro inexistente.
BDCERR_OPCION	Se pasó a BdcCalcula o BdcGloCalcula una opción inexistente.
BDCERR_MAX_OPCIONES	Se definieron más de 62 opciones en un determinado parámetro.
BDCERR_NO_LEIDO	Se intentó utilizar BdcCalcula() antes que BdcLee().
BDCERR_NO_CALCULADO	Se intentó acceder a datos de un derivado paramétrico antes de utilizar BdcCalcula().
BDCERR_DESCOMPOSICION	Se intentó acceder a un elemento de la descomposición inexistente.
BDCERR_SIN_CODIGO	No existe CODIGO definido.
BDCERR_SIN_MEMORIA	Memoria insuficiente.
BDCERR_CONCEPTO_NULO	Se pasó un HANDLE nulo.
BDCERR_FMT_NO_SOPORTADO	El formato de texto solicitado no está soportado por la BDC
BDCERR_NO_COMBINACION:	A la llamada BdcValida o BdcCalcula se le pasó una combinación inexistente.
BDCERR_NO_BANCO_PRECIOS:	No existe ningún Banco de Precios.

BDCERR_PARAMETRO_INCORRECTO "El programa pidió un parámetro incorrecto"

EJEMPLOS.

1. BASE DE DATOS EJEMPLO: BASE.DLL

En la carpeta Anexo3 que acompaña a este documento (y que pueden encontrarse en la página web de la asociación, 'http://www.fiebdc.es'), se suministran tanto los archivos que constituyen la base como los fuentes necesarios para construirla (compilarla).

El ejemplo construido posee las siguientes características (no todas las bases que se realicen conforme a este formato deben tener estas mismas características):

1. Se ha utilizado un modelo de codificación independiente de los parámetros.
2. Se ha utilizado el modelo uno de textos de pliegos.
3. Existe un paramétrico global, si bien el valor de su único parámetro sólo es utilizado en el concepto 'SBRG.1$'.
4. Existe un concepto ('Esp$') que no responde a los criterios de FIEBDC-3/95: posee más de cuatro parámetros y el código de la familia sólo tiene 4 caracteres.
5. Todos los conceptos paramétricos poseen un CODIGO terminado en '$', si bien el formato no lo exige.
6. Existe un concepto ('SBRG.1$') cuyos derivados poseen un código que no responde a los criterios de FIEBDC-3/95. Para la asignación de código se ha seguido la técnica de definir una tabla de sinónimos: así, el concepto que resulta de escoger el primer valor del único parámetro que posee, en lugar de poseer el código 'SBRG.1a' posee el código 'SBRG.1_18'.
7. Los textos de los pliegos sólo están definidos en formato ASCII y para los derivados paramétricos. No están divididos ni en secciones (facetas) ni en ámbitos.

1.1. Archivos para la distribución de la Base de Datos.

Para distribuir la base datos que se construyera con este ejemplo, se deberían proporcionar los siguientes archivos:

base.dll: DLL que contiene las descripciones paramétricas y el interfaz ESTANDAR con aplicaciones (API).

base.bc3: Archivo ASCII de la base de datos en formato FIEBDC-3/98. En el ejemplo, la base incluye en esta DLL el paramétrico global de la base, así como las descripciones paramétricas de los conceptos "ABPH.1$", "SBRG.1$", "EADR.3$" y "Esp$", por lo que al menos debe contar con los siguientes registros:

```
~P| | | BASE.DLL |
~P| ABPH.1$ | |
~P| SBRG.1$ | |
~P| EADR.3$ | |
~P| Esp$ | |
```

1.2. Archivos necesarios para la CONSTRUCCION de la Base de Datos.

Este ejemplo está preparado para compilarse con Microsoft Visual Studio 2010 juego de caracteres multibyte como DLL de 32 y 64 bits.

Para construir la DLL, son necesarios los siguientes archivos:

fiebdc.h: Archivo que define el formato.

base.h: DEFINICION de variables y defines útiles para la definición de las descripciones paramétricas.

interfaz.cpp: Implementación de las funciones del API.

aplicat.cpp: Implementación de las funciones de la descripción paramétrica.

base.cpp: Implementación de las descripciones paramétricas de la base de datos en formato C++. Es la única parte que escribirían los redactores de las bases de datos. Se ha utilizado una sintaxis similar a la descripción paramétrica del formato FIEBDC-3/95, para facilitar así el intercambio entre ambos formatos.
base.def: DEFINICION de las funciones de exportación del API.

2. APLICACION EJEMPLO: PROGRAMA.EXE

En la carpeta Anexo3 que acompaña a este documento (y que puede encontrarse en la página web de la asociación, 'http://www.fiebdc.es'), se suministran tanto el ejecutable ('Programa.EXE') como los fuentes necesarios para construirlo (compilarlo).

Esta sencilla aplicación, lee los conceptos que se incluyen en el propio fuente de la base BASE.DLL (en una aplicación real, los conceptos se definen en los registros ~P de base.bc3), y escribe en el archivo 'SALIDA.TXT' los rótulos de todos sus parámetros, así como todos los datos de todas sus combinaciones paramétricas.

Los fuentes son fácilmente modificables para poder probar con el programa cualquier base de datos que utilice paramétricos compilados en DLL de acuerdo con el API establecido en este documento.

2.1. Archivos necesarios para la CONSTRUCCION del programa ejemplo.

Este ejemplo está preparado para compilarse con Microsoft Visual Studio 2010 juego de caracteres multibyte como aplicación Windows de 32 y 64 bits en modo consola.

Para construir la aplicación, son necesarios los siguientes archivos:

fiebdc.h: Archivo que define el formato.
program.h: Declaración de variables y funciones.
program0.c: Funciones auxiliares de apertura y cierre de la base de datos y tratamiento de los mensajes de error.
program.c: Programa ejemplo.

Anexo 4. Clasificación en tipos de los Conceptos.

1) Tipos 0, 1, 2, 3, 4 y 5.

Estos tipos corresponden a los vigentes en el formato FIEBDC-3/2016.

0	Sin clasificar
1	Mano de obra
2	Maquinaria
3	Materiales
4	Componentes adicionales de residuo
5	Clasificación de residuo

2) Tipos obtenidos de los índices y fórmulas polinómicas de revisión de precios del BOE según índices oficiales del RD 1359/2011.

Esta clasificación proviene del BOE (Boletín Oficial del Estado: www.boe.es), según índices oficiales del Real Decreto 1359/2011 de aplicación para contratos de obras públicas adjudicadas a partir de 1 de agosto de 2013 para obras para la administración pública, para fijar los índices de variación de precios mediante sus respectivas fórmulas polinómicas.

A	Aluminio
B	Materiales bituminosos
C	Cemento
E	Energía
F	Focos y luminarias
L	Materiales cerámicos
M	Madera
O	Plantas
P	Productos plásticos
Q	Productos químicos
R	Áridos y rocas
S	Materiales siderúrgicos
T	Materiales electrónicos
U	Cobre
V	Vidrio
X	Materiales explosivos

A los anteriores se añaden los siguientes:

1	Mano de obra
2	Maquinaria
3	Otros materiales
%	Medios auxiliares

donde,

1	Se refiere al tipo 1 de la clasificación “Tipos 0, 1, 2, 3, 4 y 5”.
2	Se refiere al tipo 2 de la clasificación “Tipos 0, 1, 2, 3, 4 y 5”.
A, B, C, E, F, L, M, O, P, Q, R, S, T, U, V, X y 3,	se refieren al tipo 3 de la clasificación “Tipos 0, 1, 2, 3, 4 y 5”. Dicha clasificación de materiales se corresponde con la de los índices de revisión de precios de las obras oficiales, a los que se añade el tipo M para aquellos materiales que no se puedan asociar a ninguno de los anteriores.
%	Se refiere a un nuevo tipo que se corresponde con los medios auxiliares que pueden aparecer en una justificación de precios.

3) Tipos obtenidos de los índices y fórmulas polinómicas de revisión de precios del BOE según índices oficiales anteriores al RD 1359/2011 y de la de la CNC.

Esta clasificación proviene del BOE (Boletín Oficial del Estado: www.boe.es) según índices oficiales del Real Decreto 1359/2011 de aplicación para contratos de obras públicas adjudicadas hasta 31 de julio de 2013 para obras para la administración pública, y de la CNC (Confederación Nacional de la Construcción: www.cnc.es), que la utilizan para fijar los índices de variación de precios mediante sus respectivas fórmulas polinómicas.

H	Mano de obra
MC	Cemento
MCr	Cerámicas
MM	Maderas
MS	Siderúrgicos
ME	Energía
Mcu	Cobre
Mal	Aluminio
ML	Ligantes

A los anteriores se añaden los siguientes:

M	Otros materiales
Q	Maquinaria
%	Medios auxiliares

donde,

H Se refiere al tipo 1 de la clasificación “Tipos 0, 1, 2, 3, 4 y 5”.

MC, MCr, MM, MS, ME, MCu, MAl, ML y M, se refieren al tipo 3 de la clasificación “Tipos 0, 1, 2, 3, 4 y 5”. Dicha clasificación de materiales se corresponde con la de los índices de revisión de precios de las obras oficiales, a los que se añade el tipo M para aquellos materiales que no se puedan asociar a ninguno de los anteriores.

Q Se refiere al tipo 2 de la clasificación “Tipos 0, 1, 2, 3, 4 y 5”.

% Se refiere a un nuevo tipo que se corresponde con los medios auxiliares que pueden aparecer en una justificación de precios.

4) Tipos obtenidos de la Asociación de Redactores de Bases de Datos de la Construcción.

Estos tipos sirven para desarrollar el tipo 0 de la clasificación “Tipos 0, 1, 2, 3, 4 y 5”.

EA	Elemento auxiliar
EU	Elemento unitario
EC	Elemento complejo
EF	Elemento funcional
OB	Obra
PA	Partida Alzada
PU	Presupuesto Unitario

donde,

Elemento auxiliar: Elemento constructivo formado por una combinación de elementos básicos (mano de obra, materiales y maquinaria) que intervienen en la formación de una unidad de obra.
Ejemplos: Hormigón H-250 de consistencia plástica elaborado en obra; Acero AEH-400 S elaborado en obra; etc.

Elemento unitario: Elemento constructivo formado por un conjunto de elementos básicos y/o

auxiliares que configuran una unidad de obra y que lo realiza un mismo grupo de especialistas.
Ejemplos: Tabique de ladrillos cerámicos huecos sencillos de 25x12x4 cm, tomados con mortero de cemento de dosificación 1:6; Hormigón H-250 en soportes; etc.

Elemento complejo: Elemento constructivo formado por un conjunto de elementos básicos, auxiliares y/o unitarios que constituye un conjunto constructivo y que lo realiza uno o varios grupos de especialistas.
Ejemplos: Hormigón H-250 armado con 120 kg de acero AEH-400 S y encofrado con placas metálicas, en soportes…; Cerramiento de fachada formado por dos hojas cerámicas con aislamiento en cámara de aire…; etc.

Elemento funcional: Elemento constructivo formado por un conjunto de elementos básicos, auxiliares, unitarios y/o complejos que constituye un conjunto constructivo con una función completa dentro de la obra.
Ejemplos: Estructura de hormigón armado; Cocina formada por…; etc.

Obra: Elemento constructivo formado por un conjunto de elementos funcionales, complejos, unitarios, auxiliares y/o simples que configuran la totalidad de elementos que constituyen una construcción.
Ejemplos: Rehabilitación de fachada…; Edificio de viviendas plurifamiliar…; etc.

Partida Alzada: Unidad de obra a justificar.
Ejemplo: Partida alzada a justificar del 1% del PEM, para gastos de acción cultural.

Presupuesto Unitario: Concepto que se refiere a un presupuesto parcial que se compone de unidades de obra. Se asimila a Elemento Funcional (EF) pero un comportamiento diferente en su aplicación. Su comportamiento es que puede aparecer como una unidad de obra en un presupuesto a la que se le asigna una medición; y en cambio se comporta como un subcapítulo de presupuesto en cuanto a que no se le han de aplicar costes indirectos (los costes indirectos, en su caso, ya los contemplan las unidades de obra de su composición) y no aparece en los cuadros de precios 1 y 2.

Anexo 5. Ámbitos Territoriales.

Como abreviaturas de Ámbitos Territoriales correspondientes a provincias y CCAA del estado español, para su posible uso el campo ROTULO_IDENTIFICACION del registro ~V y/o el campo ABREV_AMBITO del registro ~W, se establecen las siguientes:

E	España		
	AND		Comunidad Autónoma de Andalucía
		AL	Almería
		CO	Córdoba
		H	Huelva
		CA	Cádiz
		GR	Granada
		J	Jaén
		MA	Málaga
		SE	Sevilla
	ARA		Comunidad Autónoma de Aragón
		TE	Teruel
		HU	Huesca
		Z	Zaragoza
	AST		Comunidad Autónoma del Principado de Asturias
		O	Asturias
	BAL		Comunidad Autónoma de las Islas Baleares
		PM	Baleares
	CAN		Comunidad Autónoma de Canarias
		GC	Las Palmas
		TF	Tenerife
	CBR		Comunidad Autónoma de Cantabria
		S	Cantabria
	CLM		Comunidad Autónoma de Castilla-La Mancha
		AB	Albacete
		CR	Ciudad Real
		CU	Cuenca
		GU	Guadalajara
		TO	Toledo
	CAL		Comunidad Autónoma de Castilla y León
		AV	Ávila
		SG	Segovia
		SO	Soria
		VA	Valladolid
		BU	Burgos
		LE	León
		P	Palencia
		SA	Salamanca
		ZA	Zamora
	CAT		Comunidad Autónoma de Cataluña
		B	Barcelona
		GI	Girona
		T	Tarragona
		L	Lleida
	EXT		Comunidad Autónoma de Extremadura
		BA	Badajoz
		CC	Cáceres
	GAL		Comunidad Autónoma de Galicia
		LU	Lugo
		OR	Ourense
		PO	Pontevedra
		C	A Coruña

MAD		Comunidad de Madrid
	M	Madrid
MUR		Comunidad Autónoma de la Región de Murcia
	MU	Murcia
NAV		Comunidad Foral de Navarra
	NA	Navarra
PVA		Comunidad Autónoma del País Vasco
	VI	Álava
	BI	Bizkaia
	SS	Guipuzkoa
RIO		Comunidad Autónoma de La Rioja
	LO	La Rioja
VAL		Comunidad Valenciana
	V	Valencia
	A	Alicante
	CS	Castellón

Anexo 6. Divisas.

Como abreviaturas de las Divisas (campo DIVISA del registro ~K), se establecen las especificadas por la ISO 4217. Se adjuntan algunas a modo de ejemplo:

Unidad Monetaria	**Unión Monetaria Europea**
ATS	Chelín Austriaco
BEF	Franco Belga
DEM	Marco Alemán
ESP	Peseta Española
FIM	Marco Finlandés
FRF	Franco Francés
GRD	Dracma Griega
IEP	Libra Irlandesa
ITL	Lira Italiana
LUF	Franco Luxemburgués
NLG	Florín Neerlandés
PTE	Escudo Portugués

Moneda no UME	**Resto**
AUD	Dólar Australiano
BGN	Lev Búlgaro
CAD	Dólar Canadiense
CHF	Franco Suizo
CYP	Libra Chipriota
CZK	Corona Checa
DKK	Corona Danesa
EEK	Corona Estona
EUR	Euro
GBP	Libra Esterlina
HKD	Dólar de Hong-Kong
HUF	Forint Húngaro
ISK	Corona Islandesa
JPY	Yen Japonés
KRW	Won Surcoreano
LTL	Litas Lituano
LVL	Lats Letón
MTL	Lira Maltesa
NOK	Corona Noruega
NZD	Dólar Neozelandés
PLN	Zloty Polaco
ROL	Leu Rumano
SEK	Corona Sueca
SGD	Dólar de Singapur
SIT	Tolar Esloveno
SKK	Corona Eslovaca
TRL	Lira Turca
USD	Dólar Estadounidense
ZAR	Rand Sudafricano

Anexo 7. Unidades de Medida.

"El Sistema Legal de Unidades de Medida obligatorio en España es el sistema métrico decimal de siete unidades básicas, denominado Sistema Internacional de Unidades (SI), adoptado por la Confederación General de Pesas y Medidas y vigente en la Comunidad Económica Europea". RD 2032/2009, de 30 de diciembre, por el que se establecen las unidades legales de medida.

Según este real decreto se acuerda adoptar la nomenclatura de las unidades siguientes:

m	Metro
m2	Metro cuadrado
m3	Metro cúbico
kg	Kilogramo
km	Kilómetro
t	Tonelada
l	Litro
h	Hora
d	Día
a	Área
ha	Hectárea
cm3	Centímetro cúbico
cm2	Centímetro cuadrado
dm3	Decímetro cúbico

Por similitud con dicho real decreto, y provenientes de la Asociación de Redactores de Bases de Datos de la Construcción, también se acuerda adoptar las siguientes:

u	Unidad
mu	Mil unidades
cu	Cien unidades
mes	Mes

Anexo 8. Definición de diferentes tipos de Presupuestos.

Se denomina PRESUPUESTO al sumatorio de las cantidades obtenidas en la medición de las unidades de obra por sus respectivos precios organizados según una estructura (ejemplo: raíz, capítulos, subcapítulos y partidas).

PRESUPUESTO DE EJECUCION MATERIAL (PEM).

Artículo 131 del Reglamento General de la Ley de Contratos de las Administraciones Públicas:

"Se denominará presupuesto de ejecución material el resultado obtenido por la suma de los productos del número de cada unidad de obra por su precio unitario y de las partidas alzadas."

Artículo 130. Cálculo de los precios de las distintas unidades de obra:

"1. El cálculo de los precios de las distintas unidades de obra se basará en la determinación de los costes directos e indirectos precisos para su ejecución, sin incorporar, en ningún caso, el importe del Impuesto sobre el Valor Añadido que pueda gravar las entregas de bienes o prestaciones de servicios realizados.

2. Se considerarán costes directos:
a) La mano de obra que interviene directamente en la ejecución de la unidad de obra.
b) Los materiales, a los precios resultantes a pie de obra, que quedan integrados en la unidad de que se trate o que sean necesarios para su ejecución.
c) Los gastos de personal, combustible, energía, etc. que tengan lugar por el accionamiento o funcionamiento de la maquinaria e instalaciones utilizadas en la ejecución de la unidad de obra.
d) Los gastos de amortización y conservación de la maquinaria e instalaciones anteriormente citadas.

3. Se considerarán costes indirectos:
Los gastos de instalación de oficinas a pie de obra, comunicaciones, edificación de almacenes, talleres, pabellones temporales para obreros, laboratorio, etc., los del personal técnico y administrativo adscrito exclusivamente a la obra y los imprevistos. Todos estos gastos, excepto aquéllos que se reflejen en el presupuesto valorados en unidades de obra o en partidas alzadas, se cifrarán en un porcentaje de los costes directos, igual para todas las unidades de obra, que adoptará, en cada caso, el autor del proyecto a la vista de la naturaleza de la obra proyectada, de la importancia de su presupuesto y de su previsible plazo de ejecución.

4. En aquellos casos en que oscilaciones de los precios imprevistas y ulteriores a la aprobación de los proyectos resten actualidad a los cálculos de precios que figuran en sus presupuestos podrán los órganos de contratación, si la obra merece el calificativo de urgente, proceder a su actualización aplicando un porcentaje lineal de aumento, al objeto de ajustar los expresados precios a los vigentes en el mercado al tiempo de la licitación.

5. Los órganos de contratación dictarán las instrucciones complementarias de aplicación al cálculo de los precios unitarios en los distintos proyectos elaborados por sus servicios."

Artículo 154. Partidas alzadas:

"1. Las partidas alzadas se valorarán conforme se indique en el pliego de prescripciones técnicas particulares. En su defecto se considerarán:

a) Como partidas alzadas a justificar, las susceptibles de ser medidas en todas sus partes en unidades de obra, con precios unitarios, y
b) Como partidas alzadas de abono íntegro, aquéllas que se refieren a trabajos cuya especificación figure en los documentos contractuales del proyecto y no sean susceptibles de medición según el pliego.

2. Las partidas alzadas a justificar se valorarán a los precios de la adjudicación con arreglo a las condiciones del contrato y al resultado de las mediciones correspondientes. Cuando los precios de una o varias unidades de obra no figuren incluidos en los cuadros de precios, se procederá conforme a lo dispuesto en el artículo 146.2 de la Ley, en cuyo caso, para la introducción de los nuevos precios así determinados habrán de cumplirse conjuntamente las dos condiciones siguientes:
a) Que el órgano de contratación haya aprobado, además de los nuevos precios, la justificación y descomposición del presupuesto de la partida alzada, y
b) Que el importe total de dicha partida alzada, teniendo en cuenta en su valoración tanto los precios incluidos en los cuadros de precios como los nuevos precios de aplicación, no exceda del importe de esta figurado en el proyecto.

3. Las partidas alzadas de abono íntegro se abonarán al contratista en su totalidad, una vez determinados los trabajos u obras a que se refieran, de acuerdo con las condiciones del contrato y sin perjuicio de lo que el pliego de cláusulas administrativas particulares pueda establecer respecto de su abono fraccionado en casos justificados.
Cuando la especificación de los trabajos u obras constitutivos de una partida alzada de abono íntegro no figure en los documentos contractuales del proyecto o figure de modo incompleto, impreciso o insuficiente a los fines de su ejecución, se estará a las instrucciones que a tales efectos dicte por escrito la dirección, a las que podrá oponerse el contratista en caso de disconformidad."

PRESUPUESTO BASE DE LICITACIÓN
(anteriormente denominado Presupuesto de ejecución por contrata PEC).

Artículo 131 del Reglamento General de la Ley de Contratos de las Administraciones Públicas:

"El presupuesto base de licitación se obtendrá incrementando el de ejecución material en los siguientes conceptos:

1. Gastos generales de estructura que inciden sobre el contrato, cifrados en los siguientes porcentajes aplicados sobre el presupuesto de ejecución material:

a) Del 13 al 17 por 100, a fijar por cada Departamento ministerial, a la vista de las circunstancias concurrentes, en concepto de gastos generales de la empresa, gastos financieros, cargas fiscales, Impuesto sobre el Valor Añadido excluido, tasas de la Administración legalmente establecidas, que inciden sobre el costo de las obras y demás derivados de las obligaciones del contrato. Se excluirán asimismo los impuestos que graven la renta de las personas físicas o jurídicas.

b) El 6 por 100 en concepto de beneficio industrial del contratista.

Estos porcentajes podrán ser modificados con carácter general por acuerdo de la Comisión Delegada del Gobierno para Asuntos Económicos cuando por variación de los supuestos actuales se considere necesario.

2. El Impuesto sobre el Valor Añadido que grave la ejecución de la obra, cuyo tipo se aplicará sobre la suma del presupuesto de ejecución material y los gastos generales de estructura reseñados en el apartado 1."

El Impuesto sobre el Valor Añadido puede variar dependiendo de la zona geográfica, en cuyo caso se aplicará el que corresponda, por ejemplo, Impuesto General Indirecto Canario (IGIC).

En obras privadas los porcentajes de gastos generales de la empresa y beneficio industrial del contratista pueden ser distintos de los especificados anteriormente.

PRESUPUESTO DE LICITACION (PL).

Importe que sirve de base para formular las ofertas económicas de las empresas que concurren a una licitación de obra.

PRESUPUESTO DE ADJUDICACION (PAD).

Se denominará así al importe correspondiente a la proposición económica de la empresa que resulta adjudicataria de la obra. Puede ser coincidente con el Presupuesto de Licitación, inferior a este si se produce baja, o superior a este si se produce alza.

Se recomienda que el alza o la baja se apliquen sobre el Presupuesto de Ejecución Material, puesto que la cantidad resultante será la misma y sin embargo no se reflejarán modificaciones en la imputación del IVA.

Anexo 9. Criterios para la asignación de referencias en IFC a bancos de precios en formato FIEBDC.

En los últimos años, está siendo cada vez más habitual la asignación de códigos a los elementos de los modelos BIM, para hacer referencia a conceptos existentes en bases de precios en formato BC3.

Con el objeto de estandarizar el uso de estos códigos, la Comisión Técnica Permanente de la Asociación FIEBDC propone a los usuarios de programas BIM el uso de estos criterios:

- Incluir en cada elemento BIM un parámetro o atributo de tipo texto, de nombre “BC3”. En este parámetro, se introducirá el código del concepto de la base de precios, de hasta 20 caracteres. En caso de que se necesite asignar varios conceptos de la base a un solo elemento BIM, pueden introducirse todos ellos en este parámetro, separados por comas. En el caso de varios códigos, el orden de los códigos para elementos verticales será el de considerar de exterior a interior y para elementos horizontales el de arriba a abajo.

- Opcionalmente, puede incluirse también el parámetro, de nombre “BC3_URL” con la dirección URL de acceso a la información de ese concepto de la base de precios, en formato BC3. En caso de ser varios conceptos, podrán incluirse varias direcciones URL separadas por comas. En el caso que alguna de ellas no esté disponible se dejará en blanco y solo separada por las comas.

- Opcionalmente, puede incluirse también el parámetro “BC3_BASE” con el nombre de la base de precios utilizada, sí como su año o versión. En caso de ser varios conceptos, podrán incluirse varias bases separades por comas. En el caso que alguna de ellas no esté disponible se dejará en blanco y solo separada por las comas.

Por otra parte, en el caso de que se desee vincular el modelo BIM con un presupuesto en formato BC3, puede incluirse otro parámetro global, de nombre “BC3_PRESUPUESTO_URL” que incluya la dirección URL en la que se encuentre el archivo del presupuesto en formato BC3.

Anexo 10. Enriquecimiento de modelos BIM con datos del presupuesto en formato BC3.

En los últimos años, está siendo cada vez más habitual la asignación de códigos a los elementos de los modelos BIM, para hacer referencia a conceptos existentes en bases de precios en formato BC3.

Si se va a añadir información sobre el presupuesto en un modelo BIM, se recomienda añadir en este un único PropertySet de nombre “BC3” con los siguientes campos:

Nombre	Dato a incluir	Ejemplo
BC3_BASE	Base de precios	Precio Centro
BC3_FECHA	Fecha	1/03/20
BC3_URL	URL del precio	www.base.com/2020/EHH25
BC3_CODIGO	Código del concepto	EHH25
BC3_UNIDAD_DE_MEDIDA	Unidad de medida	m3
BC3_TEXTO_RESUMEN	Texto resumen	Hormigón HA25
BC3_TEXTO_DESCRIPCION	Texto de descripción	Hormigón HA-25/B/20/I central
BC3_PRECIO	Precio unitario	68,00
BC3_MEDICION	Total de medición	0,27
BC3_IMPORTE	Importe	18,36
BC3_MONEDA	Unidad monetaria	EUR

Si hay varios conceptos asociados a un único elemento del modelo BIM, se añadirá 02, 03, etc. Por ejemplo, “BC3_CODIGO02”, “BC3_PRECIO02”, etc.

Anexo 11. API DE CONEXIÓN REMOTA de bases de datos para el formato FIEBDC.

Objetivo

Definir un API de conexión remota a bases de datos (a partir de ahora API) que permita a los proveedores de bases de datos de la construcción exponer su información para ser utilizada por programas propios o de terceros.

El acceso a esa información se realizará mediante puntos de acceso (a partir de ahora endpoints) definidos en el API. La especificación define como llamar y obtener la información de una lista de códigos de elementos utilizando dichos endpoints.

Dicho acceso se realiza mediante un endpoint que, dada una base de datos, la selección de parámetros de banco y la lista de elementos devuelve su información.

Adicionalmente se necesitarán otros endpoints auxiliares que se detallan también en el apartado endpoints.

1. Tipo de API y formato

Utilizaremos un API rest con formato json para realizar las llamadas a los endpoints. La información relacionada con los elementos de los cuales se desea obtener información se expresa en el actual formato BC3, manteniendo de esta manera la compatibilidad con los programas que actualmente están consumiendo este tipo de información.

Los tipos de datos utilizados en el formato json se representarán de la siguiente manera:

Tipo de dato	Formato
Numérico	No se utiliza el separador de miles. En caso de haber decimales el separador decimal utilizado es '.' Ejemplo: 124896.08
Booleano	true / false
Fecha	Expresada según ISO 8601. Ejemplo: 2012-04-23T18:25:43.511Z o 2012-04-23
cadena de texto	Valor entre comillas dobles. Ejemplo: "ciudad". Utilizar \ para escapar caracteres especiales: \b Backspace (asciicode 08) \f Formfedd (asciicode 0C) \n New line \r Carriagereturn \t Tab \" doublequote \\ Backslash
Null	El valor nulo se representa como null.

Arrays	En el caso que un valor sea de tipo array no diferenciamos entre si el valor es nulo o es un array vacío. En ambos casos se representan como un array vacío [].
Divisas	Se presentan según ISO4217 utilizando la columna código definida en https://es.wikipedia.org/wiki/ISO_4217
Idioma	Seguir la codificación especificada en http://www.lingoes.net/en/translator/langcode.htm
Encoding	El juego de caracteres utilizado en el api es UTF8. A excepción del contenido en formato BC3 que se codifica en base64.

Esta definición de tipos no afecta a la información representada en formato BC3 el cual tiene su propia especificación.

En el caso que la representación del valor nulo como null sea un problema en aplicaciones desarrolladas en C++ debido a que los tipos de datos utilizados no admitan valores nulos, se recomienda aplicar la solución https://devblogs.microsoft.com/cppblog/stdoptional-how-when-and-why/ propuesta por Microsoft.

2. EndPoints

2.1 Obtener información las agrupaciones de bases de datos

Es posible que el proveedor de bases de datos disponga de más de una base de datos y que el usuario tenga que seleccionar de cual desea obtener la información.

Dependiendo del proveedor, las bases de datos pueden estar organizadas en agrupaciones de bases de datos, por ejemplo: construcción, mantenimiento, ambiental… y dentro de estas agrupaciones existan diferentes bases de datos cada una de las cuales puede tener diferentes versiones.

Las agrupaciones tienen un código y una descripción, que son determinados por el proveedor de base de datos y por tanto varían dependiendo del proveedor.

Otros proveedores no dispondrán de esta organización en agrupaciones, pero sí de diferentes bases de datos donde cada una puede tener diferentes versiones.

Supongamos las siguientes agrupaciones y bases de datos:

- BEDEC Construcción:
  - BEDEC Construcción 2022-04
  - BEDEC Construcción 2021-06
- BEDEC Sostenibilidad:
  - BEDEC Sostenibilidad 2022-01
  - BEDEC Sostenibilidad 2022-03
- BEDEC Hidráulica:
  - BEDEC Hidráulica 2022-02
  - BEDEC Hidráulica 2021-03

Si solicitamos bases de datos de "BEDEC Construcción" obtendremos: "BEDEC Construcción 2022-04", "BEDEC Construcción 2021-06".

Si solicitamos bases de datos de "BEDEC Construcción" y "BEDEC Sostenibilidad" obtendremos: "BEDEC Construcción 2022-04", "BEDEC Construcción 2021-06", "BEDEC Sostenibilidad 2022-01", "BEDEC Sostenibilidad 2022-03".

Hay una agrupación que siempre está presente en todos los proveedores de bases de datos y que agrupa a todas las bases de datos "Todas las bases de datos" si solicitamos bases de datos de esta agrupación el resultado sería: "BEDEC Construcción 2022-04", "BEDEC Construcción 2021-06", "BEDEC Sostenibilidad 2022-01", "BEDEC Sostenibilidad 2022-01", "BEDEC Sostenibilidad 2022-03", "BEDEC Hidráulica 2022-02", "BEDEC Hidráulica 2021-03".

2.1.1 Request información sobre las clasificaciones de bases de datos.

Url: .../getdatabasegroups
Verb: GET
Headers: authsecurity, x-api-version
Tipo de llamada: síncrona

2.1.2 Response con la información de las clasificaciones de bases de datos.

Response:

```
{
    "status": object status,
    "cultures":  [ object culture1,…, object cultureN]
    "databasegroups": [ database group1,…, database groupN ]
}
```

Atributo	Nullable	Descripción
Status	No	Información sobre si ha habido errores o ha funcionado correctamente.
cultures	No	Array con la lista de idiomas disponibles por el proveedor de bases de datos.
databasegroups	No	Array con la lista de clasificaciones de bases de datos que tiene disponibles ese proveedor.

Ejemplos:
En la carpeta Anexo11_ejemplos/ejemplos/getdatabasegroups/response podemos encontrar ejemplos.

2.2 Obtener información sobre las bases de datos

El objetivo de este endpoint es obtener la lista de las bases de datos disponibles para un usuario en un proveedor.

2.2.1 Request información sobre las bases de datos

Url: .../getdatabases
Verb: POST
Headers: authsecurity, x-api-version
Tipo de llamada: síncrona
Request:

```
{
    id_applications: [1,...]
}
```

Atributo	Nullable	Descripción
id_applications	No	Supongamos que tenemos las agrupaciones: • 1 – BEDEC Construcción • 2 – BEDEC Sostenibilidad • 3 – BEDEC Hidráulica • 0 – Todas las bases de datos.

Podemos solicitar información de una o más agrupaciones.
La lista de bases de datos retornadas estará formada por todas aquellas bases de datos que pertenecen a las agrupaciones solicitadas.

En caso de que el proveedor de base de datos no tenga la información organizada por agrupaciones, la solicitud sería:

Request:

```
{
    id_applications: [0]
}
```

En este caso se devolverán todas las bases de datos disponibles.

Ejemplos:
En la carpeta Anexo11_ejemplos/ejemplos/getdatabases/request podemos encontrar ejemplos de peticiones válidas y no válidas.

2.2.2 Response con la información de las bases de datos

Response:

```
{
    "status": object status,
    "databases": [ object extended database1,..,object extended databaseN ]
}
```

Atributo	Nullable	Descripción
status	No	Información sobre si ha habido errores o ha funcionado correctamente.
databases	No	Lista con la información sobre las bases de datos solicitadas.

Ejemplos:
En la carpeta Anexo11_ejemplos/ejemplos/getdatabases/response podemos encontrar ejemplos de respuesta válidas y no válidas.

2.3 Obtener parámetros del banco

Antes de realizar la petición de obtención de información, el usuario ha de determinar con que parámetros globales se realiza la petición. Para ello primero se han de recuperar cuales son los parámetros disponibles para ese banco.

En el caso que el banco no disponga de parámetros devolverá una lista vacía de parámetros.

2.3.1 Request parámetros del banco

Tipo de llamada: sincrona
Url: .../getglobalparameters
Verb: POST
Headers: authsecurity, x-api-version
Tipo de llamada: síncrona
Request:

```
{
    "id_language": 0,
    "id_application": 2500727,
    "version":"2022-04-09"
}
```

Atributo	Nullable	Descripción
id_language	No	0- castellano, 1-catalán, 3-gallego, 4-euskera, 5-ingles. Esta codificación no es fija, la determina el proveedor de base de datos, cuando devuelve los idiomas, ver object culture.
id_application	No	Identificador de la agrupación: 1 – BEDEC Construcción, 2 – BEDEC Sostenibilidad, 3 – BEDEC Hidráulica, 0 – Todas las agrupaciones.
version	No	Texto que identifica la base de datos.

Ejemplos:
En la carpeta Anexo11_ejemplos/ejemplos/parametrosbanco/request podemos encontrar ejemplos de solicitudes válidas y no válidas.

2.3.2 Response parámetros del banco

Response:

```
{
    "status":  object status,
    "globalparameters ": [ object global parameter1,…,object global parameterN]
}
```

Atributo	Nullable	Descripción
status	No	Información sobre si ha habido errores o ha funcionado

		correctamente.
globalparameters	No	Array con los parámetros del banco y sus correspondientes valores.

Los parámetros del banco se representan como un array de propiedades, cada propiedad (parámetro) tiene a su vez un array (lista) de valores que representan los posibles valores que puede tomar un parámetro global.
Tanto los parámetros como los valores que pueden tomar tienen un identificador numérico proporcionado por el banco.

Ejemplos:
En la carpeta Anexo11_ejemplos/ejemplos/parametrosbanco/response podemos encontrar ejemplos de respuestas válidas y no válidas.

2.4 Drag & Drop

El drag & Drop continúa funcionando de la misma manera que hasta ahora, enviando la información en formato BC3. Por tanto, no se necesita ningún endpoint.

2.5 Adaptar

El proceso de adaptar consiste en que un cliente solicita información sobre un conjunto de códigos de elementos del banco con la finalidad que se actualice su información en base a la información contenida en la base de datos.

2.5.1 Request de elementos a adaptar

La solicitud de obtención de información de los elementos es un proceso asíncrono. Este proceso asíncrono se implementa con tres endpoints síncronos:

- Solicitud de obtención de información de elementos: envía los elementos de los cuales queremos obtener la información y recibe un identificador de tarea.
- Solicitud del estatus de la tarea asíncrona: pregunta si la información solicitada ya está disponible.
- Solicitud para obtener la información de los elementos: pide que le devuelvan la información que ya está disponible.

2.5.2 Request de la solicitud de obtención de información de elementos

Url: .../adapt/async
Verb: POST
Headers: authsecurity, x-api-version
Tipo de llamada: síncrona
Request:

```
{
    "database": "BEDEC",
    "version": "2020-01-24",
    "globalparameters ":  [ object global parameter1,…,object global parameterN ],
    "elements": [code1,...,codeN]
}
```

Atributo	Nullable	Descripción
database	No	Nombre de la base de datos, por ejemplo: "BEDEC"
version	No	Versión de la base de datos, identifica la base de datos, ejemplo: "2020-10-09".
globalparameters	No	Array con la lista de parámetros de banco y los valores seleccionados para cada parámetro.
elements	No	Array con la lista de códigos de los elementos de los cuales deseamos obtener información. Ejemplo: ["B200-7834", "GEYH-0009"]

Es responsabilidad del proveedor de bases de datos, si así lo desea, establecer un sistema de cuotas que limite el número de elementos que un usuario puede solicitar durante un periodo de tiempo. Por ejemplo: 1000 elementos/h, 1000 elementos/día, 1000 elementos/semana…

En caso de que el proveedor de bases de datos tenga implementado un sistema de cuotas y un usuario lo haya superado, al hacer la llamada en el objecto object status encontrará un código de error indicando esta situación.

Ejemplos:
En la carpeta Anexo11_ejemplos/ejemplos/adapta/request/async se encuentra un ejemplo de petición.

2.5.3 Response de la solicitud de obtención de información de elementos.

Response:

```
{
    "status": object status,
    "data": object asynctask
}
```

Atributo	Nullable	Descripción
status	No	Información sobre si ha habido errores o ha funcionado correctamente.
data	No	Contiene el identificador que el proveedor de bases de datos ha asignado a esa tarea asíncrona.

Ejemplos:
En la carpeta Anexo11_ejemplos/ejemplos/adapta/response/async se encuentra un ejemplo de petición.

2.5.4 Request del estatus de la tarea asíncrona

Url: .../adapt/async/status/<id tarea asíncrona>
Verb: GET
Headers: authsecurity, x-api-version
Tipo de llamada: síncrona

Atributo	Nullable	Descripción
id tarea asíncrona	No	Identificador de la tarea que queremos consultar si ya ha acabado o continua pendiente.

Ejemplo:
.../adapt/async/status/15729

En la carpeta Anexo11_ejemplos/ejemplos/adapta/responseasync se encuentra un ejemplo de petición.

2.5.5 Response del estatus de la tarea asíncrona

```
{
    "status": object status,
    "data": object asyncstatus
}
```

Atributo	Nullable	Descripción
status	No	Información sobre si ha habido errores o ha funcionado correctamente.
data	No	Información sobre el estado en que se encuentra la tarea, si ha comenzado, si ha finalizado, si se encuentra en ejecución...

Ejemplos:
En la carpeta Anexo11_ejemplos/ejemplos/adapta/response/status se encuentra un ejemplo de petición.

2.5.6 Request para obtener la información de los elementos

Una vez la tarea asíncrona ha finalizado se solicita la información.

Url: .../adapt/async/get/<id tarea asíncrona>
Verb: GET
Headers: authsecurity, x-api-version
Tipo de llamada: síncrona

Atributo	Nullable	Descripción
id tarea asíncrona	No	Identificador de la tarea ya finalizada de la cual queremos obtener la información.

Ejemplo:

.../adapt/async/get/15729

2.5.7 Response con la información de los elementos a adaptar.

Response:

```
{
    "status": object status,
    "database": object database,
    "globalparameters":  [object global parameter1,…,object global parameterN],
    "payload": object payload
}
```

Atributo	Nullable	Descripción
Status	No	Información sobre si ha habido errores o ha funcionado correctamente.
Database	No	Nombre de la base de datos, por ejemplo: "BEDEC"
globalparameters	No	Array con la lista de parámetros de banco y los valores seleccionados para cada parámetro. Son los parámetros con los que se ha generado la información de los elementos solicitados.
payload	No	Objeto que contiene la información de los elementos solicitados en formato BC3.

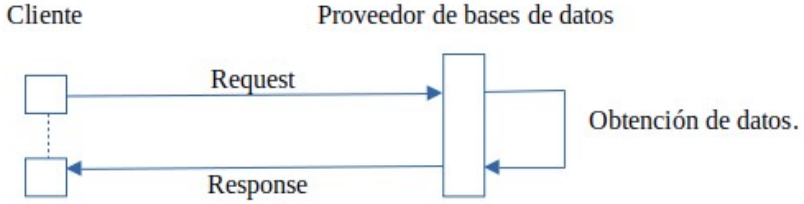
Ejemplos:
En la carpeta Anexo11_ejemplos/ejemplos/adapta/response/get podemos encontrar un ejemplo con la información de los elementos solicitados.

3. Llamadas síncronas versus asíncronas

Algunos endpoints requieren ser llamados de manera síncrona y otros de manera asíncrona. La diferencia entre un tipo y otro es la siguiente:

3.1 Síncrona

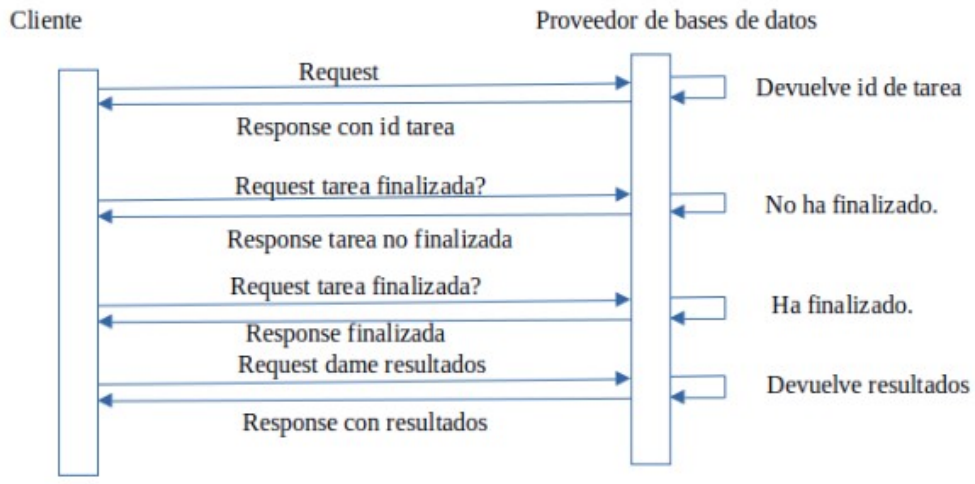
Una llamada síncrona es aquella que una vez realizada espera hasta que recibe respuesta del proveedor de bases de datos con la información solicitada y en ese momento finaliza. La espera puede ser de unos pocos segundos o inferior al segundo dependiendo del proveedor de bases de datos.



3.2 Asíncrona

A diferencia de las llamadas síncronas, las asíncronas no esperan hasta que reciben respuesta del proveedor de bases de datos, sino que la llamada finaliza sin que el proveedor haya devuelto la información solicitada.

Posteriormente, el cliente ha de ir preguntando periódicamente al proveedor de bases de datos si ya tiene disponible la información solicitada, hasta que éste contesta que ya está disponible y la obtiene.



4. Autentificación y autorización

Para realizar cualquier llamada a los endpoint es necesario presentar unas credenciales en forma de token, el token se envía en el header authsecurity. La manera de obtener el token varía dependiendo de los diferentes escenarios.

4.1 El usuario es la primera vez que solicita información sobre elementos.

Desde el programa cliente el usuario decide que quiere obtener información sobre ciertos elementos de un proveedor de base de datos.
El programa cliente se da cuenta que no tiene token para acceder al API del proveedor de base de datos, solicita al usuario que vaya a su proveedor de base de datos, consiga el token y lo introduzca en el programa cliente.
Una vez introducido el token es almacenado para ser usado en próximas ocasiones.

4.2 El usuario ya dispone de token

Desde el programa cliente el usuario decide que quiere obtener información sobre ciertos elementos de un proveedor de base de datos.
El programa cliente detecta que ya tienen un token para ese proveedor de base de datos y por lo tanto inicia el proceso de obtención de información.

4.3 El token ha caducado

Dependiendo del proveedor de base de datos los tokens pueden caducar.
Desde el programa cliente el usuario decide que quiere obtener información sobre ciertos elementos de un proveedor de base de datos. El programa cliente detecta que ya tiene token para el API de ese proveedor e inicia el proceso de obtención de información.
El proveedor de base de datos devuelve un error en las peticiones realizadas al API, indicando que el token ha caducado.
El programa cliente detecta esta situación, informa al usuario y volvemos al escenario 5.1 El usuario es la primera vez que solicita información sobre elementos. para conseguir un nuevo token.

4.4 Validación del token por el proveedor de bases de datos

Cuando el proveedor de base de datos recibe una petición obtiene el token del header authsecurity. Con este token el proveedor de bases de datos determina:

- Si el usuario tiene acceso. (Autentificación).
- A que información tiene acceso (Autorización).
- Si el token está caducado.

La información devuelta por cualquiera de los endpoint está condicionada a si el usuario tiene autorización a dicha información o no.

Dependiendo del proveedor de base de datos puede ocurrir que el token caduque y deje de ser valido para hacer peticiones. En este caso la respuesta de la petición indicara en su objeto status un error indicando que el token ha caducado

El token puede ser un JWT o cualquier otro tipo, el API es agnóstico respecto al tipo de token utilizado.

Los proveedores de bases de datos que exponen su información de manera abierta no necesitan token por tanto el header authsecurity estará vacío.

5. Comunicación cliente-servidor

La comunicación se realizará siempre por un canal seguro https, de manera que los datos viajen encriptados.

Cuando el volumen de datos devueltos sea muy elevado puede ocurrir que la comunicación se interrumpa. Esto es debido a las posibles restricciones que un proveedor de bases de datos pueda tener establecidas en sus servidores cortando todas las respuestas que superen un cierto tamaño.
Es responsabilidad de cada proveedor de bases de datos garantizar que la comunicación no se corte con el volumen de datos devuelto.

6. Versionado del API

A lo largo del tiempo es posible que se realicen cambios en la definición del API, dichos cambios generarán diferentes versiones. Dado que no todos los clientes migraran al mismo tiempo a las nuevas versiones, es necesario identificar que versión de API está utilizando el cliente, para ello en cada llamada a un endpoint en el header x-api-version se indica cual es la versión de API que se está utilizando.

Cada proveedor de base de datos determina durante cuánto tiempo da soporte a versiones antiguas del API.
El valor del header x-api-version es de la forma: v<número de version> ejemplo: v1, v2...

7. Validación del API

Dado un json será posible validar que cumple la especificación para ello se necesita:

- Un esquema json con la definición de la especificación.
- Un validador de json contra un esquema.
- El json que deseamos validar.

El esquema con la definición de la especificación se encuentra en el fichero Especificacion_ api.schema.json
Como validador de un json contra un esquema podemos utilizar https://www.jsonschemavalidator.net/ o cualquier otro que consideremos oportuno.

La posibilidad de validar los json garantiza que:

- Todos los proveedores de bases de datos devuelvan y acepten json que se ajustan a la especificación, evitando de esta manera variantes que no cumplen la especificación.
- Los clientes pueden verificar si están construyendo y recibiendo correctamente los json de sus peticiones y respuestas.

7.1 Como utilizar el validador jsonschemavalidator.net

Seguir los siguientes pasos:

- Navegar a https://www.jsonschemavalidator.net/
- En la parte izquierda selecciona en el desplegable la opción custom.
- Copiar en la parte izquierda el contenido del archivo "Especificacion_ api.schema.json"
- Indicar al esquema json que es lo que queremos validar:
  - Ir al final del esquema json copiado.
  - Reemplazar "replaceThisPlaceholder" por ejemplo con "endpointAdaptaRequest"
- En la parte derecha copiar el json que se desea validar, en este caso el json correspondiente a la request del endpoint Adapta.

En el siguiente punto se explica como reemplazar " replaceThisPlaceholder" en función del json que queramos validar

7.2 Validación de los ejemplos json

En la carpeta Anexo11_ejemplos/ejemplos se encuentran diferentes ejemplos json, explicaremos con que valor hemos de substituir " replaceThisPlaceholder" para realizar la validación.

Ejemplos	replaceThisPlaceholder	Descripción
adapta/request/async	endpointAdaptaAsyncRequest	Valida los json de la carpeta.
adapta/response/async	endpointAdaptaAsyncResponse	Valida los json de la carpeta.
adapta/response/status	endpointAdaptaAsyncStatusResponse	Valida los json de la carpeta.

adapta/response/get	endpointGetAdaptaResponse	Valida los json de la carpeta.
getdatabases/request	endpointGetDatabasesRequest	Valida los json de la carpeta.
getdatabases/response	endpointGetDabasesResponse	Valida los json de la carpeta.
parametrosbanco/request	endpointGlobalParametersRequest	Valida los json de la carpeta.
parametrosbanco/response	endpointGlobalParametersResponse	Valida los json de la carpeta.
getdatabasegroups/response	endpointGetDabasesGroupsResponse	Valida los json de la carpeta.

Además de validar los endopints también es posible validar el resto de los objetos que conforman la especificación:

Ejemplos	replaceThisPlaceholder	Descripción
culture	culture	Valida los json de la carpeta.
databases/regular database	database	Valida los json de la carpeta.
databases/extendend database	extendedDatabase	Valida los json de la carpeta.
parametrosbanco	globalParameterProperty	Valida los json de la carpeta. No los de las carpetas hijas.
payload	bc3PayLoad	Valida los json de la carpeta.
status	status	Valida los json de la carpeta.
valor	value	Valida los json de la carpeta.
databasegroups	databaseGroup	Valida los json de la carpeta.
asynctask	asyncTask	Valida los json de la carpeta.
asyncstatus	asyncStatus	Valida los json de la carpeta.

8. Object Database Group

Representa cada una de las agrupaciones de bases de datos que tiene disponible el proveedor de bases de datos

```
{
  "id": "24"
```

"description": object value,
}

Atributo	Nullable	Descripción
id	No	Identificador numérico de la agrupación, los identificadores son asignados por el proveedor de bases de datos. El identificador 0 está reservado, todos los proveedores de bases de datos lo utilizan para indicar una agrupación que contiene todas las bases de datos.
description	No	Valor multiidioma que contiene el nombre de las agrupaciones en diferentes idiomas. Por ejemplo: "BEDEC Construcción y ambiental", "BEDEC Construcció i ambiental". En object value se puede ver cómo se representa un valor multiidioma.

9. Object Extended Database

Este objeto se devuelve con Response con la información de las bases de datos.

{
"name": "BEDEC"
"version": "2020-01-14",
"description": object value,
"adaptblocksize": 300,
"id_application": 2500727,
"cultures": [object culture1,…, object cultureN]
}

Atributo	Nullable	Descripción
Name	No	Nombre de la base de datos, por ejemplo: "BEDEC"
Versión	No	Versión de la base de datos, ejemplo: "2020-10-09". Identifica la base de datos.
description	No	Valor multiidioma que contiene la descripción de las bases de datos en los idiomas soportados por el proveedor de base datos. Por ejemplo: "BEDEC proyecto y obra 2020-10-09", "BEDEC proyecto y obra 2020-10-09". En object value se puede ver cómo se representa un valor multiidioma.
id_application	No	Identificador del grupo de base de datos al que pertenece la base de datos.
adaptblocksize	No	La base de datos puede limitar el número de elementos máximo que un cliente puede realizar en cada petición. Ejemplo: Si la base de datos limita el número de elementos a 300 y

		deseamos pedir 3000 elementos tendremos que realizar 10 peticiones.
cultures	No	Array con la lista de idiomas disponibles por el proveedor de bases de datos.

10. Object Culture

Define cada uno de los idiomas disponibles por el proveedor de base de datos.

```
{
    "culture": "es-ES",
    "index": "0"
}
```

Atributo	Nullable	Descripción
culture	No	Código del idioma según lo especificado en http://www.lingoes.net/en/translator/langcode.htm Ejemplo: es-ES (Castellano), ca-ES (Catalán), eu-ES (Euskera), gl-ES (Gallego), en-US (Ingles) Dependiendo del proveedor de bases de datos dará soporte a uno, más de uno o todos los idiomas.
index	No	Los textos en más de un idioma se expresan como un array. Ejemplo: ["castellano", "català", "galego", "euskera", "english"] culture=es-ES y index=0 indica que si queremos obtener el texto en castellano hemos de acceder a la posición 0 del array. culture= ca-ES y index=1 indica que si queremos obtener el texto en catalán hemos de acceder a la posición 1 del array. culture=gl-ES y index=2 indica que hemos de acceder a la posición 2 del array para obtener el texto en gallego. culture=eu-ES y index=3 indica que hemos de acceder a la posición 3 del array para obtener el texto en euskera. culture= en-US y index=4 indica que si queremos obtener el texto en ingles hemos de acceder a la posición 4 del array. Los textos multiidioma se presentan con object value

Ejemplos:
En la carpeta Anexo11_ejemplos/ejemplos/culture se puede encontrar un ejemplo de json representando un idioma.
En la carpeta Anexo11_ejemplos/ejemplos/valor/multiculture podemos encontrar un ejemplo de cómo un mismo texto es expresado en diferentes idiomas.

11. Object Payload

Contiene la información solicitada sobre los elementos, dicha información actualmente se devuelve en formato BC3.


```
{
    "format": "bc3",
    "compression": null,
    "content":
"flZ8UklCIFNwYWlufEZJRUJEQy0zLzIwMjB8UHJlc3RvIDIxLjAzfHxBTlNJfHwyfHx8fA0Kfkx8fEVTU
FxFc3BlY2lmaWNhY2nDs1xURUNcQ2FyYWN0ZXLDrXN0aXF1ZXMgdMOoY25pcXVlc1xQUkVc
Q29uZGljaW9ucyBwcsOodmllc1xFSkVcRXhlY3VjacOzXE1FRFxBbWlkYW1lbnRcQ09OENvbnR
yb2xcU0VHXFNlZ3VyZXRhdCBpIHNhbHV0XEdNQVxHZXN0acOzIGFtYmllbnRhbFxOT1JcTm9yb
WVzIGTCkmFwbGljYWNpw7NcTUFOXE1hbnRlbmltZW50XFZBUlxEaXZlcnNvc1xURVhcVGV4dC
BzZW5zZSDDoG1iaXRcfA0Kfkt8XDJcMlwzXDJcMlwyXDJcXHwwXDBcMFwwXDB8M1wyXFwzX
DNcXDJcMlwyXDJcMlwyXDJcMlxcfA0Kflh8fHBvc3RcQ29udGluZ3V0IHJlY2ljbGF0IHBvc3Rjb25zd
W1cJVxwcmVcQ29udGluZ3V0IHJlY2ljbGF0IHByZWNvbnN1bVwlXGNlXENvc3QgRW5lcmfDqHRp
Y1xNSlxsZXJcQ29kaSBsZXIgZGUgbGEgbGxpc3RhIGV1cm9wZWEgZGUgcmVzaWR1c1xcbXBc
TWF0w6hyaWEgcHJpbWVyYVwlXG1cbWFzc2Fca2dcdlx2b2x1bVxtM1xlQ08yXEVtaXNzacOzIGR
lIENPMlxrZ1x8DQp+Q3wwIyN8fFBzdG8gRXZhIEp8MHwwNDA1MjJ8MHwNCn5EfDAjI3xQNDRD
LURQMkZcMVwwXFA0NDItREZaOFwxXDBcUDQ1MTAtRDU0VFwxXDBcUDQ1OTAtRjhEVVwxX
DBcUEUyMi0zNjdVXDFcMFxQRTQwLTYwQjZcMVwwXFBFTjltOUc3M1wxXDBcUEoyMTAtM1lLT
FwxXDBcUEoyMU0tRklDOFwxXDBcfA=="
}
```


Atributo	Nullable	Descripción
format	No	Formato en el que se encuentra la información de los elementos. Actualmente solo el formato BC3 es soportado.
compression	Si	Algoritmo de compresión utilizado: • null: no se utiliza compresión. • "zip": comprimido con zip. • "tar": comprimido con tar.
content	Si	Información sobre los elementos solicitados expresada en formato BC3 y codificada en base 64. Por lo tanto, hay que realizar una decodificación en base 64 para recuperar el BC3. En el caso de utilizar un algoritmo de compresión primero se ha de realizar la decodificación en base64 y a continuación la descompresión utilizando el algoritmo indicado.

Ejemplos:
En la carpeta Anexo11_ejemplos/ejemplos/payload se puede encontrar un ejemplo sin utilizar algoritmo de compresión y otro con compresión.

12. Object Database

Información sobre la base de datos y su versión.


```
{
```

```
    "name": "nombre del banco",
    "version": "identificador de la versión del banco"
}
```

Atributo	Nullable	Descripción
name	No	Nombre de la base de datos, banco.
version	No	Texto que identifica la base de datos entre todas las bases de datos que tiene el proveedor de bases de datos.

Ejemplo:

```
{
    "name": "BEDEC2020",
    "version": "BEDEC20200112"
}
```

13. Object Global Parameter

```
{
    "name": "parámetro I",
    "id_property": -60,
    "type": "T",
    "globalparametervalues": [object global parameter value1,...,object global parameter valueN]
}
```

Atributo	Nullable	Descripción
name	No	Nombre del parámetro global. Ejemplo: "Ámbito de precios".
id_property	No	Identificador del parámetro global. Cada proveedor de base de datos define cuales son los identificadores de sus parámetros.
type	No	Define de que tipo de datos son los valores asociados a un parámetro global. Los tipos de admitidos son: T – texto, B – booleano, N- numérico, D – Fecha.
globalparametervalues	No	Array con la lista de valores que puede tomar el parámetro global.

Ejemplos:
En la carpeta Anexo11_ejemplos/ejemplos/parametrosbanco podemos encontrar ejemplos de parámetros de banco con diferentes tipos de datos: numéricos, booleanos, fecha y texto.

14. Object Global Parameter Value

Representa cada uno de los valores que puede tomar un parámetro global. Los valores pueden ser texto, numéricos, booleanos y fechas.

```
{
    "id_value": 1169172,
    "cultures": false,
```

```
    "stvalue": ["hormigonera"],
    "dblvalue": null
}
```


Atributo	Nullable	Descripción
id_value	No	Identificador del valor. Cada proveedor de base de datos asigna los identificadores a sus valores.
cultures	No	• true: indica que valor varía según el idioma. Ejemplo: ["hormigonera", "formigonera",...] • false: indica que el valor es siempre el mismo. ["hormigonera"]. **Nota**: En el caso del valor de un parámetro global cultures será siempre **falso** porque al solicitar los parámetros de banco ya hemos especificado en que idioma los queremos.
stvalue	Si	Array solo contiene un texto que corresponde al idioma en que se han solicitado los parámetros de banco.
dblvalue	Si	Si los valores de un parámetro global son numéricos el atributo está informado.

Ejemplos:
En la carpeta Anexo11_ejemplos/ejemplos/parametrosbanco podemos encontrar ejemplos de parámetros de banco con valores de diferentes tipos de datos: numéricos, booleanos, fecha y texto.

15. Object AsyncStatus

Refleja el estado en que se encuentra la tarea.


```
{
    "id": 15729,
    "status": "finished",
    "started": false,
    "finished": true,
    "running": false,
    "queue_position": null,
    "start_time": inicio,
    "end_time": fin,
    "savedresult":true,
    "origen":"adapta tcq"
}
```


Atributo	Nullable	Descripción
id	No	Identificador de la tarea asíncrona.
status	No	• waiting: todavía no se ha comenzado a procesar la solicitud. • starting: se ha comenzado a procesar la solicitud. • finished: se ha finalizado de procesar la solicitud.
started	No	• true indica que la tarea está iniciada. • false no está iniciada.
finished	No	• true indica que la tarea está finalizada. • false no está finalizada.
running	No	• true indica que la tarea está en ejecución.

		• false no está en ejecución.
queue_position	Si	Posición que ocupa en la lista de espera para ser ejecutada.
start_time	No	Fecha y hora en la que se inicia la tarea asíncrona.
end_time	No	Fecha y hora en la finaliza la tarea asíncrona.
savedresult	No	• true: indicar que ya se puede hacer la petición para recuperar la información solicitada. • false: todavía no se puede hacer la petición para recuperar la información solicitada.
origen	No	Texto que indica quien ejecuta la tarea asíncrona.

Ejemplos:
En la carpeta Anexo11_ejemplos/ejemplos/asyncstatus se encuentran ejemplos.

16. Object AsyncTask

Cuando se solicita ejecutar una tarea asíncrona se devuelve este objecto que identifica a la tarea asíncrona.

```
{
    "id": 15729,
    "isAsync": true
}
```

Atributo	Nullable	Descripción
id	No	Identificador de la tarea asíncrona, se utiliza para ir preguntando si la tarea ha acabado.
isAsync	No	Indica que se trata de una tarea asíncrona. Siempre es true.

Ejemplos:
En la carpeta Anexo11_ejemplos/ejemplos/asynctask se encuentran ejemplos.

17. Object Value

Este objecto representa valores de tipo texto, numérico, fecha y booleanos.

```
{
    "cultures": true,
    "stvalue": ["hormigonera", "formigonera", "concret mixer"],
    "dblvalue": null
}
```

Atributo	Nullable	Descripción
cultures	No	true indica que valor varía según el idioma. false indicar que el valor es siempre el mismo para todos los idiomas.
stvalue	Si	Array que contiene una lista de valores, en el caso que cultures sea true cada valor representa lo mismo pero escrito en diferentes idiomas. Si cultures es false el Array puede contener uno o varios valores, pero si contiene varios estos no representan el mismo concepto en diferentes

		idiomas. Habitualmente si culturas es false el array solo contiene un valor. También se usa este atributo para representar fechas.
dblvalue	Si	Informado en el caso de que represente un valor numérico o booleano.

Ejemplos:
En la carpeta Anexo11_ejemplos/ejemplos/valor/boolean se encuentran ejemplos de cómo representar valores booleanos.
En la carpeta Anexo11_ejemplos/ejemplos/valor/date hay ejemplos de cómo representar una fecha tanto en formato corto como con horas, minutos y segundos.
En la carpeta Anexo11_ejemplos/ejemplos/valor/multiculture podemos encontrar un ejemplo de texto en múltiples idiomas.
En la carpeta Anexo11_ejemplos/ejemplos/valor/numerical encontramos ejemplos de representación de valores numéricos con y sin decimal y también valor nulo.

Object Status

Informa de si la petición a un endpoint ha funcionado correctamente o no.

```
{
    "returncode": 0,
    "messages": [
        {
            "code": 1.
            "message": "message1"
        }
        ,...,
        {
            "code": 23.
            "message": "messageN"
        }
    ]
}
```

Atributo	Nullable	Descripción
returncode	No	0 indica que no hay ningún error. 1 indica que el token de seguridad ha caducado y que se debe renovar. Cualquier otro valor indica que se ha producido un error. En caso de error cada proveedor de base de datos asigna sus códigos, en caso de que no haya error todos los proveedores de bases de datos retornan 0.
messages	No	Array que contiene una lista de mensajes. La lista solo contiene mensajes en el caso de que se haya producido un error.
code	No	Código que identifica el mensaje. Cada proveedor de bases de datos asigna sus códigos.
message	No	Texto en inglés que describe el problema.